A Report submitted in partial fulfilment of the regulations governing the award of the Degree of BSc (Honours) Ethical Hacking for Computer Security at the University of Northumbria at Newcastle.

Project Report

Nettynum

A Windows Domain Enumeration Tool

Oliver Morton

10005202

2013/2014

Software Engineering Project

DECLARATIONS

I declare the following:

(1) that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to **ALL** sources be they printed, electronic or personal.

(2) the Word Count of this Dissertation is: 19,103

(3) that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

(5) I have read the Northumbria University/Engineering and Environment Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

SIGNED:

Acknowledgments

Abstract

Enumeration is a key aspect of any computer security testing methodology. As many internal networks utilise a Windows domain it is important to know the information an attacker may be able to obtain by enumerating this domain. Existing tools require a significant amount of direction and information from the user to effectively gather pieces of information. There is a need for a tool which can operate independently of the user to free them to carry out other tasks. An object orientated design methodology is selected and applied in Unified Modelling Language (UML) to design a product to fill this gap. The Python programming language is used to implement the designs creating a functional product which is verified through Alpha testing in a virtual environment against the requirements specification and project objectives, and Beta testing by industry professionals. The paper concludes with a critical evaluation of the product and project process, followed by recommendations for future work both for this tool and other work in the same problem area.

Contents

1 Introduction

1.1 Aims, Objectives and Approach

The aim of this project is to develop an enumeration tool that discovers information from a Windows domain and stores information that could be used during subsequent runs of the application.

This will be accomplished by fulfilling the following nine objectives which also dictate the approach to be taken during the project:

1. To perform a literature review encompassing existing tools, useful information that can be enumerated, methods of enumeration.

In order to fully understand the problem area a literature review will be undertaken. This will include defining enumeration with particular regard to information gathering, identifying services that allow information to be enumerated, and existing tools. The existing tools will be considered in terms of the services they enumerate information from, the particular information they are capable of retrieving, and the degree to which they are automated.

2. Create a list of requirements for the tool based on the literature review.

Following the literature review (and using the information gathered), a list of requirements will then be created. These requirements will help to ensure that the tool that is created meets the needs of the problem area and learns from the features of existing products. The software requirements specification will dictate essential, desirable and excluded requirements as well as describing the interfaces from the product's perspective, the intended user characteristics, assumptions and dependencies.

3. Enhance knowledge of the Windows API

At the start of the project the author has a limited knowledge of the Windows API, however, because of the nature of the project, a detailed knowledge of the Windows API is required. Therefore one of the objectives of this project is to enhance the author's understanding of the API functions so that they might be efficiently utilised to meet the requirements of the application.

4. Enhance skills in the selected development language.

During this project a suitable development language will be selected. The author has experience with several languages that may be appropriate, however the scope of this project will require additional skill in the chosen development language. For this reason the requirement to enhance the author's skills in the selected development language has also been included in the objectives of the project.

5. Create design diagrams for structure and behaviour of the application.

An appropriate design methodology will be selected and used to create design diagrams which describe the structure and behaviour of the application which will meet the requirements specification.

6. Implement the designs in the chosen development language utilising a source code management solution and following the MS SDL.

The design diagrams will then be implemented in the selected development language using the source code management solution Git to maintain an offsite backup as well as full revision history (Chacon, 2009). The Microsoft Secure Development Lifecycle will be followed to aid in creating a secure product by considering security concerns throughout the project's lifecycle.

7. Develop and carry out a plan to test the application against the requirements specification.

In order to determine the extent to which the application meets its requirements a suitable testing strategy will be employed which tests the application at every level between unit and acceptance.

8. Undertake an analysis of the results of the testing to identify functional and non-functional results.

Once the testing has been performed the results will be analysed to identify the functional and non-functional elements of the application, which will indicate its fitness for purpose.

9. Generate user documentation for the application.

The final objective is to create user documentation for the application. This must be accurate and detailed but easy to understand by a new user.

1.2 Product Overview

The purpose of the proposed tool is to enable computer security professionals to identify the information an attacker may be able to gather from the Windows domain so preventative action can be more precisely targeted. In particular this tool is intended to be used during internal security assessments to maximise information discovery to inform other areas of the assessment. There is a large amount of information that can be enumerated from the Windows domain including the domain accounts policy which dictates the lockout threshold and duration, groups and group membership which are used to restrict permissions, and user account information.

The application should be designed to require as little input from the user as possible in order to free them to continue with other tasks. A key feature will therefore be automated operation; starting from zero knowledge of the domain or network the application should find the domain name and controller and enumerate all possible information. The user should also be able to target the automated enumeration to restrict the application's operation, for example the user could supply the domain name and domain controller in order to prevent these from being enumerated, causing the application to skip forward to the enumeration of information from the domain controller.

While the focus of this tool is on automated operation, there is always the possibility that the user will need to enumerate a specific piece of information. Therefore the application is able to be manually directed to retrieve one piece of information, such as a list of group members.

Since the purpose of this product is to increase the ease with which an assessment can be conducted the output format is an important characteristic. It must be human readable but also easy to parse by other scripts, this is so that the information that has been gathered can be immediately understood by the user and quickly fed to other tools to continue with the assessment.

1.3 Problem Context

The Microsoft Windows operating system is extremely ubiquitous in both home and corporate environments to a point where it has been “determined Microsoft had a monopoly in the market” (United States v. Microsoft - Review of the Final Judgements by the United Steates and New York Group, 2007).

A large amount of information can be enumerated from a Windows based network including: domain controllers, services running on hosts, logged on users, installed software, shared resources, user groups, policies, user rights, and user information (McClure, et al., 2009). This information may appear harmless, however it can lead to a complete compromise of the network and should therefore be eliminated from the network (McClure, et al., 2009).

A number of services can be used to gather this information from a Windows domain and several tools exist to gather specific pieces. However during the author’s time in industry it was found that several of these tools were required to be used in succession to retrieve the required information and each required some level of direction from the user.

The application will be targeted at security professionals who need to discover what information can be enumerated from their own, or a client’s, system.

1.4 Tools and Techniques

In order to meet the objectives of the project a number of tools and techniques will be used.

Throughout the project the Microsoft Secure Development Lifecycle (MS SDL), will be followed to help ensure that the application produced is as secure as possible.

At the design phase an object orientated design methodology will be applied using the Unified Modelling Language (UML) to create designs of an application that meets the specified requirements with advanced notions of object orientation including polymorphism and inheritance. A number of design patterns will be applied where appropriate to solve problems that commonly occur within software engineering as they are proven and accepted.

These designs will be implemented in the Python programming language which supports object orientation and will later be shown to be a suitable choice for this project. During the implementation phase Git will be used to manage the source code of the project, providing offsite backup and full revision history.

In order to test the application a virtual environment will be configured that represents the core of a Windows domain, namely the domain controller. Using virtual machine software two Windows servers will be installed and configured with the active directory service to create two domain controllers for a single domain.

The application will be tested against the requirements specification and designs to identify any errors within the code using an appropriate testing strategy that employs unit, integration, system and acceptance testing through a manual and automated approach.

2 Literature Review

2.1 Defining Enumeration

There are many definitions of enumeration depending upon context. A general dictionary definition of enumerate, from which the word enumeration is derived, is “to mention separately or in order; name one by one; list” (William Collins and Sons Co. Ltd, 1987). Chi-Liang Ni also provides a general definition: “Enumeration is a process for enclosing, labelling and numbering individual entities for further linking or referring to the related entities in one-to-one, one-to-many, and many-to-many relationships” (Chi-Liang Ni, 1997).

Each area of computing has its own slightly more refined definition for enumeration. When discussing USB devices, enumeration may be defined as “the mechanism by which a USB host determines the status, configuration, and capabilities of an inserted USB device” (Davis, 2013). Birkholz defines service enumeration as “identifying what is running on listening ports” and host enumeration as making an “accurate guess at OS and version” (Birkholz, 2002). Also, in terms of web application security, valid user identifiers which should be further investigated are enumerated (Stuttard & Pinto, 2008).

Within computer security the process of enumeration is an information gathering technique that uses active connections to a system and directed queries to obtain information from previously identified services (Scambray & McClure, 2008).

However, for the purposes of this project, ‘enumeration’ will be defined as: the act of gathering information from a system or service utilising active connections and queries.

Enumeration is a key aspect of many computer security testing methodologies including the Open Source Security Testing Methodology Manual (OSSTMM) (ISECOM, 2010) and is described in much of the literature (McClure, et al., 2009; Scrambray & McClure, 2008; Stuttard & Pinto, 2008, O’Dea, 2003). By learning as much as possible about the services that have been exposed to the network a complete view of the environment is obtained, this allows services with vulnerabilities to be targeted (O'Dea, 2003). It allows auditors to retrieve the information they require without the need to interrupt the network administrators (Melber, 2011).

2.2 Services to Enumerate

Information can be enumerated from practically every service that is in common use on an internal network. By querying information from open ports that have been discovered it is usually possible to retrieve information such as version information for the service and server and a list of valid usernames for services such as File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), Trivial File Transfer Protocol (TFTP), Hypertext Transfer Protocol (HTTP) and finger (McClure, et al., 2009).

The Microsoft RPC Endpoint Mapper (MSRPC) service runs on TCP Port 135 allowing data exchange and invocation of functionality in different processes on the same computer, a computer on the local area network (LAN) or across the internet (Microsoft, 2003). It can yield information about the applications and services available on the target machine along with IP addresses for each interface (McClure, et al., 2009). This information can be used to identify services and identify hosts that are multihomed (i.e. have more than one network interface and therefore more than one IP address).

The Windows Internet Name Service (WINS) provides a distributed database that maps NetBIOS names to IP addresses. WINS is required for versions of the Microsoft Windows operating system prior to Windows 2000 whereas later versions of Windows uses the Domain Name System (DNS). However WINS is enabled by default in all currently supported versions of Windows to accommodate legacy operating systems (Microsoft, 2012). The NetBIOS Name Service (NBNS) provides a means of resolving NetBIOS names to the mapped IP address in the WINS database. The NetBIOS Name Service (NBNS) runs on UDP port 137 and can be used to enumerate Windows workgroup domain and host names on the local network segment, or, if NBNS is routed over TCP/IP, the entire network (McClure, et al., 2009).

Server Message Block (SMB) protocol runs on TCP port 139 (NetBIOS Sessions) and 445 (SMB over raw TCP/IP), providing a means to share resources across a network. Once a session has been established it is possible to gather a wealth of information including network information, shares, users, groups, registry keys, and account policies (Scambray & McClure, 2008). Traditionally it has been possible to utilise 'null' credentials, i.e. a blank username and password, to establish a session with a Windows host (Veerasamy, 2009), however since Windows Server 2008 the default configuration has either completely prevented a null session from being established or restricted the information that can be obtained. Nevertheless null sessions are still allowed on many networks due to legacy servers which require null sessions in order to connect to the domain controller, authenticate users and carry out other operations. Skoudis (2013) also demonstrates that any user credentials can be used in place of null credentials to establish a session and posits that it is common to gain standard user credentials toward the beginning of a security assessment and the SMB service should therefore not be overlooked.

Allen, et al. (2006) describe Microsoft Active Directory (AD), it is built on top of Windows Server, enabling administrators to manage enterprise-wide information efficiently from a central repository. It contains information about users, groups, computers, printers, applications and services; and can limit access to this information. AD is reliant on the Domain Name System (DNS), unlike its predecessor, Windows NT, which was reliant on Windows Internet Naming Service (WINS). As DNS is an open standard it became ubiquitous across the internet, unlike WINS which is proprietary and was typically only used in Windows NT environments (Allen, et al., 2006). This integration with DNS allows DNS queries to be used to gather a list of Domain Controllers and servers running specific services.

2.3 Existing Tools

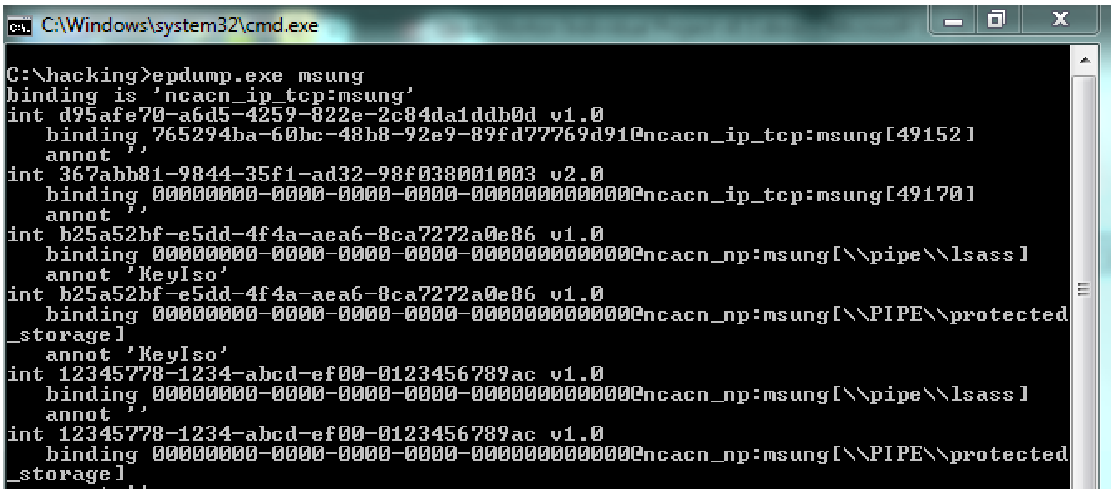
A number of tools exist which can be used to enumerate information from one or more of these services given some prerequisite information. These operate entirely on the command line with a few notable exceptions.

2.3.1 Microsoft RPC Endpoint Mapper (MSRPC)

There are a number of tools that enumerate information from MSRPC the most common of which are outlined below.

2.3.1.1 epdump

Paul Leach of Microsoft released a simple Windows tool called `epdump` in 1997 to identify the IP address and port number that a service is listening on along with the named pipes and SPX bindings (Cooper, 1997). The advantages of this tool for enumeration are that the user only needs to specify a target to enumerate in order to retrieve the information. However it can only enumerate one host at a time, it does not provide any 'help' output to indicate the required command line arguments, and has not been revised since initial release. Another difficulty with this tool is that the output is difficult both for a person to read and for a program to parse as it is spread over multiple lines (see Figure 2.1).

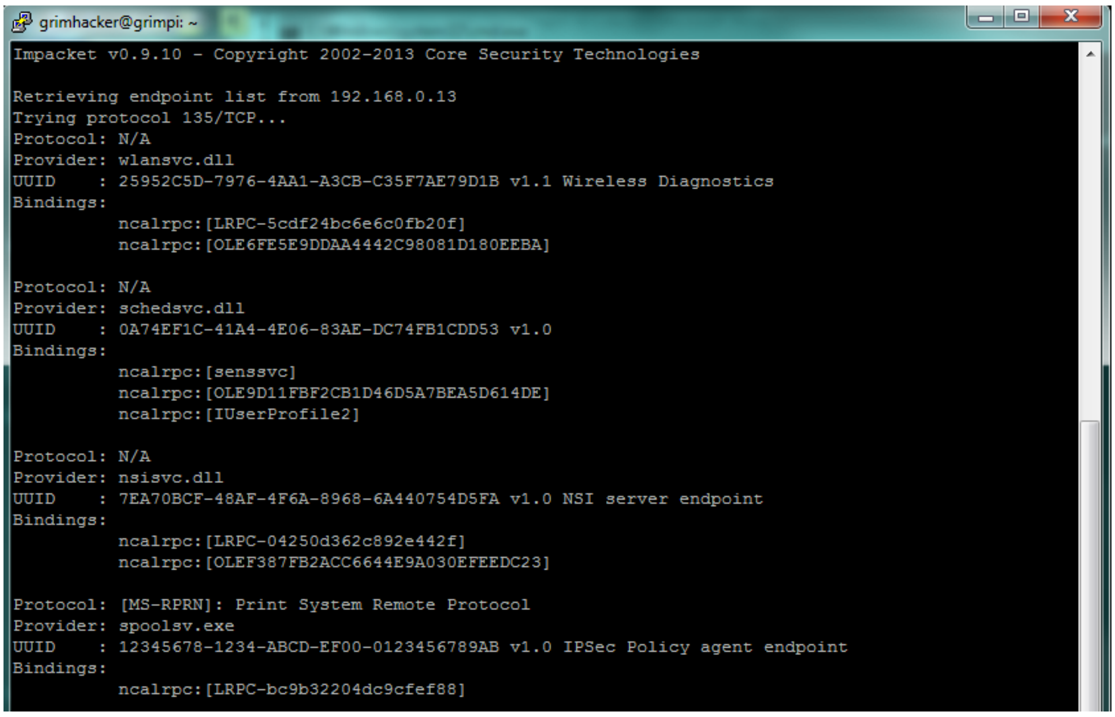


```
C:\hacking>epdump.exe msung
binding is 'ncacn_ip_tcp:msung'
int d95afe70-a6d5-4259-822e-2c84da1ddb0d v1.0
   binding 765294ba-60bc-48b8-92e9-89fd77769d91@ncacn_ip_tcp:msung[49152]
   annot ''
int 367abb81-9844-35f1-ad32-98f038001003 v2.0
   binding 00000000-0000-0000-0000-000000000000@ncacn_ip_tcp:msung[49170]
   annot ''
int b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86 v1.0
   binding 00000000-0000-0000-0000-000000000000@ncacn_np:msung[\\pipe\\lsass]
   annot 'KeyIso'
int b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86 v1.0
   binding 00000000-0000-0000-0000-000000000000@ncacn_np:msung[\\PIPE\\protected
_storage]
   annot 'KeyIso'
int 12345778-1234-abcd-ef00-0123456789ac v1.0
   binding 00000000-0000-0000-0000-000000000000@ncacn_np:msung[\\pipe\\lsass]
   annot ''
int 12345778-1234-abcd-ef00-0123456789ac v1.0
   binding 00000000-0000-0000-0000-000000000000@ncacn_np:msung[\\PIPE\\protected
_storage]
```

Figure 2.1: epdump output

2.3.1.2 RPCDump.py

`RPCDump.py` from CoreLabs is an open source Python script that lists services that are available. A benefit of using this tool over `epdump` is that it allows queries to be directed over ports and protocols such as 80/HTTP 445/SMB, 139/SMB, 135/TCP and 135/UDP (McClure, et al., 2009). Other advantages include a slightly more legible output than the alternative (see Figure 2.2), authentication using either username and password or a password hash, and a command line help text. However like `epdump` it can only enumerate one host at a time, and the output, although readable is not parseable because it is spread over multiple lines.

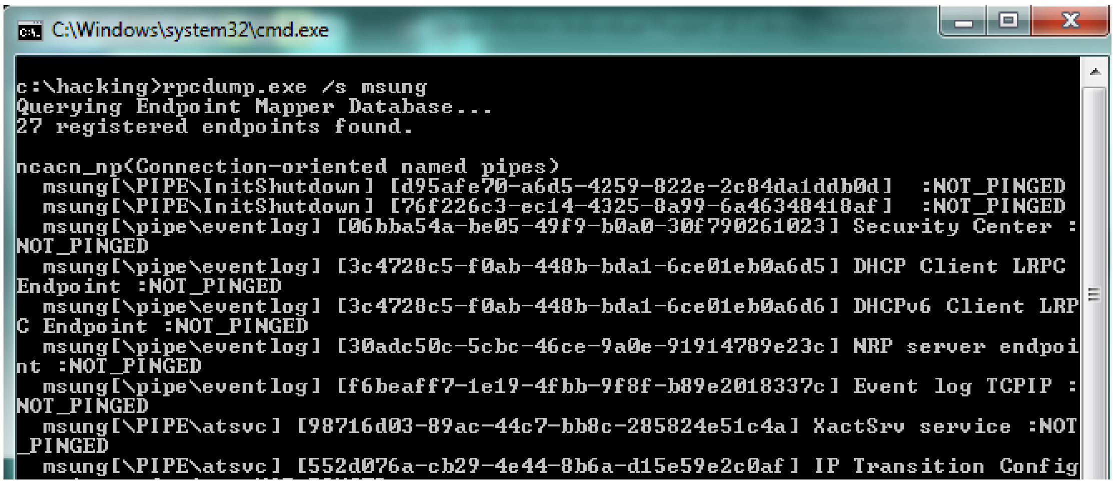


```
Impacket v0.9.10 - Copyright 2002-2013 Core Security Technologies

Retrieving endpoint list from 192.168.0.13
Trying protocol 135/TCP...
Protocol: N/A
Provider: wlansvc.dll
UUID    : 25952C5D-7976-4AA1-A3CB-C35F7AE79D1B v1.1 Wireless Diagnostics
Bindings:
          ncalrpc:[LRPC-5cdf24bc6e6c0fb20f]
          ncalrpc:[OLE6FE5E9DDAA4442C98081D180EEBA]

Protocol: N/A
Provider: schedsvc.dll
UUID    : 0A74EF1C-41A4-4E06-83AE-DC74FB1CDD53 v1.0
Bindings:
          ncalrpc:[senssvc]
          ncalrpc:[OLE9D11FBF2CB1D46D5A7BEA5D614DE]
          ncalrpc:[IUserProfile2]

Protocol: N/A
Provider: nsisvc.dll
UUID    : 7EA70BCF-48AF-4F6A-8968-6A440754D5FA v1.0 NSI server endpoint
Bindings:
          ncalrpc:[LRPC-04250d362c892e442f]
          ncalrpc:[OLEF387FB2ACC6644E9A030EFEEDC23]

Protocol: [MS-RPRN]: Print System Remote Protocol
Provider: spoolsv.exe
UUID    : 12345678-1234-ABCD-EF00-0123456789AB v1.0 IPSec Policy agent endpoint
Bindings:
          ncalrpc:[LRPC-bc9b32204dc9cfef88]
```

Figure 2.2: RPCDump.py output

2.3.1.3 RPCDump.exe

`RPCDump.exe` from the Microsoft Resource Kit can also be used to enumerate information from the MSRPC service, it too requires the user to specify the host to enumerate and lists the interfaces and services available in a similar fashion to `epdump` (Scambray & McClure, 2008). The advantages of this tool include a variable level of verbosity, command line help text, and enumeration over multiple protocols. The disadvantages are that the output is difficult to read and parse (see Figure 2.3), and this tool has not been developed for some time.



```
c:\hacking>rpcdump.exe /s msung
Querying Endpoint Mapper Database...
27 registered endpoints found.

ncacn_np(Connection-oriented named pipes)
  msung[\PIPE\InitShutdown] [d95afe70-a6d5-4259-822e-2c84da1ddb0d]  :NOT_PINGED
  msung[\PIPE\InitShutdown] [76f226c3-ec14-4325-8a99-6a46348418af]  :NOT_PINGED
  msung[\pipe\eventlog] [06bba54a-be05-49f9-b0a0-30f790261023] Security Center :
NOT_PINGED
  msung[\pipe\eventlog] [3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5] DHCP Client LRPC
Endpoint :NOT_PINGED
  msung[\pipe\eventlog] [3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6] DHCPv6 Client LRP
C Endpoint :NOT_PINGED
  msung[\pipe\eventlog] [30adc50c-5cbc-46ce-9a0e-91914789e23c] NRP server endpoi
nt :NOT_PINGED
  msung[\pipe\eventlog] [f6beaff7-1e19-4fbb-9f8f-b89e2018337c] Event log TCPIP :
NOT_PINGED
  msung[\PIPE\atsvc] [98716d03-89ac-44c7-bb8c-285824e51c4a] XactSrv service :NOT
_PINGED
  msung[\PIPE\atsvc] [552d076a-cb29-4e44-8b6a-d15e59e2c0af] IP Transition Config
```

Figure 2.3: rpcdump.exe output

2.3.2 NetBIOS Name Service (NBNS)

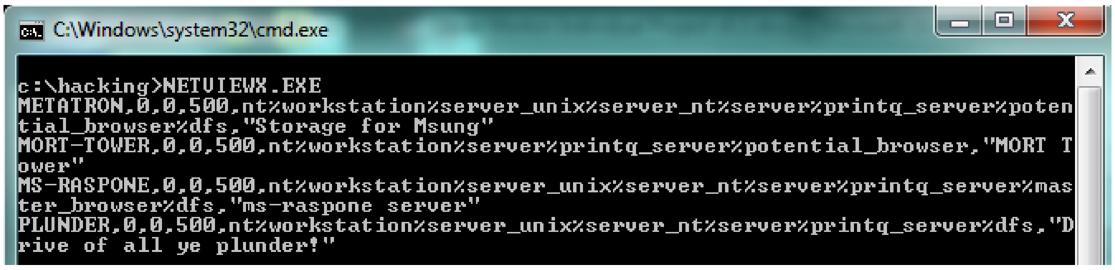
The following tools enumerate information from NBNS.

2.3.2.1 net view & nltest

The `net view` command is built into the Windows operating system and can be used to enumerate information workgroup domain and host names. Another utility from the Microsoft Resource Kit, `nltest`, can be used to identify a list of domain controllers for a specified domain name. (McClure, et al., 2009). These utilities have advantages which include a parseable and readable output, a command line help output, and continuing support from Microsoft. There is no significant disadvantage to these tools for this purpose.

2.3.2.2 netviewx

Lauristen (1999) released the closed source tool `netviewx` which is similar to `net view` but also allows identification of servers with specific services. The output of this tool is delimited by commas and new lines with the intention of making it easily parsable however it is not very readable (see Figure 2.4).

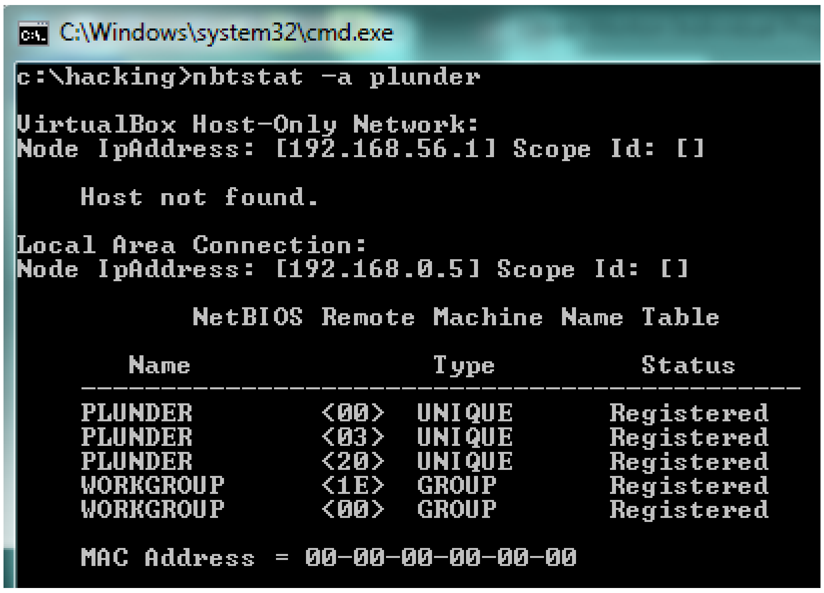


```
c:\hacking>NETVIEWX.EXE
METATRON,0,0,500,nt%workstation%server_unix%server_nt%server%printq_server%poten
tial_browser%dfs,"Storage for Msung"
MORT-TOWER,0,0,500,nt%workstation%server%printq_server%potential_browser,"MORT T
ower"
MS-RASPONE,0,0,500,nt%workstation%server_unix%server_nt%server%printq_server%mas
ter_browser%dfs,"ms-raspone server"
PLUNDER,0,0,500,nt%workstation%server_unix%server_nt%server%printq_server%dfs,"D
rive of all ye plunder!"
```

Figure 2.4: netviewx output

2.3.2.3 nbtstat

Another tool built into the Windows operating system is `nbtstat`, this tool enumerates the system name, the domain name, any running services, media access control (MAC) address and any logged in users, of a system specified by either an IP address or a system name (McClure, et al., 2009). This tool has several disadvantages. The output uses a NetBIOS service code to identify each entity, for example `<00>` indicates the Workstation service computer name; this requires users of this tool to memorise these service codes before they can effectively use the output. It is also limited to enumerating information from one host at a time, and the output is not easily parseable (see Figure 2.5). The benefits of this tool include: continued support from Microsoft, readable output, and clear command line help output.

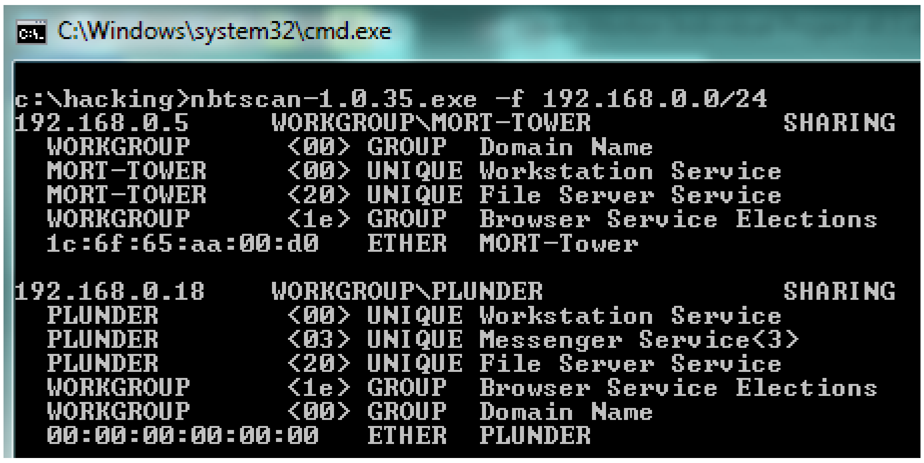


```
C:\Windows\system32\cmd.exe
c:\hacking>nbtstat -a plunder

VirtualBox Host-Only Network:
Node IpAddress: [192.168.56.1] Scope Id: []

    Host not found.

Local Area Connection:
Node IpAddress: [192.168.0.5] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    PLUNDER        <00>  UNIQUE      Registered
    PLUNDER        <03>  UNIQUE      Registered
    PLUNDER        <20>  UNIQUE      Registered
    WORKGROUP      <1E>  GROUP       Registered
    WORKGROUP      <00>  GROUP       Registered

    MAC Address = 00-00-00-00-00-00
```

Figure 2.5: nbtstat output

2.3.2.4 nbtscan

Some of the disadvantages of `nbtstat` have been addressed in a tool called `nbtscan`. This tool will perform the same enumeration as `nbtstat` but does so across an entire network range and translates the NetBIOS service codes in its output (Scambray & McClure, 2008) making it readable. This tool also has some disadvantages which include: the output is not parseable and the tool has not been worked on since 2008.



```
C:\Windows\system32\cmd.exe

c:\hacking>nbtscan-1.0.35.exe -f 192.168.0.0/24
192.168.0.5      WORKGROUP\MORT-TOWER          SHARING
  WORKGROUP       <00> GROUP  Domain Name
  MORT-TOWER      <00> UNIQUE Workstation Service
  MORT-TOWER      <20> UNIQUE File Server Service
  WORKGROUP       <1e> GROUP  Browser Service Elections
  1c:6f:65:aa:00:d0    ETHER  MORT-Tower

192.168.0.18     WORKGROUP\PLUNDER             SHARING
  PLUNDER         <00> UNIQUE Workstation Service
  PLUNDER         <03> UNIQUE Messenger Service<3>
  PLUNDER         <20> UNIQUE File Server Service
  WORKGROUP       <1e> GROUP  Browser Service Elections
  WORKGROUP       <00> GROUP  Domain Name
  00:00:00:00:00:00    ETHER  PLUNDER
```

Figure 2.6: nbtscan output

2.3.3 Server Message Block (SMB)

There are a number of tools for SMB enumeration, some of the most popular are outlined below.

2.3.3.1 net view, nltest & req

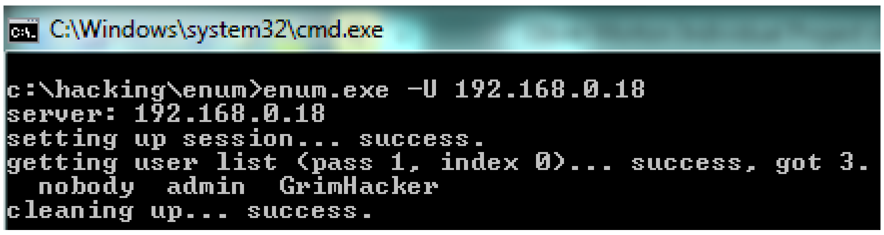
Once a session has been established the utilities built in to Windows can be used to gather even more information (Scambray & McClure, 2008). The `net view` utility on Windows can enumerate share names and comments on an individual, specified remote system. `Nltest` can enumerate the domains that a server trusts through a session. The `reg` utility can be used to enumerate registry keys on a remote host, although by default most of the registry is restricted to administrators, it is possible to retrieve the applications that start up with Windows. As previously stated, these tools have the advantage of continuing Microsoft support, readable/parseable output, and help output which specifies the arguments. The primary disadvantage of using these tools in this manner is that the user must handle creating and destroying sessions.

2.3.3.2 Dumpsec

Singleton (2008) describes `Dumpsec` as a utility that allows an IT auditor to access a Microsoft Server running Active Directory and print out users and access rights. The utility has a simple graphical interface (Berghel & Hoelzer, 2005) and can enumerate all basic account information including the username, whether the account is locked out or disabled, if a password is required, the account comment, account type and home directory (Melber, 2005). However Melber (2005) described this tool as falling short when it comes to enumerating the more advanced user account properties and will give false results for account properties that have changed technical functionality between Windows NT and Windows Active Directory. Melber also notes that "there is no tool that can decrypt the complex array of settings and possibilities that exist within a Remote Access Policy. This must be done manually" (Melber, 2005). `Dumpsec` can also be used from the command line using a large number of lengthy options and gather policies, user rights and services, but these items are restricted by default on Windows (Scambray & McClure, 2008).

2.3.3.3 Enum

Berghel & Hoelzer (2005) state that `Enum` is a freely available utility that establishes null sessions and enumerates shares across networks and also offers password and user name brute force capabilities. Scambray & McClure (2008) expound this description, indicating that `Enum` was the first tool to incorporate nearly every SMB enumeration feature in a single utility. The primary advantage of this tool over many of the alternatives is that it has the capability to enumerate a number of categories of information, it also handles authentication for the user (utilising null credentials by default) and outputs in a readable form. There are also disadvantages to this tool, it has not been developed in some time, and the output is not easily parseable (see Figure 2.7).



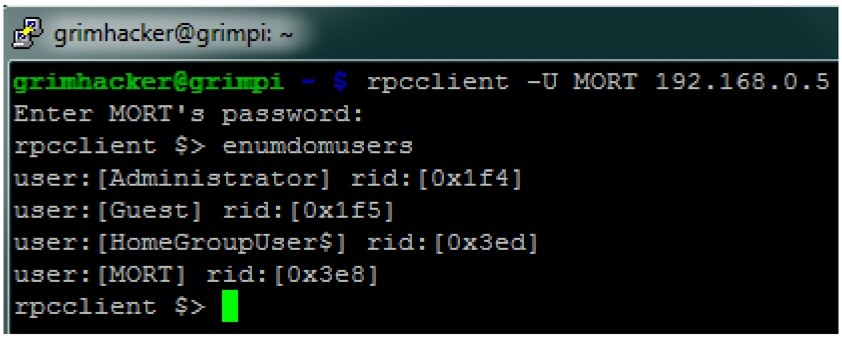
```
c:\hacking\enum>enum.exe -U 192.168.0.18
server: 192.168.0.18
setting up session... success.
getting user list (pass 1, index 0)... success, got 3.
  nobody  admin  GrimHacker
cleaning up... success.
```

Figure 2.7: enum output

2.3.3.4 rpcclient

The `rpcclient` from the `Samba Suite` is a Linux tool for executing client side MS-RPC functions, it was initially developed to test the MS-RPC functionality in Samba itself and has undergone several stages of development and stability (Samba

Team, 2010). Skoudis (2013) provides a description of how `rpcclient` can be utilised to perform SMB enumeration using null or standard user credentials. Unlike other tools, `rpcclient` can run as an interactive program with usability benefits such as tab auto completion, but also offers the capability to run a single command. A plethora of commands to enumerate specific information are available and detailed within the command line help output. Another advantage of this tool is that it is relatively easy to parse and read the output (see Figure 2.8). However this tool was not specifically developed for enumeration so also offers functions to alter information, and does not provide a means to automate gathering a variety of categories of information.



```
grimhacker@grimpi ~ $ rpcclient -U MORT 192.168.0.5
Enter MORT's password:
rpcclient $> enumdomusers
user:[Administrator] rid:[0x1f4]
user:[Guest] rid:[0x1f5]
user:[HomeGroupUser$] rid:[0x3ed]
user:[MORT] rid:[0x3e8]
rpcclient $>
```

Figure 2.8: rpcclient output

2.3.3.5 Enum4Linux

`Enum4linux` is a Linux tool written in Perl that clones the functionality of `enum` using tools from the Samba suite including rpcclient (McClure, et al., 2009). One advantage of this tool is that it provides options for conducting all simple enumeration (user list, share list, group and member list, rid cycling, operating system information, and NetBIOS Name lookup). However it requires the user to specify the host from which it enumerates information.

2.3.3.6 NetE

`NetE` is another tool that is capable of retrieving a large amount of information using the Windows Net* API functions, the latest version was released by 'Sir Dystic' in the year 2000. Unlike most other tools it offers a command line option to perform all enumeration however it is limited to a single host specified by the user (Scambray & McClure, 2008). Another key disadvantage of this tool is that it has not been updated in many years and does not run on modern versions of Windows without altering the source code and recompiling.

2.3.3.7 NBTEnum

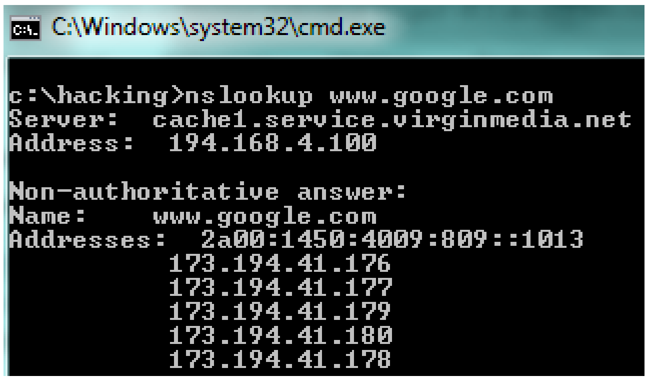
Another 'all-in-one' enumeration tool that also has password brute force capabilities and utilises NetBIOS sessions is `NBTEnum`. Among the advantages of this tool are an extensive yet easy to read HTML output and an option to allow all basic enumeration to be conducted autonomously when provided with a target and credentials. One drawback of this tool is that it has not been updated since 2006 and the product's homepage no longer exists implying that the tool will not be updated in the future. It also requires the user to specify either a single host or a file containing a list of hosts.

2.3.3.8 user2sid & sid2user

The security identifier (SID) is a variable length numeric value issued to an NT Family system at installation. The SID can be discovered from a username using a tool called `user2sid` and be used to discover a username using `sid2user` (McClure, et al., 2009). These tools are simple and have only one function, however they can be used in a loop to discover all users on a target system. The disadvantages of these tools include that they have not been developed since 1998 and while the output is easily readable it is not easily parsed.

2.3.3.9 NSlookup

One cross platform tool that is can be used for enumeration of information from the DNS service is `NSlookup`. This program comes with many TCP/IP software packages; it can be used to examine entries in the DNS database that pertain to a particular host or domain (Kessler & Shepard, 1997). `NSlookup` can be used to execute a zone transfer, which retrieves all the records from a DNS server (Scambray & McClure, 2008). Individual lookups for any kind of DNS record can also be performed. For example 'A' records for a given domain name can be queried to identify domain controllers (Allen, et al., 2006). A shortcoming of this tool is that it the output is not parseable (see Figure 2.9). However the output is readable and the tool can be run either interactively or as a command line utility.



C:\Windows\system32\cmd.exe

```
c:\hacking>nslookup www.google.com
Server:  cache1.service.virginmedia.net
Address:  194.168.4.100

Non-authoritative answer:
Name:    www.google.com
Addresses:  2a00:1450:4009:809::1013
          173.194.41.176
          173.194.41.177
          173.194.41.179
          173.194.41.180
          173.194.41.178
```

Figure 2.9: nslookup output

2.3.4 Summary

Table 1 below summarises the information that each of the existing tools can enumerate.

	epdump	rpcdump.py	net view	netviewx	nbtstat	nbtscan	nltest	reg	dumpsec	enum	rpcclient	enum4linux	NBTEnum	user2sid	sid2user	nslookup
Domain Name			✓		✓	✓	✓									✓
Domain Controller					✓	✓	✓				✓					✓
Windows Hosts			✓		✓	✓				✓						✓

	epdump	rpcdump.py	net view	netviewx	nbtstat	nbtscan	nltest	reg	dumpsec	enum	rpcclient	enum4linux	NBTEnum	user2sid	sid2user	nslookup
Services on Host	✓	✓			✓	✓					✓					
Hosts running Service				✓									✓			✓
Usernames									✓	✓	✓	✓	✓	✓	✓	
Groups and Membership									✓	✓	✓	✓				
User Account Information									✓		✓					
Account locked out									✓		✓		✓			
Account Disabled									✓		✓					
Password Policy									✓	✓	✓	✓	✓			
Account Policy									✓	✓	✓	✓	✓			
Registry keys								✓			✓					
Share Information			✓						✓	✓	✓	✓	✓			
Platform [**W**indows, **L**inux, **M**ultiple]	W	M	W	W	W	W	W	W	W	W	L	L	W	W	W	M
Domain Trusts							✓				✓					
MAC Address				✓	✓	✓										
Latest Release Date																
Requires Authentication			✓					✓	✓	✓	✓	✓	✓	✓	✓	
Handles Authentication										✓	✓	✓	✓			

	epdump	rpcdump.py	net view	netviewx	nbtstat	nbtscan	nltest	reg	dumpsec	enum	rpcclient	enum4linux	NBTEnum	user2sid	sid2user	nslookup
Automated						✓			✓	✓		✓	✓			
GUI									✓							

Table 1 Existing Tools

Many of the existing tools require the user to deal with authentication; do not provide an automated means of enumeration and require user input to obtain a large amount of information. Most commonly the user must specify a host IP address and domain name from which information will be enumerated. In the author's experience this is usually accomplished by running several tools in succession to gather the required information for enumeration from the Windows domain controller.

The majority of the existing tools were also developed some time ago and have not been updated since despite some significant changes to the Windows domain environment.

A common drawback of the existing tools is the unparseable output. This requires the user to read through and extract the required information piece by piece.

Very few of these tools offer an automated option to gather all the information, instead requiring the user to explicitly state what information should be returned.

These disadvantages will be addressed in this project by creating a new tool for Windows domain enumeration which does not suffer from these flaws.

The primary benefit of the new tool is that it requires very little input from the user to gather a large amount of information and can therefore be left to run unattended. This is in contrast to other tools which require direction from the user and often for several tools to be run consecutively to produce the same information.

The new tool will also be designed for and tested on the latest versions of the Windows Domain ensuring that the tool runs correctly against these platforms. This gives it an advantage over existing tools which have not been developed since the latest platforms have been released.

Another significant advantage of this tool will be the output format. Unlike many existing tools, the output will be in a form that is readable by a person but also parseable by a machine. This greatly increases the usability of the output because it can be interpreted by other programs and scripts which perform further automated tasks, while still allowing the user to manually interpret the information with ease.

By using official Windows API this tool will also have the benefit of running successfully on all versions of Windows and greater stability because the API functions abstract messages and commands which may change at any time within the operating system.

3 Commentary on Requirements Specification

3.1 Requirements

Multiple requirements have been specified for this project and are explained in detail in Appendix B which follows the IEEE Guidelines for a Software Requirements Specification.

The key requirements are included below.

3.1.1 Essential

The following requirements must be met by the product.

1) Run on a machine that is not a member of a domain
2) Run on a Microsoft Windows machine
3) Identify domain names on a local area network
4) Identify domain controllers
5) Retrieve a list of users from the active directory
6) Retrieve a list of groups from the active directory
7) Retrieve a list of group members from the active directory
8) Retrieve user account information from the active directory
9) Retrieve the domain accounts policy (Lockout: duration, threshold, observation window. Password: length, character set)
10) Perform automated enumeration
11) Output in a parseable and human readable format
12) Provide a help / usage guide

3.1.2 Desirable

The following requirements maybe met by the product.

1) Perform directed enumeration
2) Identify services running on hosts
3) Identify domain trusts
4) Retrieve a list of users from a local host
5) Retrieve a list of groups from a local host
6) Retrieve a list of group members a local host
7) Retrieve information user account information from a local host
8) Retrieve the local accounts policy (Lockout: duration, threshold, observation window. Password: length, character set)
9) Retrieve a list of shares from a local host

3.1.3 Excluded

The product will not have the following requirements.

1) Conduct password guessing / brute force attacks
2) Include an interactive prompt
3) Include a graphical user interface (GUI)
4) Run on a linux based platform

3.2 Design Methodology

There are several different design methodologies which could be suitable for this project. This section will outline these methodologies and justify the selection.

3.2.1 Methodologies

3.2.1.1 Functional Decomposition

Functional decomposition is “the divide-and-conquer technique applied to programming” (Bergland, 1981), essentially the problem is broken down into its constituent functions. The design process can be expressed as the following steps:

1. Clearly state the intended function.
2. Divide, connect, and check the intended function by re-expressing it as an equivalent structure of properly connected sub functions each solving part of the problem.
3. Divide, connect and check each sub function far enough to feel comfortable.

Bergland (1981) states there are several problems involved in applying this technique. One of these problems is unpredictability and variability because although decomposition must be performed it is not specified what it should be performed with respect to, i.e. time order, data flow, logical groupings, etc., and therefore each programmer is likely to take a different approach to solving the same problem. “The choice of ‘what to decompose with respect to’ has a major effect on the ‘goodness’ of the resulting program and is therefore the subject of much controversy” (Bergland, 1981).

However the major advantage of this methodology is its general applicability i.e. it is a suitable methodology for a wide variety of projects.

3.2.1.2 Data Flow Design

In its simplest form, data flow design is functional decomposition with respect to data flow, each block of the application has an input stream and an output stream which can be linked together to form the computational process (Bergland, 1981). It models the flow of data through a system and at the design level specifies the system structure in terms of the calling relationship among modules (Kitchenham & Carn, 1990).

However this decomposition tends to lead to a network of programs each of which process the data, instead of a hierarchy of programs.

The four basic steps of data flow design are:

1. Model the program as a data flow graph.
2. Identify afferent (input), efferent (output), and central transform elements.
3. Factor the afferent, efferent, and central transform branches to form a hierarchical program structure.
4. Refine and optimise.

3.2.1.3 Data Structure Design

Data structure design methods are based on analysing the structure of input and output data (Kitchenham & Carn, 1990). As the program structure is derived from the data structure, the different levels of the hierarchy tend to be a 'is composed of' relationship (Bergland, 1981). This relationship forms a firm base for modelling the problem because it does not change during the execution of the program. This design methodology results in consistency because two different programmers should also create program structures that are essentially identical since it is based on the structure of the data (Bergland, 1981).

The design strategy for this methodology is outlined as:

1. Form a system network diagram that models the problem environment.
2. Define and verify the data-stream structures.
3. Derive and verify the program structures
4. Derive and allocate the elementary operations
5. Write the structure text and program text.

The major problem with this methodology is that although it is clear how to apply it to small problems, the 'correct' method for extending it to large system problems is less clear (Bergland, 1981).

3.2.1.4 Object Orientated Design

Object orientated design is based on identifying the objects of interest in a system and treating them as abstract data types which have associated operations that interrogate and alter their state (Kitchenham & Carn, 1990).

Object orientation allows for polymorphism, this means that a many different behaviours could result from the same call depending upon the object (Shalloway & Trott, 2004). For example, consider two classes of 'shape', 'square' and 'triangle'. They both require a 'draw' method, but how they are implemented may be very different. However the calling program does not need to know the implementation details or even that there are different types of shape, it simply needs to have a method to instruct all the shapes to draw themselves. Therefore an abstracted class 'shape' from which 'square' and 'triangle' are derived, is used which defines a public method 'draw'.

By encapsulating functionality in an object and presenting a public interface to this functionality, changes to the implementation for a specific object do not impact the caller. This reduces bugs in the program, makes an objects internal behaviour transparent to other objects and prevents bugs resulting from changes to functions therefore increasing the ease with the code can be maintained.

3.2.1.5 Selection

Object orientated design has been selected for this project because the number and varied nature of the information to enumerate and the methods of enumeration lends itself to an object orientated approach. Each method of enumeration can be modelled as an object that inherits from a parent enumeration class. Polymorphism is also useful in this case as each enumeration object should have an '`enumerate`' method which performs differently for each object in order to obtain the required information.

Additionally the reduced number of bugs and greater ease of code maintainability that come with an object orientated approach due to encapsulation are assets to this project.

Although this project could be completed using any of the alternative design methodologies discussed above the disadvantages are more significant than those of object orientation.

The unpredictability of functional decomposition and data flow methodologies make them unsuitable for this project as it is intended for open source public release. This means that a large number of contributors may be involved at a later date so the application design should be apparent without direct communication between contributors.

While the data structure design method makes the application design apparent, it is reported to not scale well. This would be a handicap for this application as it may grow as requirements are added for additional information to be enumerated.

3.2.2 Unified Modelling Language

The industry standard Unified Modelling Language (UML) is a design methodology that can be used for object orientated programs, providing tools for design and implementation of software based systems as well as modelling of business and similar processes (Object Managment Group, 2012). UML has the potential to aid affective communication of a complex design in a simplified form, helping to ensure that everyone has the same understanding of the system (Shalloway & Trott, 2004). Using this specification it is possible to model the application in a form that can easily be used to create a high level testing strategy (Samuel & Mall, 2009).

There are several diagrams that are defined in UML which are used in the different phases of the project.

Use cases are textual descriptions of success and failure scenarios of an actor using a system to meet a goal; created during the analysis phase of the project (Larman, 2004). In this context an actor is something with behaviour, for example a person, organisation or another computer system. And a scenario (also known as use case instance) is a specific sequence of actions and interactions between actors and the system. A simple UML “use case diagram provides a succinct visual context diagram for the system, illustrating the external actors and how they use the system” (Larman, 2004).

When considering object interactions sequence diagrams are created to illustrate input and output events from external actors to a system. Each sequence diagram shows the set of events within a use case, external actors, the system, and the system events that are generated (Larman, 2004). Systems are represented as a black box in sequence diagrams so that the emphasis is on the “events that cross the system boundary from actors to systems” (Larman, 2004).

During the design phase class diagrams are created to visualise classes, interfaces and their associations (Larman, 2004). Class diagrams are usually derived from the sequence diagram.

A UML state machine diagram illustrates the interesting events, states and transitions of an object along with the object's behaviour in reaction to an event (Larman, 2004). These diagrams can be arbitrarily simple or complex levels of detail depending on the author's needs. In this setting an event is defined as a significant or noteworthy occurrence, a state is the condition of an object between events, and a transition is a relationship between two states that indicates the object moves from one to the other when an event occurs.

In the deployment phase, deployment diagrams are used to show the assignment of software artefacts, such as executable files, to computational nodes (something processing services) (Larman, 2004). This includes the deployment of software on the physical architecture and the communication between physical elements.

3.2.3 Design Patterns

UML design patterns are reusable constructs for software development; they describe a general solution to a problem that is common in many projects (Dong & Yang, 2003). There are three key reasons to use patterns (Shalloway & Trott, 2004). Patterns allow reuse of solutions, this reduces duplication of work and helps to avoid common pitfalls because it has the benefit of learning from the experience of others. They provide a common point of reference during the analysis and design of a project, establishing terminology and a common viewpoint of the problem. Patterns also give a higher level perspective on the problem, design and object orientation, allowing details to be ignored in the early phases.

For example the iterator pattern (Gamma, et al., 2012) provides a standard interface for traversing a collection of items. This may be used in 'reporter' section of the product to traverse through the information that has been enumerated so that it can be reported to the user. The command pattern (Gamma, et al., 2012) expresses a request, including the call and all required parameters of a command object, which may be executed immediately or held for later use. This pattern may therefore be used in the 'enumerator' functions which must make use of Windows API calls.

3.3 Security Development Lifecycle

The Microsoft Security Development Lifecycle (SDL) "is a set of processes and tools designed to reduce the number and severity of vulnerabilities in software products" (Lipner, 2010). Creating a product that is free from serious security flaws is a goal which all developers should strive to achieve and should be considered by potential users of the application.

The SDL consists of seven phases and 17 practices which span the lifetime of the project. However not all of these are relevant to an individual project so will not be covered.

Establishing Security and Privacy Requirements. Defining and integrating these requirements early minimises disruption and makes it easier to identify key milestones and deliverables. This involves defining minimum security and privacy criteria for the application and deploying a security vulnerability/work item tracking system (Microsoft, 2013).

In this project, this practice will be followed by establishing security requirements by considering potential attackers of the system and their intents and creating requirements that will prevent the attacker exploiting the system. Privacy requirements are not applicable to this project as it is a single user system that is designed to make available as much information as possible from the Windows Active Directory. A vulnerability/work item tracking system will be implemented by creating a private project on bitbucket.org with issue tracking. Using bitbucket.org has the added advantage of backing up the project source code off site.

Perform Security and Privacy Risk Assessments. Portions of a project that require threat modelling and security design review before release are identified by examining software based on costs (e.g. monetary, reputation, time) and regulatory requirements.

The security and privacy considered in this project is that of the single user and their system that runs the application, rather than the user's that the application retrieves information about because this application is intended to be used during a security assessment of a Windows domain and therefore must not limit the information retrieved. The primary focus of the risk assessments will be to determine the level of risk associated with interacting with external systems such as DNS and the Windows domain.

Practice 5: Establish Design Requirements. Validating all design specifications against a functional specification involves accurate and complete design specifications, including minimal cryptographic design requirements and a specification review.

In this project these design requirements will primarily pertain to how input into the system by the user and external systems is handled to prevent security and privacy issues.

Practice 6: Perform Attack Surface Analysis/Reduction. "A system's attack surface is the subset of the system's resources that an attacker can use to attack the system" (Manadhata & Wing, 2011). Limiting the opportunities for attackers to exploit potential weaknesses in the application requires an analysis of the attack surface and applying the principle of least privilege and layered defences. The principle of least privilege is that every user and program of a system should only have the minimum amount of permissions required to complete their task, the military security rule of 'need-to-know' is an example of this principle (Saltzer & Schroeder, 1975).

In this project the avenues of attack, such as file and network input/output, will be identified and reduced to the minimum required for the application to operate. Every feature will act under the principle of least privilege so that any flaw in the feature does not result in complete compromise of the system's security. Layered defences will be utilised, such as: validating and sanitising input and following coding standards to increase ease of bug detection.

Practice 7: Use Threat Modelling. Applying a structured approach to threat scenarios enables vulnerabilities to be identified, risks associated with those threats to be determined and appropriate mitigations established.

The threats to the system, i.e. an attacker's potential goals, will be identified by analysing the program's use cases against the STRIDE threat types (Torr, 2005):

- **Spoofing** – Impersonating something or someone else when a communication passes over a trust boundary.
- **Tampering** – Modifying data in transit or while it resides on a machine.
- **Repudiation** – performing an action that cannot be traced back to them
- **Information Disclosure** – information exposed to unauthorised parties.
- **Denial of Service** – Preventing the system's legitimate operation
- **Elevation of Privilege** – performing actions without proper authorisation.

3.4 Language Choice

Several factors must be considered when choosing one of the many available programming languages for this project. These factors include programmer productivity, maintainability, efficiency, portability, tool support, and software and hardware interfaces (Spinellis, 2006).

The languages listed below will be compared using the factors outlined above in order to identify an appropriate development language for this project. These are not the only languages that could be used to create a product that meets the needs of this project, however these are the three that the author has most experience with and time constraints preclude the use of a language that must first be learned from by the author.

- C
- C++
- Python

Programming languages can be split into two broad categories: compiled languages (such as C and C++) and scripting languages (such as Python).

The C programming by Kernighan and Ritchie (1988) is a general purpose procedural programming language, by design it has the greatest degree of independence possible from specific hardware platforms. It is a low level language i.e. dealing with the same sort of objects as computers do: character, numbers and addresses. It does not have operations to manipulate composite objects such as strings, does not have any built in heap or garbage collection, or support multiprogramming or parallel operations. As C is independent of machine architecture, it is relatively easy to write portable programs, programs that can be run on a variety of hardware without change. C is a small language that does not have any high-level mechanisms, such as input and output, built in; instead these mechanisms must be explicitly called from the extensive standard library.

C++ created by Stroustrup (1997) is based on C so retains many of its predecessor's positive aspects including speed and platform independence. However C++ implements additional features that improves on the C programming language including object orientation (OO) through the 'class' type. C++ is in widespread use in part because it allows fast reliable code to be written in areas where requirements change significantly over time, proving itself to be maintainable and providing ease of extension and testing.

The core Python language and libraries are platform neutral, allowing programs to be run on any of the major platforms Lutz (2006). Scripting languages are also orders of magnitude easier and quicker to use than their compiled counterparts, the resulting systems are easier to change and reuse over time, and tasks such as network scripting and multitasking are a lot less cumbersome. Python is a modular, object orientated language which allows code reuse. Its clearly readable syntax and coherent design leads to readable code which can be more easily maintained. The speed of development is also increased as interpreter handles details that must be explicitly dealt with in lower level languages such as C and C++.

The Python programming language is object orientated (Lutz, 2013), supporting advanced notions like polymorphism, operator overloading and multiple inheritance. However it is remarkably easy to apply object orientated programming in Python due to its simple syntax and dynamic typing.

Each of these languages can utilise the Windows API, however the level of documentation for each language is vastly different. C++ is well supported by Microsoft as each API function is documented with a C++ syntax snippet. C is stated as supported by Microsoft, however there is no C code available in the documentation. Python on the other hand accesses the Windows API using the `pywin32` library, documentation for this library is available and references the Microsoft documentation where appropriate.

Prechelt (2000) conducted an empirical comparison of multiple programming languages including C, C++ and Python in which each language was used to implement the same functionality by several programmers. It was found that the development time (i.e. the time it takes to create a working product) for a Python program was half or less than that of C and C++, and the resulting code base (i.e. number of lines of code), was also half the size. C and C++ were revealed to have a significant advantage at run time, during initialisation of the program they were five to 10 times faster than other languages and during the main phase they were twice as fast. The memory consumption for the program in C and C++ was also determined to be half that of the alternatives.

After considering each of the prospective languages Python was selected for this project for the following key reasons:

- The lower development time (i.e. the time it takes to create a working product) will allow the project to be completed by the deadline.
- The smaller code base (i.e. number of lines of code) and readable syntax increases ease of maintainability.

- The proposed application is not expected to be time critical, so the performance benefits of C and C++ do not outweigh the increased development time.
- Python has the capability to integrate with C compatible languages. Therefore if an area of the application is found to require the efficiency that C provides, this section can be written in C and integrated into the larger Python application.
- Out of the three, Python is the language that the author is most fluent in.

4 Analysis Models and Design Specifications

4.1 Use Cases

The requirements dictate being able to perform manual and automated assessment of the domain and hosts. To fulfil these requirements use cases were produced which segregated these tasks. Architecturally, the highest level of the application handles user input and selects the appropriate middle level use cases based on the user's input. The middle level use cases designated as 'Automated Domain Enumeration', 'Automated Host Enumeration', and 'Manual Enumeration', each of which uses lower level use cases. The low level use cases carry out specific actions, for example the 'Enumerate Domain Groups' will enumerate a list of groups that exist in the domain.

There were several ways in which the desired functionality could have been split into hierarchal use cases. One alternative was to bundle all automated use cases together and all manual use cases together, however the significant differences between domain and host data was predicted to be unwieldy during implementation. Likewise bundling manual and automated domain use cases and manual and automated local use cases was considered to be unnecessarily difficult to implement due to the differences between automated and manual enumeration methods.

For each specific requirement use cases were created to describe their behaviour in detail. For example 'UC3' (see Appendix C) was produced from essential requirement 6) which dictates the retrieval of a list of groups from the active directory. It thoroughly documents the intended feature including conditions that must exist for the use case to run, the possible conditions after the use case has run, the normal flow of the use case along with any exceptions that may occur and how they should be handled, and any assumptions that the use case makes.

4.2 Sequence Diagrams

The content of each use case was used to create sequence diagrams which show the sequence of messages that are exchanged between elements of the application. For example the `GetDomainGroups` sequence diagram (see Figure 4.1) shows how the Actor sends the message `get_domain_groups()` to the `Enumerate Domain Groups` element which sends the message `win32net.NetGroupEnum()` to the `pywin32` element during normal operation.

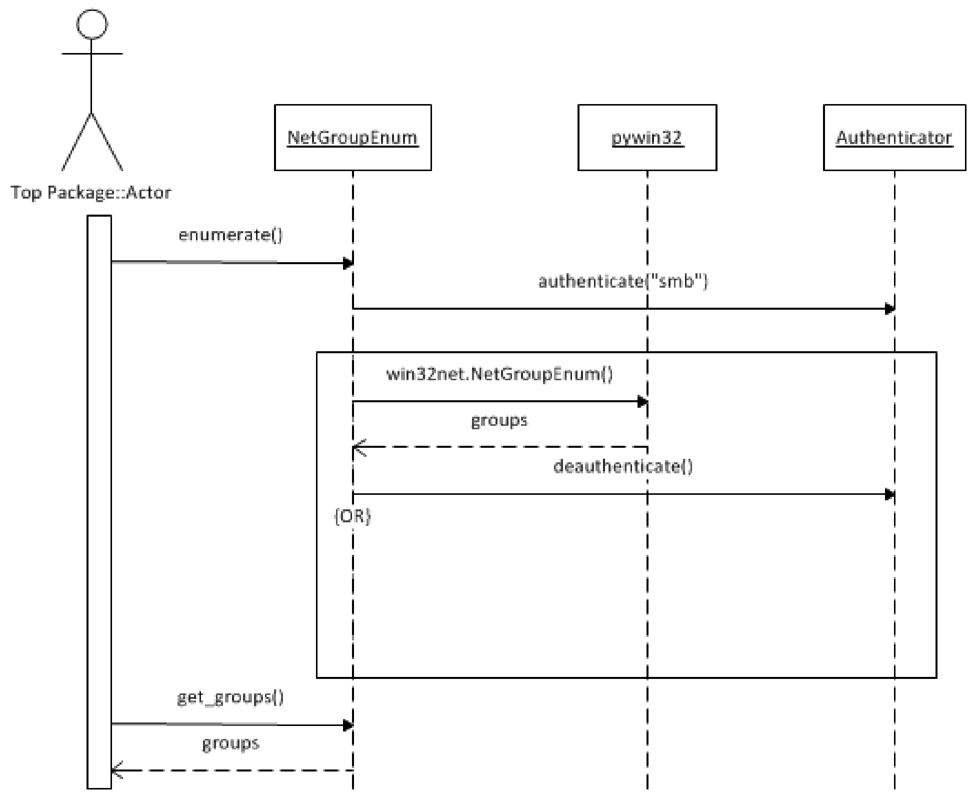


Figure 4.1: GetDomainGroups Sequence Diagram

4.3 Class Diagrams

The sequence diagrams and use cases were then used to derive the class diagram. The class diagram catalogues an object's inheritance, attributes and methods and indicates its relationship to other objects. For example the class diagram in Figure 4.2 shows that the `GroupEnumerator` class inherits the `BaseEnumerator` class and is inherited by the `NetGroupEnum` class. Grouping similar objects together and extracting their common properties into an abstracted parent class, which can be inherited, reduces code duplication and increases maintainability as each feature is only implemented once in the parent class and then inherited instead of being implemented separately in every child class.

The class diagrams also clearly indicate the levels at which attributes and methods are defined as well as where methods are overridden. For example the `GroupEnumerator` class defines the `self._host` attribute and methods to access it, its child, `NetGroupEnumerator`, inherits these and defines an `enumerate` method which overrides the `enumerate` method that it inherited from `BaseEnumerator` (via `GroupEnumerator`). Overriding methods implements polymorphism; each object implements the `enumerate` method in its own specific way but presents a common interface to the rest of the application, therefore a function can call the enumerate method on several different objects in the same way, but the actions that each object takes is different. Polymorphism makes it simple for a controller class to call all objects that do a specific type of enumeration, because it need only iterate through a list of the objects and call the `enumerate` method of each.

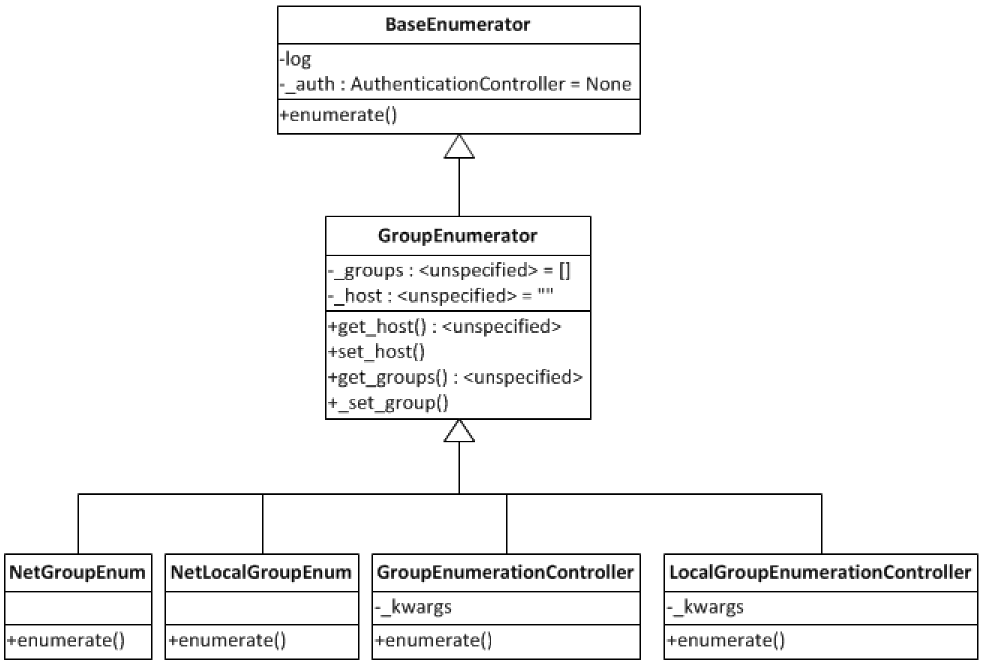

BaseEnumerator
-log
-_auth : AuthenticationController = None
+enumerate()
GroupEnumerator
-_groups : <unspecified> = []
-_host : <unspecified> = ""
+get_host() : <unspecified>
+set_host()
+get_groups() : <unspecified>
+_set_group()
NetGroupEnum
+enumerate()
NetLocalGroupEnum
+enumerate()
GroupEnumerationController
-_kwargs
+enumerate()
LocalGroupEnumerationController
-_kwargs
+enumerate()

Figure 4.2: GroupEnumerator Class Diagram

4.4 Output Format

The requirement specification specifies that the program should output in a parseable but readable form. There were several options available including creating a textual format that could be parsed by code developed specifically for it. However this was not considered to be the best available option as it would essentially require the development of a new standard specifically for this product which is obviously wasted effort when well known standards already exist to fill this need.

There are two widely used textual data formats that are human and machine readable and therefore suitable for this project: Extensible Markup Language (XML) and YAML Ain't Markup Language (YAML). XML is a markup language, meaning that it contains instructions for the software displaying the text but these are not displayed to the end user. It focuses on documents, but can be used to represent arbitrary data structures and can be transformed using Extensible Stylesheet Transformations (XSLT) to present the data in formats that are even easier for the user to read, for example as a web page using Extensible HyperText Markup Language (XHTML).

On the other hand YAML is a data serialisation language that was designed to map to common data types increasing the ease with which data can be ported between programming languages. It also uses the minimum amount of structural characters to display the information making it more readable for the user.

Consideration was also given to the libraries required to utilise these formats. The standard Python library contains several modules for XML, however YAML requires an additional module to be installed and configured.

XML was selected as the most appropriate output format for this project, the primary reason being that it is a document markup language and the intended use in this application is to create a report document. The advantage of having standard library modules to manipulate this format was also cause to choose Python as it reduces the deployment requirements for the application. Finally the ease with which XML can be transformed into a web page contributed to the selection because although XML and YAML are both readable by a human in their raw form they still require the user to be familiar with the format, whereas a web page can be used to display the data in a way that is immediately understandable.

The XML style sheet can either be a separate file or embedded within the XML file. While embedding the style sheet has the advantage of creating a single portable output file it increases the size of the file significantly which could become an issue if a large number of output files are saved. Referencing an external style sheet also has the advantage of being easily replaceable if the user wishes to modify how the data is displayed. Therefore the decision was taken to use an external style sheet that will be distributed with the project.

4.5 Structure of Data

Within the application data is stored in several different structures dependent upon the information that needs to be stored and the level within the program. The two highest level structures are `Domain` and `Host`, which reflect the level of information enumerated, either domain or local. These structures are designed as objects to encompass the entirety of the information that could be enumerated in a manner that is conceptually similar to how the information is stored within the Windows Active Directory and Security Account Manager (SAM).

The attributes of the top level structures store the enumerated information as an appropriate data type, with methods to get and set the value. This abstraction allows the internal format to be altered without affecting the rest of the application. The type of the attribute is dictated by the relationship between the information and the top level structure, for example there are one or more domain controllers per domain therefore the `self._domain_controllers` attribute is a list which can store multiple other objects, but there is only one domain name per domain so the `self._domain_name` attribute is a string.

Within each of these attributes the data type selected is dependent on the information it stores, for example `self.domain_controllers` list is populated with dictionaries because a single domain controller can be known by three distinct names (ip address, fully qualified domain name, and NetBIOS name). The policies attribute `self._policies` is also a dictionary because although there are several policies per domain, each has a discrete name and value and logically they should be grouped together as they work together to influence the security posture of the domain.

The attributes discussed so far are relatively simple and therefore the built in types of the Python language have been sufficient to handle them. However two of the categories of information (groups and users) are more complex and call for a correspondingly more intricate structure to store them. The information for each group is stored in its own instance of a `Group` object, the name and comment for the group are stored as string attributes, and the members of the group are stored in a list attribute as strings. Similarly a user's account information is stored in an instance of a `User` object with the majority of attributes as strings, but the `self._groups` attribute is a list of strings. Both of these objects implement methods to interact with the data, primarily adding to it or retrieving it all.

Wherever viable Python's built in types have been used because these are more efficient than types a developer can implement. It is technically possible to implement the entire data structure using built in types, nominally by nesting several dictionaries; however this would quickly become difficult to manage and would not allow methods to be implemented to validate data before it is assigned or abstract the internals of the data structure from the rest of the application.

The lower levels of the application do not deal with this high level structure at all, instead the information they require to perform their action is passed to them as arguments and they return the information in as a built in type. This simplifies the lower level objects as they do not need to be able to handle the complex data structure.

The higher levels of the application access the data structure using the appropriate methods to provide prerequisite information to the lower level objects and populate it with the information that is returned.

4.6 Authentication Design

The authentication design required careful consideration in order to fit sensibly into the object orientated design because, following the object orientated approach, each object must be able to perform all actions necessary to accomplish its objective.

Practically every enumeration object requires authentication to perform its task successfully, a simplistic solution would be for each enumeration object to create an instance of an authentication object and call its `authenticate` and `deauthenticate` methods as appropriate for its needs. This would function correctly and be suitable if there was only one enumeration object per type of authentication (for example SMB), however this application has multiple enumeration objects which use the same type of authentication so the simplistic solution described above would result in a large number of sessions being created and destroyed which is not very efficient.

A better solution requires authentication to be handled at a level above the enumeration objects so that a session can be created before the objects require it, and destroyed once all have finished with it.

Several objects are involved in the authentication process, the lowest of these are the authenticator objects that implement that authentication and deauthentication as dictated by the type of authentication. For example the object `SMBAuthenticator` has the methods `authenticate` and `deauthenticate` which can be called to authenticate to a specified host using a given set of credentials or deauthenticate respectively (see Figure 4.3).

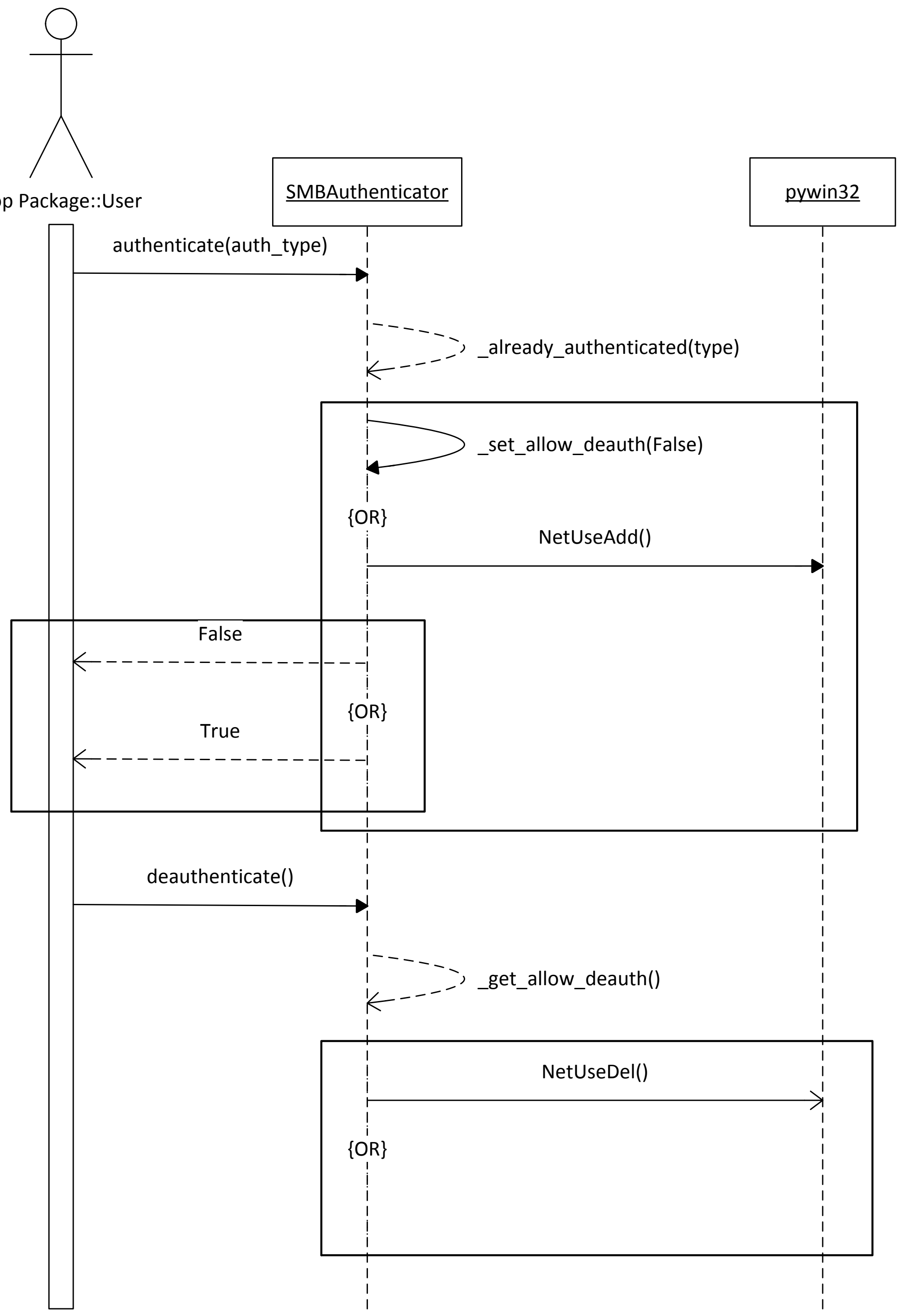


Figure 4.3: SMBAuthenticator Sequence Diagram

The `AuthenticationController` defines a `create_sessions` method, as the name suggests this method creates sessions using the specified details by invoking the authenticate method of the lower level authenticator objects (see Figure 4.4). It also defines an `authenticate` method which checks if a session has been successfully established, and a `deauthenticate` method which does nothing.

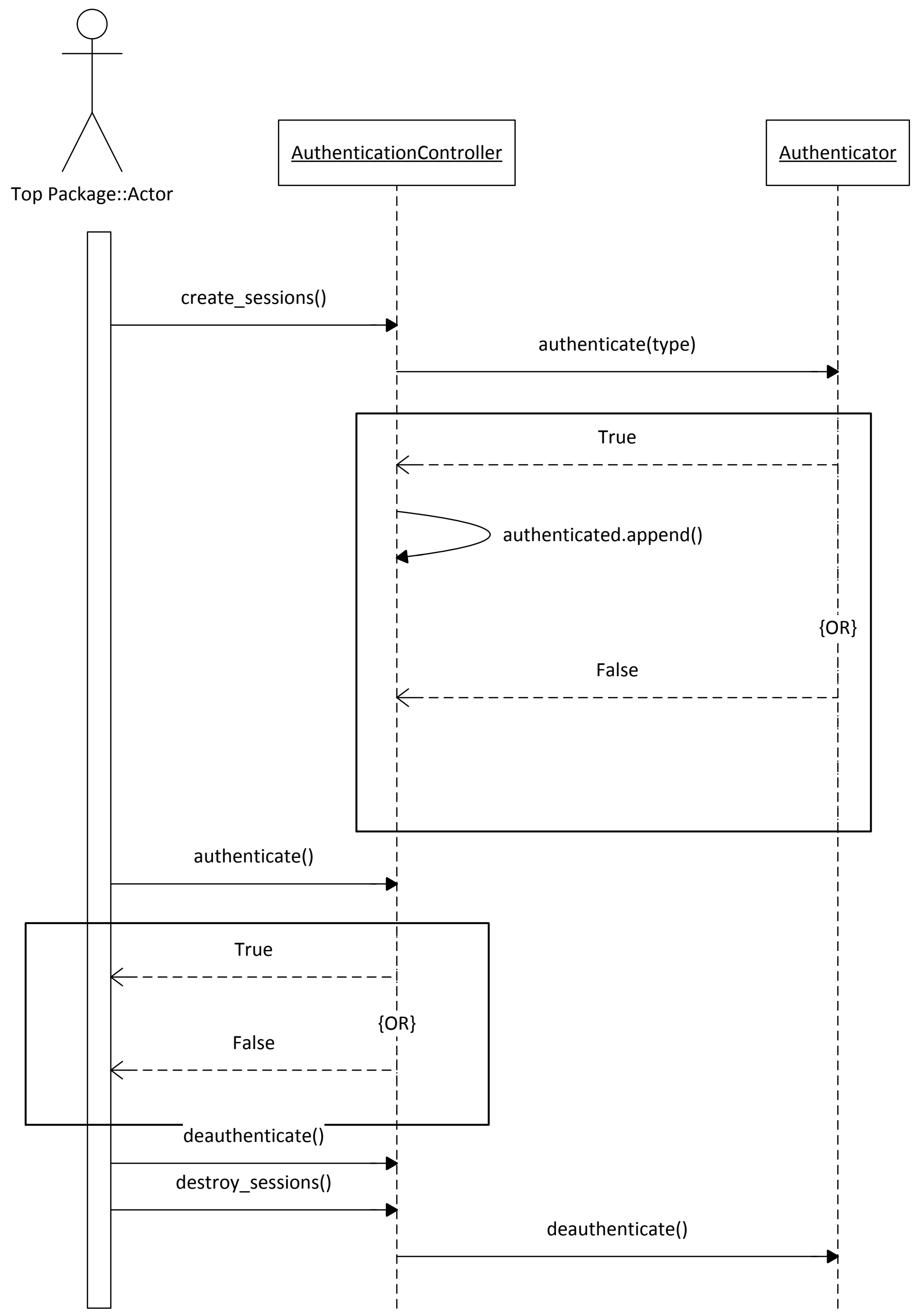


Figure 4.4: AuthenticationController Sequence Diagram

The higher level objects, such as `DomainAutomation`, instantiate the `AuthenticationController` object and call the `create_sessions` method to establish the sessions, then pass this instance to the lower level enumeration objects. Each of the low level enumeration objects calls the `authenticate` method of the instance, if `True` is returned the object performs its enumeration and then calls the `deauthenticate` method. When all of the lower level enumeration objects have completed the higher level object will call the `destroy_sessions` method of the `AuthenticationController` to terminate the sessions.

Since the `authenticate` method of the `AuthenticationController` only checks if a session is already established, and the `deauthenticate` method does nothing, these methods have no real effect on the authentication with the remote system and merely serve the logic of the lower level enumeration objects. In this usage the low level enumeration object's invocation of the `deauthenticate` method could be removed with no impact on the operation of the application.

However it is also possible to pass the lower level enumeration object an instance of a lower level authenticator, in which case the invocation of the `authenticate` and `deauthenticate` methods would establish and terminate the session with the remote host. This use of abstraction between the authentication and enumeration objects increases their flexibility, potentially allowing them to be reused in other applications without any change to the underlying code.

4.7 Design Patterns

The mediator design pattern defines an object that encapsulates how a set of objects interact (Agerbo & Cornils, 1998). This pattern promotes loose coupling between objects by preventing them from referring to each other explicitly.

Within this application this design pattern has been applied in automated use cases. For example the `DomainAutomation` object acts as the 'Mediator' for the enumeration objects 'Colleagues', because the majority of the 'Colleagues' require information from the others in order to carry out their task but require this information from the 'Mediator' rather than each other.

This reduces the dependency that each object has on each other leading to more maintainable code because all interactions are carried out via the 'Mediator' so any change to any one 'Colleague' will only affect one interaction between 'Colleague' and 'Mediator' rather than many interactions that would otherwise exist between 'Colleagues'.

The template method design pattern dictates that step by step algorithms are provided where each step can invoke an abstract method of a subclass or a base method (Wirfs-Brock & Johnson, 1990). Therefore the specific behaviour of an algorithm is provided by the subclass.

This design pattern is utilised within the reporter objects of the application. The `BaseXMLReporter` object defines the template method `generate` which invokes the abstract method `_to_rough_xml`. The `to_rough_xml` method is that of the child's class (i.e. `DomainXMLReporter` or `LocalXMLReporter`) and specifies how the XML to be generated.

5 Product Code

5.1 Standards

Coding conventions such as these are widely accepted to improve code reliability, portability and maintainability, although there is little empirical evidence to support this (Boogerd & Moonem, 2008). Conventions are said to improve code readability therefore allowing engineers to understand the code quickly and thoroughly, since 80% of a software's lifetime cost is on maintenance and hardly any software is maintained by the original author for its entire lifetime, this is very important (Sun Microsystems Inc, 1999).

During this project several standards for the Python programming language were followed, these standards are documented in Python Enhancement Proposals (PEPs). PEP 8 'A Style Guide for Python Code' (van Rossum, et al., 2013) is once such standard. It specifies coding conventions for all Python code, including the standard library. It covers code layout including indentation, line length, blank lines, file encoding, imports, use of whitespace, comments, version strings, and naming conventions. By following this standard the program can be easily be understood and maintained by other developers.

Another standard that was followed is PEP 257 'Docstring Conventions' (Goodger & van Rossum, 2001). "A docstring is a string literal that occurs as the first statement in a module, function, class or method definition" (Goodger & van Rossum, 2001). These document aspects of the code to aid the understanding of other developers therefore increasing code quality and may be extracted by other utilities to form documentation for a program. PEP 257 documents the recommended content and format of both single and multiline docstrings without dictating any markup syntax that may be used.

5.2 Built

The vast majority of proposed features in the requirements specification have been successfully implemented. However, one proposed feature that could not be implemented was identifying domain trusts. This feature was intended to identify other domains on the network which may also need to be enumerated. However the Windows API function that enables this (`DsEnumerateDomainTrusts`) has not yet been implemented in the `pywin32` library. Since the resources of this project do not allow for the function to be implemented, this feature has been omitted from the current product.

Inheritance has been heavily used in the application design. For example the `NetGroupEnum` object is a child of the `GroupEnumerator` object which is itself a child of the `BaseEnumerator` object. This allows features that are common between several types of objects to be implemented once and then inherited, for example the `GroupEnumerator` object implements methods to set and retrieve the host attribute which are inherited by all of its children. If inheritance was not used each object would have to implement this shared functionality individually increasing the code base and making it more difficult to maintain.

Each object has its own specific initialisation that is relevant only to it so is not in the parent's initialisation function. However as the child's initialisation function overrides the parent's, the child must specifically call the parent's in order to inherit the shared attributes. This is accomplished using the `super` function at the start of object's initialisation to obtain a proxy object to the parent class and then calling its `__init__` function.

An interesting nuance to how this has been implemented means that the child object does not need to know the required parameters of any of its parents. The child function accepts any number of keyword arguments and passes them directly to the parent's initialisation. After the parent's initialisation has been executed the child can also use the keyword arguments for its own initialisation. This maintains the abstraction between the layers of inheritance as the child does not need to know anything about its parent. This is demonstrated in Figure 5.1 where the `GroupEnumerator` object uses `super` to call the `__init__` function of its parent, in this case `BaseEnumerator`.

Since the parent class does not know whether keyword arguments will be supplied during initialisation or not, error handling must be employed to prevent a non-existent argument derailing the application. This is accomplished using the `try` and `except` statements as shown in Figure 5.1, if the keyword argument host is supplied it is assigned to the `self._host` attribute, however if it not supplied then a `KeyError` will be raised and caught by the except block which assigns an empty string literal to `self._host`. This implementation allows future changes to the parent or child class parameters to be realised without altering the other classes.

```
class GroupEnumerator(BaseEnumerator):
    """Group enumerator class."""
    def __init__(self, **kwargs):
        """Initialise group enumerator class. Extends
BaseEnumerator.__init__()."""
        super(GroupEnumerator, self).__init__(**kwargs)
        self._groups = []
        try:
            self._host = kwargs['host']
        except:
            self._host = ""
...
```

Figure 5.1: Inheriting Initialisation

During the design phase the decision was made to segregate authentication from enumeration functions. This was in keeping with the object orientated paradigm and helped to ensure that different sections of the program could not interfere with each other because they were encapsulated. However this design decision had a large impact when implementing the SNMP functions because in SNMP the authentication and enumeration functions are bundled together, i.e. every message must contain the authentication community string.

In order to maintain the designed sequence of ensuring authentication before attempting enumeration, SNMP authentication was checked using the supplied community string and the System OID which is guaranteed to be present on an

SNMP device. If this was successful then the application acted as though a session had been created. The SNMP enumeration objects check if authentication was successful and if so retrieve the community string form the authenticator object and use it to enumerate the desired information. This is not a perfect solution as the validity of the community string may change between the time of the check and action, resulting in either superfluous SNMP messages which will be denied access or information not being enumerated, however it is effective and complies with the designs.

Three standard library modules are used to implement the reporter objects. The first is `ElementTree`, this is a widely recommended lightweight XML processor, providing objects to store hierarchical data in memory including methods to both parse and build XML documents. This module is used to generate the XML structure from the internal data structure.

The second module `saxutils`, provides the method `escape` which encodes the characters ‘<’, ‘>’ and ‘&’ as their HTML entities (‘&lt;’, ‘&gt;’ and ‘&amp;’ respectively). This is to prevent an attack, known as XML injection attacks, where untrusted user input is stored as XML without proper sanitisation. For example in this application, if an attacker can cause XML to be returned by one of the enumeration objects (perhaps by specifying a XML in a domain group comment) it will be inserted into the XML output file. An attacker could insert misleading information into the output or use XML external entities to cause denial of service or create TCP connections (possibly as part of a distributed denial of service attack against another system, or to steal credentials).

Using the `escape` method to encode appropriate characters mitigates this vulnerability when data is inserted into the text area of an XML tag as this application does. However it is not sufficient if the data is inserted into the attributes of the tag because an attacker could break out of the attribute using quote characters, which are not encoded by the `escape` method, and manipulate the document from within an existing tag.

This has been implemented by calling the `escape` method on data before it is used to create a sub element using the `SubElement` method of `ElementTree` as shown in Figure 5.2. Whenever a subelement is created `_subelement_with_text` is called and passed the data, this method creates the XML tag then calls the `_escape_text` method on the data before assigning it to the text attribute of the tag.

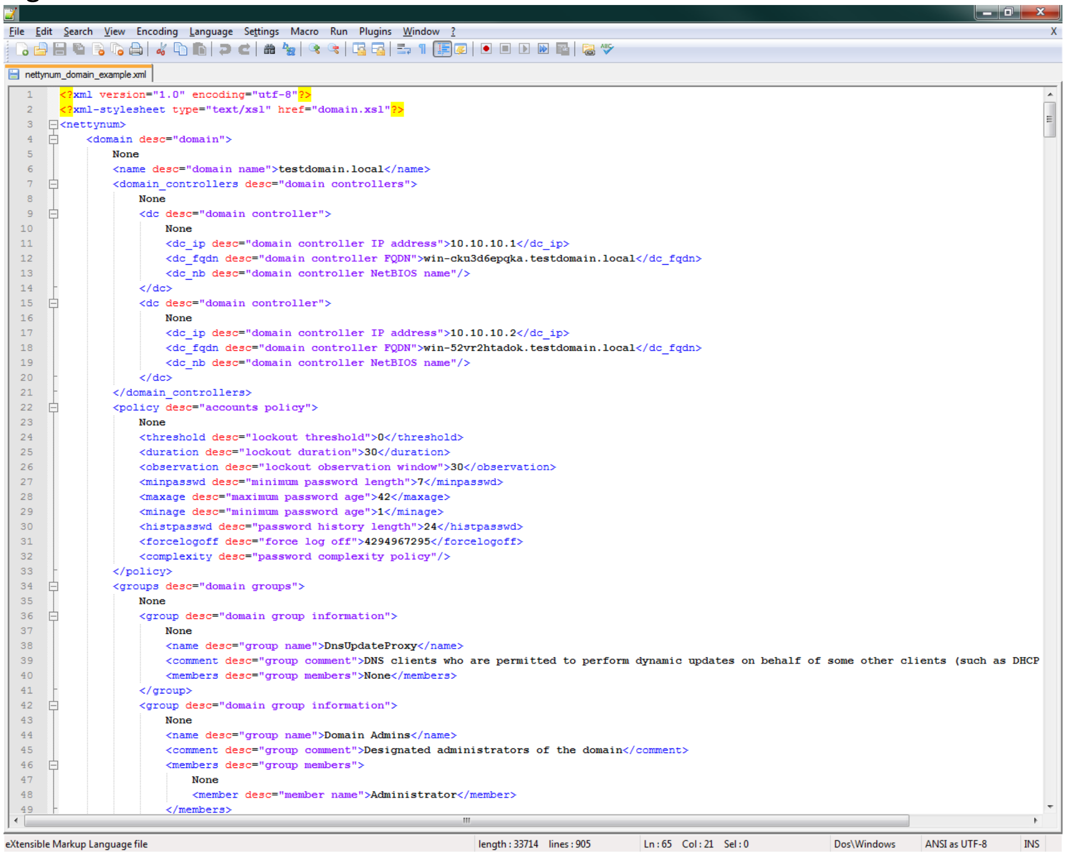
```
class BaseXMLReporter(BaseReporter):
    ...
    def _escape_text(self, text):
        """Convert text to a string and escape it to prevent XML 
injection.
        WARNING: This function is not sufficient for data in 
attribute values."""
        return escape(str(text))

    ...

    def _subelement_with_text(self, parent, tag, text, attrib={}, 
**kwargs):
        """Create an xml subelement."""
        elem = ET.SubElement(parent, tag, attrib, **kwargs)
        elem.text = self._escape_text(text)
        return elem
```

Figure 5.2: Escaping Data

The `ElementTree` module has a significant limitation, while it creates well-formed XML, it does not include new lines or indentation, making it difficult for a person to read. Since output readability is a requirement of this project this deficiency has been addressed using another XML module, `minidom`. This module is minimal implementation of the Document Object Model interface. Only three methods from `minidom` are used `parseString` to parse the XML string output by the `ElementTree` module, `createProcessingInstruction` which is used to add the reference to the XSLT file for formatting, and `toprettyxml` to provide the XML structure as a string containing new lines and indentation making it easier to read.

Two XSLT documents have been created to format the XML output file as HTML in a browser, providing many more formatting options and features which can be used to make the information more readable than in plain XMl. JavaScript has been used to create collapsible sections of data, allowing the user to more easily find the information they are looking for.

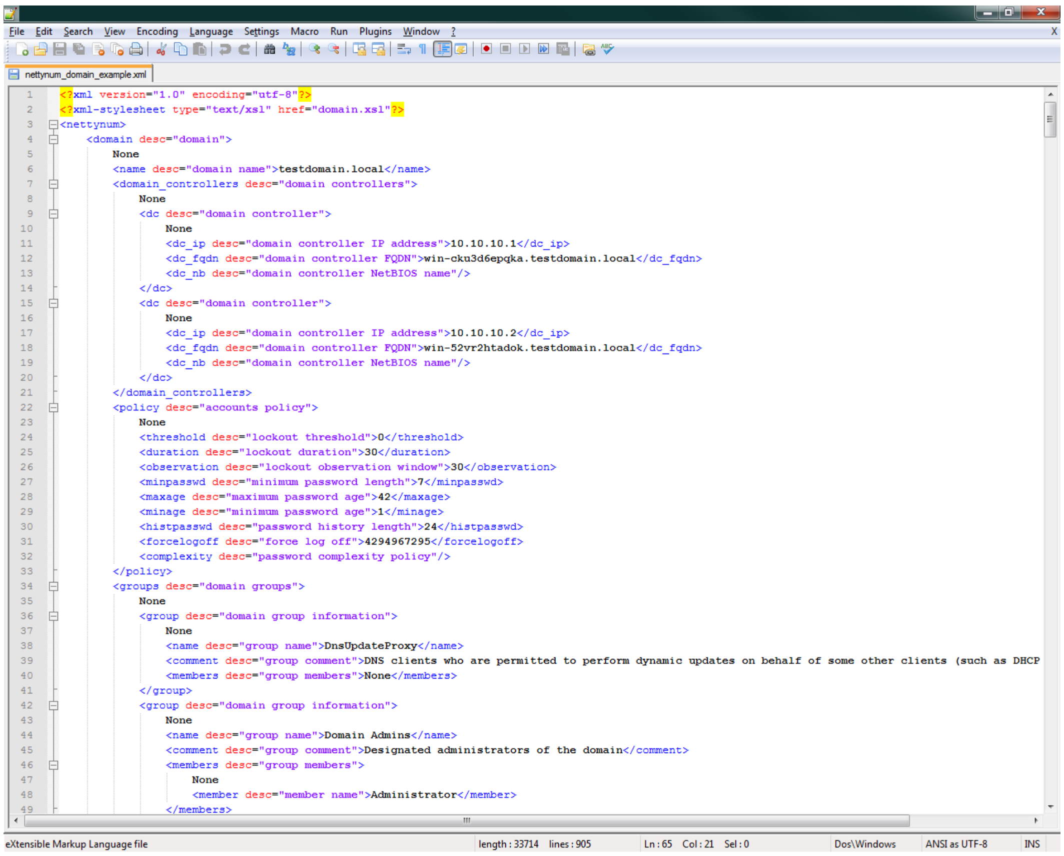
Figure 5.3



```xml
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="domain.xsl"?>
<nettynum>
    <domain desc="domain">
        None
        <name desc="domain name">testdomain.local</name>
        <domain_controllers desc="domain controllers">
            None
            <dc desc="domain controller">
                None
                <dc_ip desc="domain controller IP address">10.10.10.1</dc_ip>
                <dc_fqdn desc="domain controller FQDN">win-cku3d6epqka.testdomain.local</dc_fqdn>
                <dc_nb desc="domain controller NetBIOS name"/>
            </dc>
            <dc desc="domain controller">
                None
                <dc_ip desc="domain controller IP address">10.10.10.2</dc_ip>
                <dc_fqdn desc="domain controller FQDN">win-52vr2htadok.testdomain.local</dc_fqdn>
                <dc_nb desc="domain controller NetBIOS name"/>
            </dc>
        </domain_controllers>
        <policy desc="accounts policy">
            None
            <threshold desc="lockout threshold">0</threshold>
            <duration desc="lockout duration">30</duration>
            <observation desc="lockout observation window">30</observation>
            <minpasswd desc="minimum password length">7</minpasswd>
            <maxage desc="maximum password age">42</maxage>
            <minage desc="minimum password age">1</minage>
            <histpasswd desc="password history length">24</histpasswd>
            <forcelogoff desc="force log off">4294967295</forcelogoff>
            <complexity desc="password complexity policy"/>
        </policy>
        <groups desc="domain groups">
            None
            <group desc="domain group information">
                None
                <name desc="group name">DnsUpdateProxy</name>
                <comment desc="group comment">DNS clients who are permitted to perform dynamic updates on behalf of some other clients (such as DHCP
                <members desc="group members">None</members>
            </group>
            <group desc="domain group information">
                None
                <name desc="group name">Domain Admins</name>
                <comment desc="group comment">Designated administrators of the domain</comment>
                <members desc="group members">
                    None
                    <member desc="member name">Administrator</member>
                </members>
```

eXtensible Markup Language file | length : 33714 lines : 905 | Ln : 65 Col : 21 Sel : 0 | Dos\Windows | ANSI as UTF-8 | INS

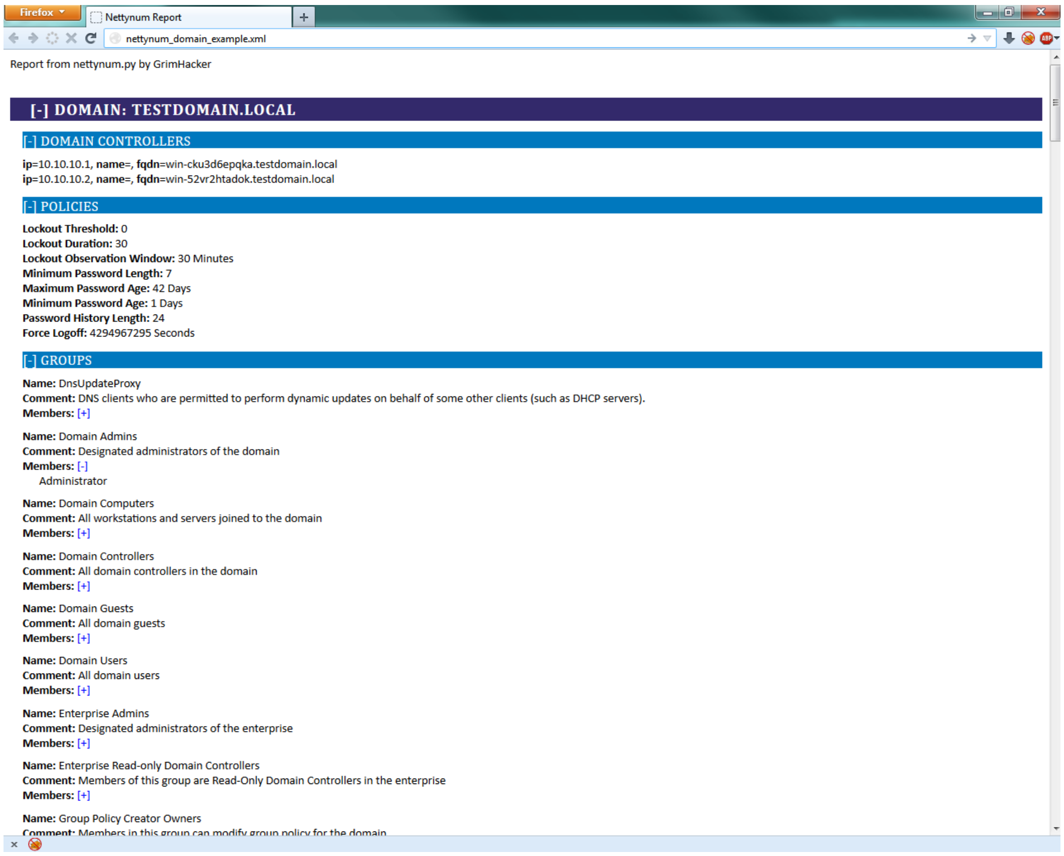
Figure 5.3 shows an example domain enumeration output file opened in a text editor and Figure 5.4 shows the same file open in a browser with the style sheet applied.


File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
nettynum_domain_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="domain.xsl"?>
<nettynum>
    <domain desc="domain">
        None
        <name desc="domain name">testdomain.local</name>
        <domain_controllers desc="domain controllers">
            None
            <dc desc="domain controller">
                None
                <dc_ip desc="domain controller IP address">10.10.10.1</dc_ip>
                <dc_fqdn desc="domain controller FQDN">win-cku3d6epqka.testdomain.local</dc_fqdn>
                <dc_nb desc="domain controller NetBIOS name"/>
            </dc>
            <dc desc="domain controller">
                None
                <dc_ip desc="domain controller IP address">10.10.10.2</dc_ip>
                <dc_fqdn desc="domain controller FQDN">win-52vr2htadok.testdomain.local</dc_fqdn>
                <dc_nb desc="domain controller NetBIOS name"/>
            </dc>
        </domain_controllers>
        <policy desc="accounts policy">
            None
            <threshold desc="lockout threshold">0</threshold>
            <duration desc="lockout duration">30</duration>
            <observation desc="lockout observation window">30</observation>
            <minpasswd desc="minimum password length">7</minpasswd>
            <maxage desc="maximum password age">42</maxage>
            <minage desc="minimum password age">1</minage>
            <histpasswd desc="password history length">24</histpasswd>
            <forcelogoff desc="force log off">4294967295</forcelogoff>
            <complexity desc="password complexity policy"/>
        </policy>
        <groups desc="domain groups">
            None
            <group desc="domain group information">
                None
                <name desc="group name">DnsUpdateProxy</name>
                <comment desc="group comment">DNS clients who are permitted to perform dynamic updates on behalf of some other clients (such as DHCP
                <members desc="group members">None</members>
            </group>
            <group desc="domain group information">
                None
                <name desc="group name">Domain Admins</name>
                <comment desc="group comment">Designated administrators of the domain</comment>
                <members desc="group members">
                    None
                    <member desc="member name">Administrator</member>
                </members>
```

```
eXtensible Markup Language file    length : 33714    lines : 905    Ln : 65    Col : 21    Sel : 0    Dos\Windows    ANSI as UTF-8    INS
```

Figure 5.3: Output File in Text Editor



Report from nettynum.py by GrimHacker

[-] DOMAIN: TESTDOMAIN.LOCAL

[-] DOMAIN CONTROLLERS

ip=10.10.10.1, **name**=, **fqdn**=win-cku3d6epqka.testdomain.local
ip=10.10.10.2, **name**=, **fqdn**=win-52vr2htadok.testdomain.local

[-] POLICIES

Lockout Threshold: 0
Lockout Duration: 30
Lockout Observation Window: 30 Minutes
Minimum Password Length: 7
Maximum Password Age: 42 Days
Minimum Password Age: 1 Days
Password History Length: 24
Force Logoff: 4294967295 Seconds

[-] GROUPS

Name: DnsUpdateProxy
Comment: DNS clients who are permitted to perform dynamic updates on behalf of some other clients (such as DHCP servers).
Members: [+]

Name: Domain Admins
Comment: Designated administrators of the domain
Members: [-]
Administrator

Name: Domain Computers
Comment: All workstations and servers joined to the domain
Members: [+]

Name: Domain Controllers
Comment: All domain controllers in the domain
Members: [+]

Name: Domain Guests
Comment: All domain guests
Members: [+]

Name: Domain Users
Comment: All domain users
Members: [+]

Name: Enterprise Admins
Comment: Designated administrators of the enterprise
Members: [+]

Name: Enterprise Read-only Domain Controllers
Comment: Members of this group are Read-Only Domain Controllers in the enterprise
Members: [+]

Name: Group Policy Creator Owners
Comment: Members in this group can modify group policy for the domain

Figure 5.4: Output File in Browser

5.3 Problems

During the implementation phase it was discovered that not all Windows API functions have been implemented within the `pywin32` library. This caused a problem, because the designs called for some functions to be used that have not yet been implemented. Notably `DsBindWithCred` was planned to be used for SMB authentication as this function establishes a sessions using a specified set of credentials to a domain controller. A choice needed to be made between implementing the function in Python and using the older, but available, `NetUseAdd` function. The time and human resources available for this project dictated that implementing the `DsBindWithCred` function in Python was not a viable option, therefore `NetUseAdd` was used in its place. The modular, abstracted design of this product meant that this did impact the other areas of the program, and allows the `DsBindWithCred` function to be substituted in during a later revision of the code, when it becomes available.

An interesting bug was encountered during development that caused every instance of the `Group` object to have an identical list of members despite the members only being assigned to one instance. It took some time to identify the cause of this error by stepping through the program but the cause was eventually discovered to be in the class declaration of the `Group` object. As can be seen in Figure 5.5 the `members` parameter has a default value of an empty list which is then assigned to `self._members`, however because this is declared in the method definition, it is a class variable and as such is shared between all instances of this object.

```
class Group(object):
    """Object that holds group data."""
    def __init__(self, name="", comment="", members=[]):
        ...
        self._members = members
...
```

Figure 5.5: Erroneous Group Declaration

To overcome this problem the declaration of the list within the method declaration was replaced with `None` and instead defined within the code block of the method to make it an instance variable. As can be seen in Figure 5.6 a Boolean `or` operation is used to assign `self._members` to the value of members or a new instance of an empty list, the effect of this is that if an argument is supplied to the `members` parameter then it will be used, otherwise an empty list will be created.

```
class Group(object):
    """Object that holds group data."""
    def __init__(self, name="", comment="", members=None):
        ...
        self._members = members or []
...
```

Figure 5.6: Corrected Group Declaration

6 Testing

6.1 Methodology

The software was tested at several levels (utilising static and dynamic analysis), described by Luo (2001), to determine its quality. The lowest level of testing is the unit test that tests the smallest pieces of software possible. When two or more units are combined together, integration testing is used. System testing is used to confirm end-to-end quality of the entire system based on the requirements specification. Finally acceptance testing is carried out when the system is provided to customers or users and gives confidence that the system is working as required.

Static analysis which focuses on determining software quality without actually executing it usually using code inspection (Luo, 2001). Static Analysis was used continuously while writing the code to reduce the number of bugs introduced due to coding errors. This involved reading the code, checking object names and verifying the logical flow matched the designs.

Dynamic analysis requires execution of the application and was selected as an appropriate testing style for this project because the development language does not require compilation. It therefore takes very little time to run tests, modify code accordingly and re-run the tests.

The software was tested at several levels, described by Luo (2001), to determine its quality. The lowest level of testing is the unit test which tests the smallest pieces of software possible. When two or more units are combined together, integration testing is used. System testing is used to confirm end to end quality of the entire system based on the requirements specification. Finally acceptance testing is carried out when the system is provided to customers or users and gives confidence that the system is working as required.

6.2 Test Bed

In order to test the application a controlled environment was required that simulated a Windows domain network. Two possibilities presented themselves, either a physical or virtual network could be setup and configured. The equipment required to implement a physical network made this option cost prohibitive during this project. Therefore a virtual network consisting of two Windows servers (2008 R2 and 2012) were created with Active Directory installed to create a Windows Domain. In addition to the active directory service, each server also had the DNS service installed as Active Directory relies heavily on DNS to locate objects, and the WINS service installed to serve NetBIOS names. Further details on the configuration of the test environment can be found in Appendix D. These operating systems were chosen for the test bed because they are the two latest releases in the Microsoft Windows Server line and are therefore recommended for all current and near future Windows domains, and they are available to the author under an educational license.

6.3 Tests

Unit tests were conducted using automated testing by creating appropriate test cases with the standard library Python module `pyunit`. For each of the lowest level units in the designs (i.e. authenticators, reporters, and some enumerators), test were created to determine their behaviour based on differing input. The first test for each unit verifies that the unit operates as expected with expected input, and the rest check its reaction to invalid input.

It was not possible to conduct unit testing of all the enumeration objects because in order to function they require authentication to an external host, which requires the integration of the authentication objects. For these modules integration testing was carried out using `pyunit` test cases which create an authentication object during their setup process and pass it to the enumeration object as required.

Further integration testing was conducted manually using the application from the command line to determine if the components work together correctly and all necessary command line arguments are present.

Test cases created from the requirements specification were completed manually to test the application at the system level.

Another system level test that was devised discovered the precision and recall of the application. Recall is the proportion of relevant information, and is used to measure how well the system retrieves all relevant information (Maarek, et al., 1991). Precision is defined as the proportion of the retrieved material that is relevant and is therefore used to measure how well the system retrieves only the relevant information (Maarek, et al., 1991). Powers (2011) defines recall as the proportion of Real Positive (RP) cases that are correctly Predicted Positive (PP) and precision as the proportion of Predicted Positive (PP) cases that are correctly Real Positives (RP); where the system's Predicted Positive results can be categories as either True Positives (TP) or False Positives (FP). See equations (1) and (2) respectively.

$$\text{Recall} = TP / RP \tag{1}$$

$$\text{Precision} = TP / PP \tag{2}$$

The Real Positive cases for each of the types of information that the application is capable of enumerating (e.g. domain names, group names) within the test bed are known and shown in Appendix D.

Once the application passed the Alpha testing phase described above it was released to a small group of Beta testers. These Beta testers are computer security professionals who were asked to use the software in place of their regular Windows domain enumeration tools and feedback issues they encountered along with their general opinion of the tool.

6.4 Results

The testing results are shown in Appendix E, however this section will highlight some of the key findings of the testing.

The unit and integration tests demonstrated that every unit of the application work together and behave as expected when provided with valid and invalid input with the exception of the `FromMaster` object. The `FromMaster` object fails to enumerate domain names from the master browser because it has not been implemented. This decision was taken when it became apparent that identifying the Master Browser without knowing the domain name required every IP address in the network range to be queried sequentially until the Master Browser was found. If implemented this would have had a significant negative impact on the performance of the tool because of the amount of time required to query such a volume of hosts and the timeouts that would be encountered during the process.

The system level tests of the program showed that the application meets all of the essential requirements and all but one of the desirable requirements specified at the start of the project. The desirable requirement "Identify domain trusts" was not met because implementing this functionality required the use of a Windows API function that is not yet available in the `pywin32` library. The documentation was followed on a clean build to ensure it accurately described the requirements of the application and its usage.

As this system retrieves information the measurements of precision and recall were calculated to determine reliability. The results show that the application's precision in every area is '1'. In all areas the recall of the application is also calculated to be '1' with the exception of user account information enumeration which is calculated to be '0.931'. These values indicate that on the whole the application will recall almost all of the relevant information and all of the information it does retrieve will be relevant.

There were several pieces of constructive feedback from the Beta testers. The first and most difficult to implement was from the Beta testers that used a Linux based operating system as their primary machine and would like to have the tool work natively rather than within a virtual Windows machine. This is a valid opinion, however operating on a Linux platform was defined within the excluded requirement for this project due to time constraints. Based on this feedback later versions of this tool should have Linux support as a requirement.

A couple of errors were also identified during the Beta test. First, an error that resulted in ungraceful exit of the program when enumerating shares was encountered. Investigation of this error found that the issue was caused by the share type returned during execution not being listed in the Microsoft documentation and was therefore not handled by the application. To address this error the share enumeration section of the program was altered: when looking up the numerical share type identifier to find the associated string a default value is used so that a `KeyError` will never be raised (see Figure 6.1) and the share type that was identified during testing was added to the list.

```
share_types.get(share_info['type'], "UNKNOWN_SHARE_TYPE")
```

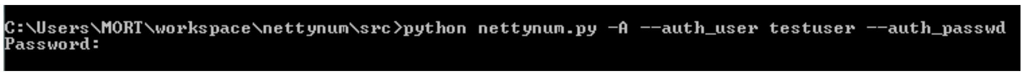
Figure 6.1 Share Type Default Value

Another error that was identified involved the targeting parameters. It was found to be possible to cause an error in the program by specifying the domain controller without the domain name when performing automated domain enumeration. This was found to be an oversight in the input validation code, while the specific enumeration function's parameters where checked to ensure prerequisite information was provided, the automated enumeration functions were not. The application was updated to contain validation code which prints an error and exists gracefully if the targeting parameters are not correct.

A potential security issue was also identified during the Beta testing. The password parameter accepts the password on the command line, this causes it to be stored in the command history of the host and it is visible to anyone looking at the user's screen. In order to mitigate this issue the input of the application was changed so that instead of supplying the password as an argument on the command line it is supplied by the user when the program runs without echoing it to the screen. This is accomplished using the `getpass` function of the `getpass` standard Python library for simplicity and portability. Figure 6.2 shows the code used to either get the password from the user if the command line arguments indicated that a password will be supplied or set it to the default value of an empty string. An example of the prompt displayed to the user as a result of this code is shown in Figure 6.3.

```
if args.auth_passwd:
        # Prompt for the password without echoing. Using default
prompt "Password: "
        auth_passwd = getpass.getpass()
    else:
        auth_passwd = "" # Default password
```

Figure 6.2 Password Handling with getpass()



```
C:\Users\MORT\workspace\nettynum\src>python nettynum.py -A --auth_user testuser --auth_passwd
Password:
```

Figure 6.3 Password Handling Command Line

Some Beta testers commented that while the documentation was accurate with regard to the command line parameters, some of the parameters could have clearer names most notably `domain` and `domain_name` which have very different functions but are easily confused. For ease of use these and similar parameters have been renamed, for example `domain` is now `auth_domain` and `domain_name` is `domain` making it clear which parameter is for the authentication credentials and which is not.

A few suggestions were also made regarding the output. First the account policies are stated in seconds, however this results in very large numbers that must be converted to larger units and it was therefore suggested that these calculations be done by the application. In order to deal with this request the low level function that retrieved information was altered to convert the values before storing them and the XSLT file was altered to state the unit of each.

Another output suggestion was that the information should be displayed as a table rather than in collapsible sections. While this was not implemented during the project it would be possible by creating a new style sheet to display the output to the particular user's taste without altering any of the code of the application itself or affecting other user; the new style sheet could also be applied to existing output files without rerunning the application. This flexibility is one of the advantages of keeping the XSLT file separate from the generated XML.

An issue with the collapsible areas was also identified during Beta testing. Although the sections both collapsed and expanded each action also resulted in moving to the top of the page. To fix this issue the XSLT file was altered so that the hyperlink used to invoke the collapse function referenced the JavaScript void function instead of the anchor of the page.

Other comments about the application were very positive with many stating that they would find this tool very useful during their work and the code and design was of a high standard and could be integrated into other penetration testing systems (see Appendix F). However there were also suggestions to increase the amount of comments and error handling within the code to increase readability and robustness.

6.5 Testing Conclusions

The tests conducted have been designed to identify as many faults as possible so that they can be removed before completion of the project.

Syntax errors introduced by typographical mistakes appear to have been eliminated because the program successfully runs through tests which cover every code path within the application without raising any syntax errors.

Logical errors that may have been included at the design phase have also been removed as is evidenced by the fact that the application behaves as expected in every test with no infinite loops or other logical defects.

Interface faults are a particular concern in this case as the system is heavily reliant on correct usage of APIs in order to enumerate information. The enumeration unit tests show that this type of fault is not present by demonstrating that each API in use is receiving the correct parameters and returns the desired information.

The acceptance testing performed by the Beta testers indicates that the tool is useful for its intended purpose and with the stated modifications can be easily used.

It is possible that some faults have persisted through the testing phase of this product because the testing environment used may not accurately represent a genuine Windows domain or the small number of domains used by the Beta testers. This is to be expected since many Windows domains consist of a multitude of groups, user accounts and servers which may present unique issues which cannot be realistically simulated using the resources allocated for this project and may not have occurred during Beta testing. When used in a large variety of genuine environments the range of data that the tool encounters may reveal issues with data handling or performance that were not identified during testing.

Performance testing is one category of testing that was not carried out, because the requirements specification did not stipulate any performance measures that the tool had to meet.

During the course of testing it was not possible to use fuzzing techniques to identify security vulnerabilities within the application because this would require creating a specialist program which could respond to the application’s queries with fuzz strings. The time and human resources of this project did not extend to creating the fuzzing tool, however it should be considered in future testing schemes.

7 Evaluation

7.1 Evaluation of the Product

Two factors were considered when evaluating the product, build quality and fitness for purpose.

In terms of build quality the testing that has been conducted indicates that the product is able to handle both valid and invalid input gracefully. At no point during testing did the product terminate unexpectedly, this is due to the use of exception handling at every interface which allows exceptions raised by other modules to be handled gracefully, performing 'clean up' actions and continuing with other functions. The application was also able to handle exceptions raised by external modules, for example the `pywin32` function `NetGroupEnum` raises an exception when access is denied, if unhandled this would cause the program to cease, however it is caught and changes the flow of the program as appropriate.

Another advantage of this tool's build is the use of inheritance which eases code maintenance by reducing code duplication as shared functionality is written once and then inherited. For example all of the group enumeration objects inherit a function to set members, therefore if at a later date the format of storing members changes this function can be changed once in the parent function but affect every child function.

The use of inheritance and abstraction also allows features to be added at a later date with minimal changes to the rest of the code base. For example a new method of enumerating domain groups would only require a new object that inherits the `GroupEnumerator` object to be written and a single line to be added to the `GroupEnumerationController`.

It is also possible to use new services to enumerate information if an appropriate authenticator is created. This is possible due to the abstraction that is built into the product design, as long as the new object presents the appropriate interface it can be included with very little effort. This allows the product to continue to be improved without large changes to the code base therefore improving maintainability.

Another key feature of the design, and one of the requirements, is the paresable yet readable output file. The output file uses XML as the output format, this format can be trivially parsed using a wide variety of programming languages, and therefore the output of this tool can be programmatically used by other scripts. The XSLT file that is distributed with the program is used to transform the XML file into HTML dynamically in the browser making it easier to read. The XSLT file is independent of the code within the application, and therefore it can be altered to display the information in a more pleasing manner with absolutely no change to the application. This is an advantage as users can tailor the display of the enumerated information to suit their specific needs, for example a user may decide that the only user account information they wish to view in the browser is the username and comment which can be accomplished by commenting out a section of the XSLT file.

The PEP8 style guide for Python code has been followed for this project with the exception of the maximum line length recommendation. This makes it much easier for another developer to understand the code as common conventions are used throughout. The line length recommendation has been ignored in this case because it was felt that restricting the line to 80 characters would result in practically every line of code spanning two or more lines which would reduce readability, this is stated within PEP8 as an acceptable reason to ignore a guideline.

Part of the aim of this project was to create a product to store information from each run which can be used to improve subsequent runs. When the user manually specifies a group name this information will be retained, if the user indicates to do so, and used during subsequent automated enumeration to gather the members of specific groups regardless of other filtering.

There are several positive and negative aspects that arise when considering the product’s fitness for purpose.

On the whole the product can be considered fit for purpose because the system level testing showed that it meets all of the essential criteria defined in the requirements specification along with the majority of the desirable criteria. It also handles both NetBIOS and DNS style domain and host names allowing it to be used in both new and legacy Windows domains.

The testing demonstrated that the tool will perform well in its role as an information retrieval tool. It had a high level of precision and recall indicating that it will retrieve the vast majority of relevant information and the information it does retrieve will likely be relevant.

However there are a number of limitations to this tool that became evident during the testing phase.

Firstly, although the application can handle both NetBIOS and DNS style names it currently cannot link the two together, so in mixed technology domains information is enumerated once for the NetBIOS name and once for the DNS name which results in duplicate information existing in the output file. While duplication of information is inconvenient it is preferable to missing information entirely and can often easily be identified by the user when reading the output.

Another drawback of the tool is that it does not provide a means for the user to specify the service to use for enumeration which may result in extraneous traffic crossing the network. This drawback stems from the abstraction that is built into the application, only the lower enumeration levels require any understanding of how the information is enumerated so the higher levels cannot allow or disallow specific services.

However the focus of this tool is enabling the information to be retrieved rather than allowing the user to specify an exact method of how it should be retrieved. Since the product makes a single authentication attempt per protocol and then enumerates information using the corresponding methods if it succeeded, a work-around for this issue could be for the user to supply invalid authentication credentials for the protocol they wish to disable, this would result in a single failed authentication attempt and information not being enumerated using that protocol.

Another potential issue with the functionality of this tool is that it does not provide a means for the user to specify a number of hosts to enumerate, although it does perform automated enumeration of a number of hosts that it finds. While performing enumeration on a list of user supplied hosts was not identified as a requirement at the start of the project and therefore cannot be considered a failing of the product, feedback from users may require this feature to be added in later versions.

Three objectives directly relating to the product were defined at the start of the project.

Objective 5 required design diagrams for the structure and behaviour of the application to be created. This objective has been met in full, design diagrams were created and followed to create application that uses object orientation and abstraction. These are included in Appendix C.

Objective 6 stated that the designs should be implemented in the chosen development language utilising a source code management solution and following the MS SDL. This objective has been met, the designs have been implemented in the Python programming language and Git was used to manage the source code. The MS SDL has been considered throughout the project process to help create an end product that has fewer security vulnerabilities than would otherwise be the case.

Objective 9 required user documentation to be generated for the application. This objective has also been fulfilled; the documentation takes the form of a ‘readme’ file that is to be distributed with the application. It contains a description of the application, its requirements, and its usage, with examples of the different command line parameters that can be used to perform enumeration. The Beta testers were able to make use of the application without any assistance from the author implying that the documentation is sufficient.

7.2 Evaluation of the Project Process

This project was the author’s first experience of a significant software development project and was therefore at times very challenging, requiring individual learning of new skills and concepts in a short time frame.

7.2.1 Achievement of Relevant Objectives

One of the objectives of this project was to create a list of requirements for the tool. This required careful consideration of the literature review that was conducted at the start of the project to identify key areas that a new tool in an existing area must cover in order to be useful in the field. These requirements were then expressed using the ‘IEEE Recommended Practice for Software Requirements Specification’, although this is not a standard it is widely used as it provides clear guidance on the content, and qualities of a software requirement specification.

Another objective of the project was to enhance the author's knowledge of the Windows API using online documentation. While this proved to be challenging due to the varied quality and detail of the documentation, it was successfully accomplished leaving the author with a wider understanding of the different Windows API functions, including their requirements for input and output data structures, and the varied levels of authentication required to use them.

Objective 8 states that an analysis of the testing results was to be undertaken to identify the functional and non-functional aspects of the application. This objective has been met; the different aspects of the application have been verified using the testing results. These results have shown that the product is suitable for its intended purpose as it meets its requirements and has a high level of precision and recall.

At the commencement of this project the author had a fair understanding of the Python programming language but knew that a project of this magnitude would require a higher level of skill. Therefore one of the objectives of the project was to enhance the developer's skills in the selected development language. Using online resources the developer was able to learn to utilise the more advanced features of the Python language and its conventions to create a code base that meets the product requirements while being maintainable by any Python developer.

From the outset the author's lack of software engineering background was identified and objectives put in place that identified key milestones in a software engineering project, namely design diagrams and testing. This required the author to learn appropriate design and testing methodologies and tools, particularly UML which was completely new to the author but used extensively to design the application.

7.2.2 Suitability of Tools and Techniques

The selected programming language, Python, proved to be a suitable choice for this project. The syntax made the code easy to read but offered the advanced notions of object orientated languages and as the interpreter handles details that must be explicitly dealt with in lower level languages it had a low development time. The downside of this language is that it has a higher execution time than the alternative compiled languages, however this was not raised as an issue by the Beta testers but they did note that there was a noticeable delay when authenticating with incorrect credentials. This delay is caused by the timeout implemented by the remote server and is identical regardless of the language the client is developed in.

One issue that was encountered when using the Python programming language was that not all Windows API functions were implemented in the `pywin32` library, this resulted in some intended features not being implemented. If this project were to be repeated this should be taken into consideration when selecting the development language as the full set of functions is available in both the C and C++ programming language, whereas additional development is required to utilise them in Python.

A number of tools were used during the course of this project with varying degrees of success. Microsoft Visio was used to create UML diagrams, this tool offers the basic features required to create simple UML diagrams, and integrates well with Microsoft Word allowing the diagrams to be explained in the report. However the drawback to this tool is that it does not offer the full range of features defined in the UML specification so is not suitable for extremely complicated or intricate diagrams that include complex models. An additional consideration was cost, the author already owned a copy of Microsoft Visio so no additional expenditure existed, but tools that fully implement the UML specification have a significant price or only offer limited functionality in the community version.

With these respects in mind, it is the author's opinion that Microsoft Visio was a suitable tool to use for the design diagrams in this project. However if the project was to be extended with additional features that increased the complexity, a tool that offers the additional features to handle this should be used.

The Microsoft Secure Development Lifecycle MSSDL was an asset during the project because it required the author to consider the security implications during each stage of development rather than as a single consideration at the end of the project. This increased the amount of time that was spent on securing the application and therefore the security of the finished product.

The Git source code management system was a useful tool during this project as it easily allowed off site backup of the source code along with revision history. This allowed the code to be rolled back to a known working state at any time during implementation of a feature if it was found to be problematic. Git was found to be a reliable and convenient method of managing source code for this project and the author fully intends to use this system for future software engineering projects.

7.2.3 Managing Project Work

This project has been a new experience for the author as it is the first time they have conducted a significant piece of software engineering. It has involved designing, implementing and testing a piece of software while maintaining documentation. A key factor of this project was time management, the final deadline for the completed product and report was fixed so the work had to be completed on time. In order to facilitate this work that needed to be carried out was planned, and this schedule was adhered to as much as possible. If at any point the work fell behind the schedule extra hours were completed in order to finish on time.

When completing a piece of work individually it is always tempting to neglect the documentation since no other developers exist to share it with during the project. However in this case the author maintained all required documentation for the project so that it could be referred to by the author at later stages and by other developers in the future.

In a team of developers there is usually a range of experience to draw from when implementing novel features, however in this project there was a sole developer with a limited set of experience. This meant that the programming was a constant learning experience where unusual requirements necessitated the author researching the appropriate solution either by searching documentation or by experimentation. This was particularly evident when using the `pywin32` interface to the Windows API since at times neither set of documentation is entirely clear on the parameters that are required, so trial and error were required to obtain the expected arguments for the functions.

7.2.4 Personal Performance and Problem Handling

During development there were a number of problems in the application. Along with the problems already discussed there were also typographical errors which caused the application to crash and logic flaws which caused subtle malfunctions in the program. Each of these issues was tracked from the symptoms to the cause by examining the program's operation in relation to the source code to identify the errors that led to the anomalous behaviour. When the cause was discovered a method of eradicating the error was devised and examined to ensure it would not introduce any problems in the rest of the application. The solution was then implemented and tested both to ensure that the original problem had been removed and no new problems had been introduced.

This project can be considered the author's largest piece of academic work to date and has necessitated the development of several skill areas. This has included conducting a literature review to critically analyse existing products and determine requirements. Design, implementation and testing methodologies and technologies were also learned to a significant level in order to complete this project and will be useful for future software engineering tasks. Use of documentation for this type of project has also been learnt including proposal and evaluation documentation. Significantly a major report for this project has been completed; this type of report is likely to be required for the author's career and will therefore be advantageous.

One problem encountered during this project was Windows API functions not being available in the `pywin32` library. This was not identified during the design phase and therefore resulted in some features of the application being impossible to implement within the time frame of the project. In the future the author will endeavour to identify situations such as this early on in the project and allocate time to devise a solution. For example the required Windows API features could have been made available by implementing them in C++ using the `ctypes` library and providing a Python interface if sufficient time had been allocated.

7.2.5 Project Plan Reflection

On the whole the project plan worked well as an approximate guide as to how much time needed to be spent on each task in order to achieve the objectives. However there were some aspects of the plan that underestimated the amount of time required. For example conducting tests of the application within the test bed was assigned only 3 hours, this failed to take into account the amount of time required to acquire, install and configure Windows Servers in a test environment.

These underestimates where counter balanced by tasks which were assigned resources too generously, notably generation of user documentation was assigned 4 hours, but it only took a fraction of this time. This combined with the contingency time that was built into the plan to account for unforeseen delays allowed the project to keep largely to the devised schedule and complete before the deadline.

7.2.6 Legal, Ethical and Professional Issues

When creating a computer security tool such as this the ethical issue of malicious use often arises. The argument being that creating a tool that an individual can use for nefarious purposes is wrong and therefore should not be done. However many common network administration tools can also be used maliciously by suitably motivated parties but these are required for the administrator to perform their role efficiently. The same is true for this tool, it may be able to be used for malicious purposes, but it can and should be used by administrators and computer security practitioners to identify flaws in their systems before an attacker does so that they can be properly mitigated.

The Computer Misuse Act (Great Britain, 2006) states that 1) "unauthorised access to computer material" and 2) "unauthorised access with intent to commit or facilitate commission of further offences" are illegal. Since this tool accesses information on the domain it could be used to break the Section 1 and as this is an enumeration tool it is by definition a precursor to further activities and could therefore be used to contravene Section 2. Therefore during the Alpha testing phase of the application it was not possible to test the tool in a genuine environment as it was not possible to get permission and conducting the tests without permission would have resulted in breaking the Act. In order to stay within the law the Alpha testing phase was conducted in a virtual domain specifically created for the purpose and owned by the author.

During Beta testing the application was used by professional computer security experts during genuine assessments, this means that it was used in a very realistic scenario but the results it retrieved are covered by the non-disclosure agreement between the tester and the client and can therefore not be shared with the author. The feedback for the Beta testing phase therefore does not contain any quantitative data and instead relies on the qualitative feedback of the testers.

7.3 Recommendations

There are a number of recommendations that can be made at the culmination of this project.

First, during the acceptance testing phase a number of suggestions were received regarding improvements to the application for example the most significant of which were Linux support and associating DNS and NetBIOS style names.

This tool, with some redevelopment, would also be suitable as a plugin of a larger framework used in computer security assessments. This is because the information enumerated could be used by other plugins for example a password auditing tool could use the usernames and lockout timings to refine its activities in order to prevent disruption.

The product could also be extended with new methods of enumeration including different protocols and information to enumerate. For example LDAP may be able to be used to gather information from Active Directory and is occasionally available over the internet. Extending this product to include LDAP enumeration may extend the use cases of this tool from internal security assessments to external.

This application may also be applied to the area of network monitoring with some alterations. The application retrieves the incorrect password count and the last successful logon time, with some additional code and regular execution this tool may be used to detect a brute force attacks against domain users.

Finally, while this tool is intended for use by ethical computer security professionals, it has been noted that the information that can be retrieved from the Windows domain is useful to attackers. Further work should therefore be carried out to ensure that products and procedures to identify Windows domain enumeration both exist and are highly effective.

7.4 Conclusions

During this project a new tool for enumeration of a Windows domain has been designed and created based on research of existing tools.

An analysis of the existing tools and their limitation has been undertaken which identified a need for a new domain enumeration tool to address the limitations of existing tools.

This analysis informed the requirements specification for the tool that was created.

An appropriate design methodology, software development lifecycle and programming language were selected in order to create an application that meets the requirements specification.

The designs for the application were created using UML, this included use cases, sequence diagrams and class diagrams. These designs were then implemented using appropriate design patterns and following suitable standards for the selected programming language.

A testing strategy was developed to provide assurance of the application, with particular regard to the functional aspects of the application and the objectives of the project. With the testing conducted the results were then analysed to determine whether the requirements of the product and objectives of the project were met.

Finally an evaluation of the product and project process was undertaken to identify the strengths and weaknesses of the product and the personal and professional growth of the author.

The culmination of this project is enhanced skills in research, design and implementation of a Windows domain enumeration tool in the Python language. All the project objectives have been met in full. The product has been tested and proven to be fit for purpose and of a high build quality emanating from active professionals in the computer security industry.

8 References

Agerbo, E. & Cornils, A., (1998) How to preserve the benefits of design patterns. *ACM SIGPLAN Notices,* 33(10), pp. 134-143.

Allen, R., Lowe-Norris, A. G. & Richards, J., (2006) *Active Directory.* 3rd ed. s.l.:O'Reilly.

Berghel, H. & Hoelzer, D., (2005) Pernicious ports. *Communications of the ACM - The semantic e-bussiness vision*, 12 December, 48(12), pp. 23-30.

Bergland, G. D., (1981) A Guided Tour of Program Design Methodologies. *Computer,* 14(10), pp. 13-37.

Birkholz, E. P., (2002) *How to Fix A Broken Window.* Las Vegas, Black Hat.

Boogerd, C. & Moonem, L., (2008) *Assessing the value of coding standards: An empirical study.* Beijing, China, IEEE International Converence on Software Maintenance, pp. 227-286.

Chacon, S., 2009. *Pro Git.* 1st ed. s.l.:Apress.

Chi-Liang Ni, D., (1997) Enumeration and traceability tools for UNIX™ and WINDOWS™ environments. *Journal of Systems and Software,* 39(1), pp. 15-25.

Cooper, R., (1997) '[NTSEC]' CPU 100% Update (fwd), *ISS NT Security Mailing List,* 28/01. Available at: http://diswww.mit.edu/menelaus/bt/3980 [Accessed: 15 October 2013]

Davis, A., (2013) *Revealing Embedded Fingerprints: Deriving Intelligence from USB Stack Interactions.* Las Vagas, Defcon.

Dong, J. & Yang, S., (2003) *Visualizing design patters with a UML proile.* Auckland, New Zealand, IEEE, pp. 123-125.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J., (2012) *Gang of Four Design Patterns Reference Sheet.* [Online] Available at: http://www.blackwasp.co.uk/GangOfFour.aspx [Accessed 19 November 2013].

Goodger, D. & van Rossum, G., (2001) *PEP 257 -- Docstring Conventions.* [Online] Available at: http://legacy.python.org/dev/peps/pep-0257/ [Accessed 23 February 2014].

Great Britain, (2006) *Computer Misue Act 1990: Elizabeth II. Chapter 18 as amended by the Police and Justices Act 2006: Elizabeth II. Capter 48..* s.l.:s.n.

ISECOM, (2010) *Open Source Security Testing Methodology Manual.* 3rd ed. s.l.:ISECOM.

Kernighan, B. W. & Ritchie, D. M., (1988) *The C Programming Language.* 2nd ed. USA: Prentice Hall.

Kessler, G. & Shepard, S., (1997) *RFC 2151 - A Primer On Internet and TCP/IP Tools and Utilities.* [Online] Available at: http://xml2rfc.tools.ietf.org/html/rfc2151 [Accessed 5 November 2013].

Kitchenham, B. & Carn, R., (1990) Research and Practice: Software Design Methods and Tools. In: *Psychology of Programming.* s.l.:European Association of Cognitive Ergonomics and Academic Press, pp. 271-284.

Larman, C., (2004) *Applying UML and Patterns: An Introduction to Object-Orientated Analysis and Design and Iterative Development.* 3rd ed. Westford, Massachusetts, US: Addison Wesley Professional.

Lauristen, J., (1999) *NetViewX.* [Online] Available at: http://www.ibt.ku.dk/jesper/netviewx/ [Accessed 29 October 2013].

Lipner, S., (2010) Security development lifecycle. *Datenschutz und Datensicherheit - DuD,* 34(3), pp. 135-137.

Luo, L., (2001) Software testing techniques. *Institute for Software Research International Carnegie Mellon University Pittsburgh,* Volume 19, pp. 1-19.

Lutz, M., (2006) *Programming Python.* 3rd ed. USA: O'Reilly.

Lutz, M., (2013) *Learning Python.* 5th ed. United States of America: O'Reilly Media Inc.

Maarek, Y. S., Berry, D. M. & Kaiser, G. E., (1991) An information retrieval approach for automatically constructing software libraries. *IEEE Transactions on Software Engineering,* 17(8), pp. 800-813.

Manadhata, P. K. & Wing, J. M., (2011) An attack surface metric. *IEEE Transactions on Software Engineering,* 37(3), pp. 371-386.

McClure, S., Scambray, J. & Kurtz, G., (2009) *Hacking Exposed Network Security Secrets & Solutions.* 6th ed. United States: Mc Graw Hill.

McClure, S., Scambray, J. & Kurtz, G., (2009) *Hacking Exposed: Network Security Secrets & Solutions.* 6th ed. s.l.:McGraw-Hill.

Melber, D., (2005) Auditing User Accounts. *Internal Auditing,* 20(6), pp. 41-44.

Melber, D., (2011) Using LDP to Enumerate Active Directory Information. *Internal Auditing,* 26(6), pp. 40-44.

Microsoft, (2003) *RPC Technical Reference: Remote Procedure Call (RPC)* [Online] Available at: http://technet.microsoft.com/en-us/library/cc759499.aspx [Accessed 20 November 2013].

Microsoft, (2012) *Windows Internet Name Service (WINS) Overview.* [Online] Available at: http://technet.microsoft.com/en-us/library/hh831671.aspx [Accessed 20 November 2013].

Microsoft, (2013) *Microsoft Security Development Lifecycle.* [Online] Available at: http://www.microsoft.com/security/sdl/default.aspx [Accessed 13 November 2013].

Object Managment Group, (2012) *OMG Unified Modeling Language TM (OMG UML) Version 2.5 FTF - Beta 1.* [Online] Available at: http://www.omg.org/spec/UML/2.5 [Accessed 14 October 2013].

O'Dea, M., (2003) *Hack Notes Windows Security Portable Reference.* 1st ed. Emeryville, California: McGraw-Hill/Osborne.

Powers, D. M., (2011) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness & correlation. *Joournal of Machine Learning Technologies,* 2(1), pp. 37-63.

Prechelt, L., (2000) An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl. *IEE Computer,* 33(10), pp. 23-29.

Saltzer, J. H. & Schroeder, M. D., (1975) The Protection of Information in Computer Systems. *Proceedings of the IEEE,* 63(9), pp. 1278-1308.

Samba Team, (2010) *rpcclient.* [Online] Available at: http://www.met.rdg.ac.uk/~it/cgi-bin/man.cgi?section=1&topic=rpcclient [Accessed 3 November 2013].

Samuel, P. & Mall, R., (2009) Slicing-Based Test Case Generation from UML Activity Diagrams. *ACM SIGSOFT Software Engineering Notes,* 34(6), pp. 1-14.

Scambray, J. & McClure, S., (2008) *Hacking Windows Exposed.* 3rd ed. United States: McGraw Hill.

Shalloway, A. & Trott, J. R., (2004) *Design patterns explained: a new perspective on object orientated design.* 2nd ed. s.l.:Perason Eductaion.

Singleton, T. W., (2008) What Every IT Auditor Should Know About Access Controls.. *Information System Control,* Volume 4.

Skoudis, E., (2013) *Plunder Windows Account Info via **Authenticated** SMB Sessions.* [Online] Available at: http://pen-testing.sans.org/blog/pen-testing/2013/07/24/plundering-windows-account-info-via-authenticated-smb-sessions [Accessed 29 October 2013].

Spinellis, D., (2006) Choosing a programming language. *Software, IEEE,* 23(4), p. 62.63.

Stroustrup, B., (1997) *The C++ Programming Language.* 3rd Edition ed. s.l.:Addison-Wesley Professional.

Stuttard, D. & Pinto, M., (2008) *The Web Application Hacker's Handbook.* 1st ed. Indianapolis, Indiana: Wiley.

Sun Microsystems Inc, (1999) *Java Code Conventions.* [Online] Available at: http://www.oracle.com/technetwork/java/codeconventions-150003.pdf [Accessed 23 February 2014].

Torr, P., (2005) Demystifying the threat modeling process. *IEEE Security & Privacy,* 3(5), pp. 66-70.

United States v. Microsoft - Review of the Final Judgements by the United Steates and New York Group (2007) 98-1232 (CKK)

van Rossum, G., Warsaw, B. & Coghlan, N., (2013) *PEP 8 -- Style Guide for Python Code.* [Online] Available at: http://legacy.python.org/dev/peps/pep-0008/ [Accessed 23 February 2014].

Veerasamy, N., (2009) 'High-level Methodology for Carrying out Combined Red and Blue Teams', *Second International Conference on Computer and Electrical Engineering.* Dubai, 28-30 December, doi: 10.1109/ICCEE.2009.177

William Collins and Sons Co. Ltd, (1987) *Collins English Dictioanry and Thesaurus.* 1st ed. Glasgow: Collins.

Wirfs-Brock, R. J. & Johnson, R. E., (1990) Surveying current research in object-orientated design. *Communications of the ACM,* 33(9), pp. 104-124.

9 Appendices

Appendix A: Terms Of Reference

Project Terms of Reference

CM0645: Individual Project

Nettynum

A Windows Domain Enumeration Tool

Software Engineering Project

Student Name:	Oliver Morton
Student Number:	W10005202
Course:	Ethical Hacking for Computer Security BSc (Hons)
Supervisor:	Dr Christopher Laing
Second Marker:	Dr Fouad Khelifi

Contents

1. Project Title

Nettynum – A Windows Domain Enumeration Tool

2. Background to Project

The Microsoft Windows operating system is extremely ubiquitous in both home and corporate environments to a point where it has been “determined Microsoft had a monopoly in the marked” (United States v. Microsoft - Review of the Final Judgements by the United Steates and New York Group, 2007).

Enumeration is an information gathering technique that probes identified services for known weaknesses, unlike other techniques, such as foot printing and scanning, it involves active connections to systems and directed queries (McClure, et al., 2009). It forms a key part of many security testing methodologies including the Open Source Security Testing Methodology Manual (OSSTMM) (Herzog, 2010) and the OWASP Testing Guide (OWASP Foundation, 2008).

A large amount of information can be enumerated from a Windows based network including: domain controllers, services running on hosts, logged on users, installed software, shared resources, user groups, policies, user rights, and user information (McClure, et al., 2009). This information may appear harmless, however it lead to a complete compromise of the network and should therefore be eliminated from the network (McClure, et al., 2009).

A number of services can be used to gather this information from a Windows domain:

As the Microsoft Active Directory (AD) namespace is based on the Domain Name Service (DNS), services are advertised through DNS SRV records. It is therefore possible to identify servers running specific services by searching for the appropriate SRV record, for example domain controllers can be found by searching for the Kerberos Authentication service (_kerberos._tcp) (Scambray & McClure, 2008).

The NetBIOS Name Service (NBNS), has now largely been replaced by DNS, but is still enabled by default on Windows domains. It can be queried to find workgroups, domains, domain controllers, network services, logged on users, MAC addresses and servers (McClure, et al., 2009).

Services and the ports they are running on along with internal and virtual IP addresses can be found using the Microsoft RPC Endpoint Mapper Service (MSRPC), this information is not as detailed as that from NBNS but it has been known for this service to be available in situations where NBNS is not (Scambray & McClure, 2008).

A NetBIOS session is a logical connection between any two processes on the network (Microsoft Corporation, 2010). A null session is an unauthenticated connection, also known as null session connection, anonymous logon and anonymous connection, over which it is possible to gather information about: network information, shares, users and groups and registry keys (Microsoft Corporation, 2006). It is considered the “biggest Achilles’ heel for Windows if not adequately protected” (McClure, et al., 2009).

The Simple Network Management Protocol (SNMP) agent service is not installed by default on Windows but it is commonly installed by organisations that manage their networks using this service (Scambray & McClure, 2008). This service uses READ and WRITE community strings, roughly equivalent to a password, for authentication, however there are several strings known to be in wide spread use; “public” is the industry standard READ string. By reading specific object identifiers (OID) from the LAN Manager Management Information Base (MIB) it is possible to gather information such as: running services, share names, share paths, comments on shares, usernames, and the primary domain name (Scambray & McClure, 2008).

Another technology that can be used to enumerate information from the Windows Active Directory (AD) is an implementation of the Lightweight Directory Access Protocol (LDAP). By connecting to the AD server it is possible to browse the contents of the directory including all existing users and groups (Scambray & McClure, 2008).

Tools which use these one or more of these methods to gather information include: nltest, nslookup, dumpsec, Netbios Auditing Tools (NAT), user2sid, sid2user, NBTEnum, enum4linux.pl, NetE, GetAct, nbtscan, mbenum, nbtspade, ldapenum, winfo, enumeration modules are also available in the Metasploit Framework.

During their time in industry the author found that several of these tools were required to be used in succession to retrieve the required information and each requires some level of direction from the user. Some flaws were also encountered while using the existing tools, for example only retrieving the first 100 groups from the Active Directory and requiring sessions to be established before running the tool.

The proposed application will be targeted at security professionals who need to discover what information can be enumerated from their own, or a client’s, system.

There are several languages that are appropriate for developing a tool which uses these methods of enumeration to retrieve information including: C, C++, Java, Ruby, Perl, and Python. Each has advantages and disadvantages which should be examined, particular factors for selecting a development language include: the targeted platform, the elasticity of the language, the time to production, the performance and the support and community (Reghunadh & Jain, 2011).

The Microsoft “Security Development Lifecycle (SDL) is a security assurance process that is focused on software development” (Microsoft Corporation, 2010). Implementing a number of security activities throughout the traditional software development life cycle instead of on an ad-hoc basis leads to greater security gains. There are 16 mandatory security activities that must be undertaken to comply with the Microsoft SDL process:

1. Training Requirements
2. Security Requirements
3. Quality Gates/Bug Bars
4. Security and Privacy Risk Assessment

5. Design Requirements
6. Attack Surface Reduction
7. Threat Modelling
8. Use Approved Tools
9. Deprecate Unsafe Functions
10. Static Analysis
11. Dynamic Program Analysis
12. Fuzz Testing
13. Threat Model and Attack Surface Review
14. Incident Response Plan
15. Final Security Review
16. Release/Archive

Unified Modelling Language provides tolls for analysis, design and implementation of software based systems as well as for modelling business and similar processes (Object Management Group, 2012). Using the tools in the UML specification it is possible to model the properties and behaviour of a system and therefore ensure that all requirements of a product have been accounted for in the plan.

Key challenges that must be overcome during this piece of work include: selection of appropriate information to store which will improve subsequent runs of the tool; and using the available technologies to effectively gather information which is required for other enumeration methods without user input.

3. Proposed Work

The author proposes to identify, through a literature review, the information that can be enumerated from a Windows domain. Possible development languages will also be considered along with the methods of gathering this information using common network services such as DNS, DHCP, SNMP and NetBIOS. Existing Windows enumeration products will be discussed in terms of the amount of information that they retrieve and the level of user input required before retrieving this information.

Following Microsoft's Security Development Lifecycle (MS SDL) (Microsoft Corporation, 2010) a product will then be designed using Unified Modelling Language (UML) (Object Management Group, 2012)diagrams, and implemented which uses the identified methods to gather the information from the Windows domain. The design should include a feedback and improvement mechanism that is persistent across multiple runs of the command line tool.

Primarily the tool will be designed to require no user input and enumerate as much information as possible when run, however, for instances where autonomous running is insufficient, the application will also allow the user to direct enumeration. It should we written in an as an object oriented program with the functionality abstracted to allow future expansion and increase maintainability. It should output in a human readable and machine parsable form to maximise the usefulness of the results.

A testing plan will be devised to assess the functional and non-functional aspects of the product, this will require the configuration of a virtual/lab Windows domain. The testing plan will then be conducted and the results analysed to evaluate the product's level of functionality.

Finally user documentation will be developed for the product which will describe how the product can be used and give examples of both input arguments and corresponding output.

4. Aims of Project

The aim of the project is to develop an enumeration tool that that discovers information from a Windows domain and stores information from each run which can be used to improve subsequent runs.

5. Objectives

In order to accomplish the aim of the project the following objectives have been devised.

1. To perform a literature review encompassing existing tools, useful information that can be enumerated, methods of enumeration, and persistent storage solutions. [Tasks 1-8]
2. Create a list of requirements for the tool based on the literature review. [Tasks 9-13]
3. Enhance knowledge of the Windows API – online documentation [Tasks 14-16]
4. Enhance skills in the selected development language. – online tutorials [Tasks 17-20]
5. Create design diagrams for structure and behaviour of the application. (object oriented, abstracted) [Tasks 21-25]
6. Implement the designs in the chosen development language utilising a source code management solution (Git) and following the MS SDL. [Tasks 26-31]
7. Develop and carry out a plan to test the application against the requirements specification. [Tasks 32-37]
8. Undertake an analysis of the results of the testing to identify functional and non-functional results. [Tasks 38-39]
9. Generate user documentation for the application. [Tasks 40-43]

6. Skills

The successful completion of this project relies on a number of skills and areas of knowledge. Some of these are familiar to the authors but others must be attained as part of the project.

Software programming and functional and non-functional testing skills have been obtained through “EN0273 – Programming in C”, “EN0572 – Operating Systems and Concurrency” and online tutorials in the selected development language.

Additional software development skills in the area of design using Unified Modelling Language (UML) (Object Management Group, 2012) and the Microsoft Secure Development Lifecycle (MS SDL) (Microsoft Corporation, 2010) will be gained during the project by reading the materials recommended by the organisations website.

Correct usage of a source code management solution will be an asset during the project. The author has gained some experience of using the source code management solution “Git” (Git, 2013) during their year in industry; these skills will be enhanced by studying the documentation available the projects website and “Pro Git” (Chacon, 2009).

Knowledge of the information that is useful that should be enumerated during a penetration test was also gained during the year in industry. This knowledge will be supplemented with information from Hacking Exposed 6 Network Security Secrets and Solutions and other relevant texts.

It is anticipated that a number of Windows domain controllers and servers will be required as a test bed to assess the function of the product. Therefore knowledge of installation and configuration of a Windows domain will be sought through online documentation at Microsoft Developers Network (MSDN) (Microsoft Corporation, 2013).

7. References

Appcelerator Inc, 2013. *PyDev.* [Online] Available at: http://pydev.org/ [Accessed 14 October 2013].

Chacon, S., 2009. *Pro Git.* 1st ed. s.l.:Apress.

Git, 2013. *Git.* [Online] Available at: http://git-scm.com/ [Accessed 14 October 2013].

Herzog, P., 2010. *The Opensource Security Testing Methodology Manual.* 3rd ed. Cardedeu, Spain: ISECOM.

McClure, S., Scambray, J. & Kurtz, G., 2009. *Hacking Exposed Network Security Secrets & Solutions.* 6th ed. United States: Mc Graw Hill.

Microsoft Corporation, 2006. *The effects of removing null sessions from the Microsfot Windows 2000 and Microsoft Windows NT environment.* [Online] Available at: http://support.microsoft.com/kb/890161 [Accessed 14 October 2013].

Microsoft Corporation, 2010. *NetBIOS Session (Windows).* [Online] Available at: http://msdn.microsoft.com/en-us/library/bb870911(v=vs.85).aspx [Accessed 14 October 2013].

Microsoft Corporation, 2010. *Simplified Implementation of the Microsoft SDL.* [Online] Available at: http://go.microsoft.com/?linkid=9708425 [Accessed 14 October 2013].

Microsoft Corporation, 2013. *Microsfot DreamSpark.* [Online] Available at: https://www.dreamspark.com/ [Accessed 14 October 2013].

Microsoft Corporation, 2013. *Microsoft Developer Network.* [Online] Available at: http://msdn.microsoft.com/en-uk [Accessed 14 October 2013].

Object Management Group, 2012. *OMG Unified Modeling Language TM (OMG UML) Version 2.5 FTF – Beta 1.* [Online] Available at: http://www.omg.org/spec/UML/2.5 [Accessed 14 October 2013].

Oracle, 2013. *Oracle VM VirtualBox.* [Online] Available at: https://www.virtualbox.org/ [Accessed 14 October 2013].

OWASP Foundation, 2008. *OWASP Testing Guide.* 3rd ed. s.l.:OWASP.

Python Software Foundation, 2013. *Python Programming Language - Official Website.* [Online] Available at: http://python.org/ [Accessed 14 October 2013].

Reghunadh, J. & Jain, N., 2011. *Selecting the optimal programming language.* [Online] Available at: http://www.ibm.com/developerworks/library/wa-optimal/ [Accessed 15 October 2013].

Scambray, J. & McClure, S., 2008. *Hacking Windows Exposed.* 3rd ed. United States: McGraw Hill.

The Eclipse Foundation, 2013. *EGit.* [Online] Available at: http://www.eclipse.org/egit/ [Accessed 14 October 2013].

The Eclipse Foundation, 2013. *The Eclipse Foundation Open Source Community Website.* [Online] Available at: http://www.eclipse.org/ [Accessed 14 October 2013].

United States v. Microsoft - Review of the Final Judgements by the United Steates and New York Group (2007) 98-1232 (CKK).

VMware Inc, 2013. *VMware Workstation: Run Multiple OS, Linuix, Windows 8 & More; - UK.* [Online] Available at: http://www.vmware.com/uk/products/workstation/ [Accessed 14 October 2013].

8. Resources – Statement of Hardware / Software Required

Fulfilment of the proposed piece of work requires some standard computing equipment along with some more specialised software.

The selected development language, Python 2.7.5 (Python Software Foundation, 2013), is freely available for download from the project's homepage. Any text editor can be used to develop with Python, however an Integrated Development Environments (IDEs) are designed to increase productivity incorporating an editor, debugger, and console. The author's preferred IDE is Eclipse (The Eclipse Foundation, 2013), which is freely available from the project homepage and offers extensions for Python (PyDev (Appcelerator Inc, 2013)) and source code management (EGit (The Eclipse Foundation, 2013)).

Testing the product requires a functional Windows domain environment; to simulate real world situations multiple Windows server versions (2003, 2008 and 2012) and configurations will be required. This is licensed software that will be should be provided by the university, however if this is not possible, these operating systems are available through Microsoft DreamSpark (Microsoft Corporation, 2013) (formally Microsoft Developer Network Academic Alliance [MSDNAA]) to which the author has access.

Virtual machine software will be an asset during the testing process as multiple installations and configurations can be deployed rapidly, it also reduces the amount of physical hardware required for the project. VMWare Workstation is a suitable solution which supports "teams" of virtual machines (VMware Inc, 2013). It is anticipated that this software will be provided by the university, however if this is not the case then Oracle Virtual Box is a free virtual machine solution that could be used or multiple physical computers could be used (Oracle, 2013).

One or more physical computers will be required dependant on the specification; either a single machine capable of running several Windows virtual machines concurrently along with the IDE, or multiple machines each capable or running of fulfilling a specific role. These machines should be provided by the university; however the author's personal computer and laptop may be used as a contingency.

9. Structure and Contents of the Project Report

The following chapter are planned for the report.

1. Introduction [Tasks 44]

   This chapter will cover the aims and objectives of the project and an overview of the product, its: purpose, scope, main features and characteristics. It will go on to discuss the problem context and reasons for undertaking the project and summarise the approach taken and tools used.

2. Literature Review [Task 1-8]

   The literature review will cover the subject area in terms of enumeration as a step in a security assessment, information which can be enumerated from the Windows domain and the technologies that can be used to enumerate this information. This chapter will also cover the current tools that exist regarding what information is retrieved and how much user input is required to gather this information.

3. Requirements Specification [Task 9-13]

   A discussion of the key aspects of the specification will be included focusing on the design decisions that came from the literature review.

4. Analysis Models and Design Specifications [Task 21-25]

   Models produces from the requirement specification will be explained and justified.

   Interesting and important design decisions relating to the structure of the application, the use of particular design patterns, and interface designs will be discussed.

5. Product Code [Task 26-31, 40-43]

   The implementation of the designs will be explained in detail with attention paid to enumeration functions and feedback and improvement has been implemented.

6. Testing [Task 32-37]

   The testing plans devised for the product will be detailed and the reasons for each test justified.

   The testing strategy will also include a qualitative approach using qualified penetration testers from Sec-1 Ltd.

   The results of these tests will be documented and analysed resulting in a clear understanding of the state of the product that has been produced.

7. Evaluation of the Product [Task 38-39]

   A critical evaluation of the product will be undertaken against the requirements specification. The results of the testing phase will be used along with a reflection on the design and implementation.

8. Evaluation of the Project Process [Task 44]

   The project as a whole will be evaluated including aspects that would be conducted in a different manner if the project were to be repeated.

9. Conclusions and Recommendations [Task 44]

   This section will define to what extent the project's aims and objectives were met. Recommendations for further work will be stated including areas of improvement and further features that may be added.

10. Appendices

    Terms of Reference

    Design Documents

Product source code (Digital copy)

User documentation

10. Marking Scheme

10.1. Project Type

The project type that has been selected for this project is “Software Engineering”.

10.2. Project Report

The planned chapters of the report below address the indicated sections of the marking scheme.

Marking Scheme	Chapter Titles
Introduction	Introduction
Analysis	Literature Review Requirements Specification
Synthesis	Analysis Models and Design Specifications Product Code Testing
Evaluation, Conclusions & Recommendations	Evaluation of the Product Evaluation of the Project Process Conclusions and Recommendations

10.3. Product

Several deliverables will make up the finished product at the end of the project. The most significant of these is the application developed to meet the project aim. Other deliverables include the requirements specification, design diagrams, testing plans, test results and user documentation.

Fitness for purpose and build quality should be assessed against the following criteria:

Fitness for Purpose 40%

- Meeting of Requirements as identified during the project
- Quality of Functionality

Build Quality 60%

- Requirements specification & analysis
- Design specification
- Code quality
- Test plans and results

11. Project Plan – Schedule of Activities

Objective	Details/Description of Activities	w/c 21/10/2013	w/c 28/10/2013	w/c 4/11/2013	w/c 11/11/2013	w/c 18/11/2013	w/c 25/11/2013	w/c 2/12/2013	w/c 9/12/2013	w/c 16/12/2013	w/c 23/12/2013	w/c 30/12/2013	w/c 6/1/2014	w/c 13/1/2014	w/c 27/1/2014	w/c 3/2/2014	w/c 10/2/2014	w/c 17/2/2014	w/c 24/2/2014	w/c 3/3/2014	w/c 10/3/2014	w/c 17/3/2014	w/c 24/3/2014	w/c 31/3/2014
1. To perform a literature review encompassing existing tools, useful information that can be enumerated, methods of enumeration, and persistent storage solutions.	1. Identify relevant information sources	2hrs																						
	2. Identify information to enumerate	2hrs																						
	3. Identify methods of enumeration	2hrs																						
	4. Identify existing tools	2hrs																						
	5. Identify potential development languages	2hrs																						
	6. Discuss subject sections	4hrs																						
	7. Discuss existing tools	4hrs																						
	8. Discuss development languages	4hrs																						
2. Create a list of requirements for the tool based on the literature review.	9. MS SDL Practice 2: Security Requirements		1hr																					
	10. MS SDL Practice 3: Quality Gates/Bug Bars		2hrs																					
	11. MS SDL Practice 4: Security and Privacy Risk Assessment		2hrs																					
	12. Functional requirements		2hrs																					
	13. Non-functional requirements		2hrs																					
3. Enhance knowledge of the Windows API – online documentation	14. Identify API calls for required information		2hrs																					
	15. Identify the perquisite information the API calls require to operate		1hr																					
	16. Identify the authentication required to use the API calls		1hr																					
4. Enhance skills in the selected development language. – online tutorials	17. Read appropriate sections of the development language documentation			2hrs																				
	18. Read sections of Learning Python 2^{nd} Edition			3hrs																				
	19. Read the relevant programming style documentation for the development language			1hr																				
	20. Identify modules to enumerate the required information.			3hrs																				
5. Create design diagrams for structure and behaviour of the application. (object oriented, abstracted)	21. MS SDL Practice 5: Design Requirements			3hrs																				
	22. MS SDL Practice 6: Attack surface reduction			3hrs																				
	23. MS SDL Practice 7: Threat modelling			4hrs																				
	24. Read the UML specification and tutorials			5hrs																				
	25. Create design diagrams based on requirements specification.				10hrs																			
6. Implement the designs in the chosen development language utilising a source code management solution (git) and following the MS SDL.	26. Create Git repository				30min																			
	27. Read the MS SDL				1hr																			
	28. Program the application based on the designs following the MS SDL				180hrs																			
	29. MS SDL Practice 8: Use Approved Tools				1hr																			
	30. MS SDL Practice 9: Deprecate Unsafe Functions				1hr																			
	31. MS SDL Practice 10: Static Analysis															5hrs								
7. Develop and carry out a plan to test the application against the requirements specification.	32. Devise tests to determine if functional requirements have been met. (MS SDL Practice 11: Dynamic Program Analysis)																2hrs							
	33. Devise tests to determine if non-functional requirements have been met. (MS SDL Practice 11: Dynamic Program Analysis)																2hrs							
	34. Devise tests to meet MS SDL Practice 12: Fuzz Testing																2hrs							
	35. MS SDL Practice 13: Threat Model and Attack Surface Review																2hrs							
	36. Create an appropriate test bed to carry out testing.																3hrs							
	37. Conduct tests of the application within the test bed.																3hrs							
8. Undertake an analysis of the results of the testing to identify functional and non-	38. Compare test results to requirements																1hr							
	39. Discuss the testing																10hrs							
9. Generate user documentation for the application.	40. Specify application dependencies																	1hr						
	41. Specify steps required to create deployment environment																	1hr						
	42. Specify intended usage scenario																	1hr						
	43. Provide example user input and program output																	1hr						
Report Writing					20hrs														60hrs					
Contingency																						■	■	■

Appendix B: Software Requirements Specification

Software Requirements Specification

Nettynum

A Windows Domain Enumeration Tool

Oliver Morton

2013/2014

Software Engineering Project

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe the requirements specifications of Nettynum, a Windows domain enumeration tool. It explains the functional features, and design constraints.

1.2 Scope

Nettynum is intended to function in a Microsoft Windows Active Directory domain. The main objective is to enumerate information that can be used during a security assessment of an internal corporate network.

1.2.1 Essential

The following requirements must be met by the product.

13) Run on a machine that is not a member of a domain
14) Run on a Microsoft Windows machine
15) Identify domain names on a local area network
16) Identify domain controllers
17) Retrieve a list of users from the active directory
18) Retrieve a list of groups from the active directory
19) Retrieve a list of group members from the active directory
20) Retrieve user account information from the active directory
21) Retrieve the domain accounts policy (Lockout: duration, threshold, observation window. Password: length, character set)
22) Perform automated enumeration
23) Output in a parseable and human readable format
24) Provide a help / usage guide

1.2.2 Desirable

The following requirements maybe met by the product.

10) Perform directed enumeration
11) Identify services running on hosts
12) Identify domain trusts
13) Retrieve a list of users from a local host
14) Retrieve a list of groups from a local host
15) Retrieve a list of group members a local host
16) Retrieve information user account information from a local host
17) Retrieve the local accounts policy (Lockout: duration, threshold, observation window. Password: length, character set)
18) Retrieve a list of shares from a local host

1.2.3 Excluded

The product will not have the following requirments.

5) Conduct password guessing / brute force attacks
6) Include an interactive prompt
7) Include a graphical user interface (GUI)
8) Run on a linux based platform

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Enumeration

Within computer security the process of enumeration is an information gathering technique that uses active connections to a system and directed queries to obtain information from previously identified services (Scambray & McClure, 2008).

1.3.2 Domain

Allen, et al., (2006) provides a definition of the Windows domain. The "domain" concept was introduced in Windows NT, providing a way to group resources based on administrative and security boundaries. In active directory a domain is made up of a hierarchical structure of containers and objects, a DNS domain name as a unique identifier, a security service, and policies that dictate how functionality is restricted for users or machines within the domain.

1.3.3 Active Directory (AD)

Allen, et al. (2006) describe Microsoft Active Directory (AD), it is built on top of Windows Server, it enables administrators to manage enterprise-wide information efficiently from a central repository. It contains information about users, groups, computers, printers, applications and services; and can limit access to this information.

1.3.4 Domain Name System (DNS)

The Domain Name System (DNS), specified by a Internet Engineering Task Force (IETF) standard, is distributed database that contains the mappings between DNS domain names and various types of data including IP addresses (Microsoft, 2013). It allows the IP address of a resource, which the machine requires, to be located using a user friendly name.

1.3.5 Domain Controller

A domain controller is a Windows Server running Active Directory, it stores directory data and manages user and domain interactions, including user logon processes, authentication, and directory searches (Allen, et al., 2006).

1.4 References

Allen, R., Lowe-Norris, A. G. & Richards, J., 2006. *Active Directory.* 3rd ed. s.l.:O'Reilly.

Microsoft, 2013. *Domain Name System (DNS) on Microsoft TechNet.* [Online]
Available at: http://technet.microsoft.com/en-US/network/bb629410.aspx
[Accessed 5 November 2013].

Scambray, J. & McClure, S., 2008. *Hacking Windows Exposed.* 3rd ed. United States: McGraw Hill.

2 Overall description

2.1 Product Perspective

The system will stand alone on a host that is not joined to the Windows domain and be used to gather information from the Windows domain on the local area network on which it is deployed.

The system will communicate with a variety of systems within the Windows domain including: the DNS server, domain controller and member servers. The DNS server will provide DNS records at the request of the system. The domain controller will authenticate the system, where necessary, and provide information from the active directory.

The enumerated information will be returned to the user in a readable and parseable form.

2.1.1 System Interfaces

The system will utilise network services such as DNS and NetBIOS to enumerate information. This will be accomplished using the Windows API and the Python DNS library which in turn will use the machine's physical network connection.

In order to save the information that has been enumerated for later reference the file system will be utilise to store an XML file.

2.1.2 User interfaces / Operations

The product is a command line utility, and is therefore not interactive. When running the program the user must specify one or more command line arguments which dictate the behaviour of the application.

One of these command line arguments specifies the level of logging messages that should be displayed to the user.

The user should be able to specify 'targeting' parameters for the automated enumeration, e.g. the user can specify the domain name and domain controller that should be enumerated.

2.1.3 Hardware interfaces

The application requires network connectivity through either a wired or wireless network connection.

2.1.4 Software interfaces

This section identifies the libraries which will be used to interact with other software.

Name: The Python for Windows Extensions

Mnemonic: PyWin23

Version Number: 218.3

Source: Active State Python http://www.activestate.com/activepython

Purpose of Interfacing: Access to Windows API functions which will be used to perform enumeration of the Windows Active Directory.

Definition of Interface: Interface is well documented at http://docs.activestate.com/activepython/2.7/ and http://msdn.microsoft.com/en-us/library/windows/desktop/ff818516(v=vs.85).aspx

Name: dnspython

Version Number: 1.11.1

Source: http://www.dnspython.org/

Purpose of Interfacing: Provides high and low level classes to perform DNS queries.

Definition of Interface: Interface documentation is available at http://www.dnspython.org/docs/1.11.1/

2.1.5 Communications interfaces

Communications are handled using the libraries specified above. Windows Active Directory Service Functions communicate using TCP/IP. DNS queries and responses use the User Datagram Protocol (UDP), unless the response data exceeds 512 bytes, in which case the Transmission Control Protocol (TCP) is used.

2.1.6 Memory

There are no constraints placed on the amount of memory the application can use.

2.1.7 Site adaptation requirements

The application is designed to be run on many different Windows domain based networks. It is intended to be installed on a machine that will be taken to each site during a security assessment. There are therefore no initialisation sequences that are site specific.

2.2 Product Functions

Automated enumeration:

The application will determine the domain names on the network and their domain controllers. Using the specified credentials the application will connect to a domain controller. If the default null credentials are used the application will attempt to connect to each domain controller in turn until a session is established or there are no more domain controllers. Once a connection to a domain controller is established the application will retrieve the accounts policy, list of administrator groups, list of members of the administrator groups, administrator group members account information.

The user can specify some or all of this information to 'target' the automated enumeration.

Manual enumeration:

The user can select, using command line options, a single category of information to enumerate. The categories are:

1) Domain names on a local area network
2) Domain controllers
3) List of users from the active directory
4) List of groups from the active directory
5) List of group members from the active directory
6) User account information from the active directory
7) Domain accounts policy (lockout: duration, threshold, observation window. Password: length, character set)

2.3 User Characteristics

The intended user of this product is a computer security practitioner or windows domain administrator. It is therefore assumed that the user has a working knowledge of the Windows Active Directory structure and the information which it contains.

As a computer security practitioner is often external to the organisation on which they are conducting an assessment, it is assumed that the user does not have any prior knowledge of the Windows domain on which the product will be used.

2.4 Constraints

There are no additional constraints which limit the developer's options.

2.5 Assumptions and Dependencies

The application will be run on a Windows operating system newer than Windows 2000.

The host has a network connection and valid IP address information (IP address, default gateway, default DNS server).

The product is being run in a Windows Active Directory domain.

2.6 Apportioning of Requirements

The requirements specified in Section "3.1.2 Desirable" above may be delayed until a future version of the product.

Appendix C: UML Designs

1. Use Cases

2.7 Enumerate domain names

Use Case ID:	UC1		
Use Case Name:	Enumerate Domain Names		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	3/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather domain names and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured.
Postconditions:	Success guarantees: Discovered domain names reported in output file. Program exits.
Normal Flow:	1. The user directs the system to enumerate domain names 2. The system retrieves the domain names from the network 3. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	If the system fails to find domain names it will display a message (Error Message Use Case) to the user and not create the file.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of domain names.
Assumptions:	The application is being run in a Windows domain environment.

Notes and Issues:	User instructs the application to enumerate domain name through command line argument. NetBIOS and DNS domain names should be gathered. There is likely to be DNS domain name for each NetBIOS domain name, i.e two names for the same domain. If possible NetBIOS names should be mapped to DNS names and reported together. NetBIOS Domain names are legacy and may be removed from the Windows domain. DNS domain names should therefore take precedence over NetBIOS domain names.

2.8 Enumerate domain controller

Use Case ID:	UC2		
Use Case Name:	Enumerate Domain Controllers		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather domain controllers for each domain name and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. Domain Name Known
Postconditions:	Success guarantees: Discovered domain controllers reported in output file. Program exits.
Normal Flow:	1. The user directs the system to enumerate domain controllers 2. The system retrieves the domain controllers for the specified domain name from the network 3. The enumerated information is returned to the calling use case.
Alternative Flows: [Alternative Flow 1 – Not in Network]	
Exceptions:	If the system fails to find domain controllers it will display a message (Error Message Use Case) to the.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of domain controllers. Report IP address and fully qualified domain name (FQDN)
Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to enumerate domain controllers through command line argument. Fully qualified DNS domain names (FQDN) should be gathered along with IP address. If possible NetBIOS names should be mapped to DNS names and reported together. NetBIOS names are legacy and may be removed from the Windows domain. DNS domain names should therefore take precedence over NetBIOS domain names.

2.9 Enumerate domain groups

Use Case ID:	UC3		
Use Case Name:	Enumerate Domain Groups		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather domain groups from the domain controller and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. Domain name and domain controller known.
Postconditions:	Success guarantees: Discovered domain groups reported in output file. Program exits.
Normal Flow:	1. The user directs the system to enumerate domain groups 2. The system authenticates to the domain controller (Authentication Use Case) 3. The system retrieves the list of domain group names and group comment for the specified domain name from the domain controller. 4. They system deauthenticates from the domain controller. (Deauthentication Use Case) 5. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If system fails to authenticate to the domain controller it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find domain groups it will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the domain controller an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of domain groups.
Assumptions:	The application is being run in a Windows domain environment.

Notes and Issues:	User instructs the application to enumerate domain groups through command line argument, specifying the domain name and controller.

2.10 Enumerate domain group members

Use Case ID:	UC4		
Use Case Name:	Enumerate Domain Group Members		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather a domain group's members from the domain controller and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. Domain name, domain controller and domain group name is known.
Postconditions:	Success guarantees: Discovered list of members for a domain group reported in output file. Program exits.
Normal Flow:	1. The user directs the system to enumerate a domain group's members 2. The system authenticates to the domain controller (Authentication Use Case) 3. The system retrieves the list of the domain group's members from the domain controller. 4. They system deauthenticates from the domain controller. (Deauthentication Use Case) 5. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If system fails to authenticate to the domain controller it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find the domain group's members will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the domain controller an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of domain groups.
Assumptions:	The application is being run in a Windows domain environment.

Notes and Issues:	User instructs the application to enumerate the domain group’s members through command line argument and specifies the domain name, controller and group name.

2.11 Enumerate domain user information

Use Case ID:	UC5		
Use Case Name:	Enumerate Domain User Information		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	
Actors:	Manual Enumeration Automated Use Case		
Description:	Gather a domain user's account information from the domain controller and report.		
Trigger:			
Preconditions:	Network connection and IP addresses configured. Domain name, domain controller and username known.		
Postconditions:	Success guarantees: Discovered user information reported in output file. Program exits.		
Normal Flow:	1. The user directs the system to enumerate domain user's information 2. The system authenticates to the domain controller (Authentication Use Case) 3. The system retrieves user information from the domain controller. 4. They system deauthenticates from the domain controller. (Deauthentication Use Case) 5. The enumerated information is returned to the calling use case.		
Alternative Flows: **[Alternative Flow 1 – Not in Network]**			
Exceptions:	2. If system fails to authenticate to the domain controller it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find the domain user's information it will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the domain controller an error message (Error Message Use Case) is displayed to the user and the normal flow continues.		
Includes:	Error Message Use Case		
Frequency of Use:	Potentially every instance of product use.		
Special Requirements:	No duplication of domain user information.		
Assumptions:	The application is being run in a Windows domain environment.		

Notes and Issues:	User instructs the application to enumerate the domain group’s members through command line argument and specifies the domain name, controller and group name.

2.12 Enumerate domain accounts policy

Use Case ID:	UC6		
Use Case Name:	Enumerate Domain Accounts Policy		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather a domain's account policy (observation windows, lockout threshold, lockout duration, password complexity, password age, password history length, etc) from the domain controller and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. Domain name and domain controller known.
Postconditions:	Success guarantees: Discovered domain account policy reported in output file. Program exits.
Normal Flow:	1. The user directs the system to enumerate domain account policy 2. The system authenticates to the domain controller (Authentication Use Case) 3. The system retrieves account policy from the domain controller. 4. They system deauthenticates from the domain controller. (Deauthentication Use Case) 5. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If system fails to authenticate to the domain controller it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find the domain account policy it will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the domain controller an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of domain user information.

Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to enumerate the domain account policy through command line argument and specifies the domain name and controller.

2.13 Enumerate domain trusts

Use Case ID:	UC7		
Use Case Name:	Enumerate Domain Trusts		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather the trusts between domains and report
Trigger:	
Preconditions:	Network connection and IP addresses configured. Domain name and a member server (or domain controller) are known.
Postconditions:	Success guarantees: Discovered domain trusts reported in output file. Program exits.
Normal Flow:	1. The user directs the system to enumerate domain trusts 2. The system authenticates to the specified server (Authentication Use Case) 3. The system retrieves account policy from the domain controller. 4. They system deauthenticates from the specified server. (Deauthentication Use Case) 5. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If system fails to authenticate to the domain controller it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find the domain trusts it will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the domain controller an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of domain user information.
Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to enumerate the domain trusts through command line argument and specifies the domain name and a member server.

2.14 Enumerate local groups

Use Case ID:	UC8		
Use Case Name:	Enumerate Local Groups		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather local groups from the specified host and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. FQDN or IP address of host is known.
Postconditions:	Success guarantees: Discovered local groups reported in output file. Program exits.
Normal Flow:	6. The user directs the system to enumerate local groups 7. The system authenticates to the specified host (Authentication Use Case) 8. The system retrieves the list of local group names and group comment for the specified local name from the specified host. 9. They system deauthenticates from the specified host. (Deauthentication Use Case) 10. The enumerated information is returned to the calling use case.
Alternative Flows: [Alternative Flow 1 – Not in Network]	
Exceptions:	2. If system fails to authenticate to the specified host it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find local groups it will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the specified host an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of local groups.
Assumptions:	The application is being run in a Windows domain environment.

Notes and Issues:	User instructs the application to enumerate local groups through command line argument, specifying the local name and controller.

2.15 Enumerate local group members

Use Case ID:	UC9		
Use Case Name:	Manual Enumeration Automated Use Case		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather a local group's members from the specified host and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. FQDN or IP address of host the group name is known.
Postconditions:	Success guarantees: Discovered list of members for a local group reported in output file. Program exits.
Normal Flow:	6. The user directs the system to enumerate a local group's members 7. The system authenticates to the specified host (Authentication Use Case) 8. The system retrieves the list of the local group's members from the specified host. 9. They system deauthenticates from the specified host. (Deauthentication Use Case) 10. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If system fails to authenticate to the specified host it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find the local group's members will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the specified host an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of local groups.

Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to enumerate the local group's members through command line argument and specifies the local name, controller and group name.

2.16 Enumerate local user information

Use Case ID:	UC10		
Use Case Name:	Enumerate Local User Information		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather a local user's account information from the specified host and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. FQDN or IP address of host and username is known.
Postconditions:	Success guarantees: Discovered user information reported in output file. Program exits.
Normal Flow:	6. The user directs the system to enumerate local user's information 7. The system authenticates to the specified host (Authentication Use Case) 8. The system retrieves user information from the specified host. 9. They system deauthenticates from the specified host. (Deauthentication Use Case) 10. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If system fails to authenticate to the specified host it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find the local user's information it will display a message (Error Message Use Case) to the user. 4. If the system fails to deauthenticate from the specified host an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of local user information.
Assumptions:	The application is being run in a Windows domain environment.

Notes and Issues:	User instructs the application to enumerate the local group's members through command line argument and specifies the local name, controller and group name.

2.17 Enumerate local accounts policy

Use Case ID:	UC11		
Use Case Name:	Enumerate Local Accounts Policy		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather a local's account policy (observation windows, lockout threshold, lockout duration, password complexity, password age, password history length, etc) from the specified host and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured. FQDN or IP address of host is known.
Postconditions:	Success guarantees: Discovered local account policy reported in output file. Program exits.
Normal Flow:	6. The user directs the system to enumerate local account policy 7. The system authenticates to the specified host (Authentication Use Case) 8. The system retrieves account policy from the specified host. 9. They system deauthenticates from the specified host. (Deauthentication Use Case) 10. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If system fails to authenticate to the specified host it will display a message (Error Message Use Case) to the user and not create the file. 3. If the system fails to find the local account policy it will display a message (Error Message Use Case) to the user and not creating the file. 4. If the system fails to deauthenticate from the specified host an error message (Error Message Use Case) is displayed to the user and the normal flow continues.
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of local user information.

Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to enumerate the local account policy through command line argument and specifies the local name and controller.

2.18 Enumerate Hosts with Services

Use Case ID:	UC12		
Use Case Name:	Enumerate Hosts with Services		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Gather a list of hosts running interesting services (SQL server etc) and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured.
Postconditions:	Success guarantees: Discovered hosts reported in output file. Program exits.
Normal Flow:	1. The user directs the system to enumerate machines with 'interesting' services running. 2. The system queries the DNS serer for a list of hosts running specific services. 3. The enumerated information is returned to the calling use case.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	2. If the system fails to find hosts it will display a message (Error Message Use Case) to the user.
Includes:	
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	No duplication of local hosts per service.
Assumptions:	The application is being run in a Windows domain environment. Hosts are configured to advertise their services through DNS service (SRV) records.
Notes and Issues:	User instructs the application to enumerate hosts running 'interesting' services through command line argument.

2.19 Automated Domain Enumeration

Use Case ID:	UC13		
Use Case Name:	Automated Domain Enumeration		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	User
Description:	Conduct automated enumeration of the domain and report.
Trigger:	
Preconditions:	Network connection and IP addresses configured.
Postconditions:	Success guarantees: Discovered information reported in output file. Program exits.
Normal Flow:	1. The user directs the system to perform automated enumeration of the domain. 2. Domain Names are enumerated (Enumerate Domain Name Use Case) For each domain name: 3. Domain controllers are enumerated (Enumerate Domain Controllers Use Case) 4. Attempt to authenticate to each domain controller (Authentication Use Case) with default null credentials until authentication is successful or all possibilities have been attempted. 5. A List of domain groups are enumerated (Enumerate Domain Groups Use Case) and each name is checked to determine if it contains the string ‘admin’ or if it is in the list of groups to enumerate. 6. For each group a list of members is enumerated (Enumerate Domain Group Members Use Case) 7. For each member the user’s account information is enumerated (Enumerate Domain User Information Use Case) 8. Enumerate the domain accounts policy (Enumerate Domain Accounts Policy Use Case) 9. Enumerate domain trusts (Enumerate Domain Trusts Use Case)

	10. Discover hosts running ‘interesting’ services (Enumerate Hosts with Services) 11. The system creates a file with the enumerated information (Reporter Use Case).
Alternative Flows: [Alternative Flow 1 – Not in Network]	If the user specifies a domain name, step 2 is skipped. If the user specifies a domain controller, step 3 is skipped If user specifies credentials are being used step 4 is replaced with: Authenticate to a domain controller (Authentication Use Case). If the user specifies a domain group, step 5 is skipped. If new domain names are discovered in step 9 repeat the use case for each.
Exceptions:	At 3 if the system fails to discover domain names it will display a message (Error Message Use Case) to the user and skip to 10. If authentication fails at 4 display a message (Error Message Use Case) to the user and skip to 10. If any of 5-9 fail, skip to the next step. If all steps failed do not create an output file.
Includes:	Error Message Use Case Reporter Use Case Enumerate Domain Name Use Case Enumerate Domain Controllers Use Case Authentication Use Case Enumerate Domain Groups Use Case Enumerate Domain Group Members Use Case Enumerate Domain User Information Use Case Enumerate Domain Accounts Policy Use Case Enumerate Domain Trusts Use Case Enumerate Hosts with Services
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	
Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to perform automated domain enumeration, and can specify some targeting options (i.e. supply domain name etc) through command line argument. Steps 7-10 cannot be specified by the user so cannot be skipped in this use case.

2.20 Automated Enumeration of Host

Use Case ID:	UC14		
Use Case Name:	Automated Enumeration of Host Automated Use Case		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	
Actors:	User		
Description:	Conduct automated enumeration of a host and report.		
Trigger:			
Preconditions:	Network connection and IP addresses configured.		
Postconditions:	Success guarantees: Discovered information reported in output file. Program exits.		
Normal Flow:	1. The user directs the system to perform automated enumeration a local host. 2. Hosts running ‘interesting’ services discovered (Enumerate Hosts with Services) For each host: 3. Authenticate to the host (Authentication Use Case). 4. A list of local groups are enumerated (Enumerate Local Groups Use Case) and each name is checked to determine if it contains the string ‘admin’ or if it is in the list of groups to enumerate. 5. For each group a list of members is enumerated (Enumerate Local Group Members Use Case) 6. For each member the user’s account information is enumerated (Enumerate Local User Information Use Case) 7. Enumerate the local accounts policy (Enumerate Local Accounts Policy Use Case) 8. Enumerate domain trusts (Enumerate Domain Trusts Use Case) 9. The enumerated information is returned to the calling use case.		
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	If the user specifies host(s), skip step 2. If the use specifies a group name, step 4 is skipped.		

Exceptions:	If no hosts are discovered in step 2, display and error message to the user, skip the remaining steps and do not create a file. If authentication fails in step 3, display an error message to the use and skip to the next host. If any of steps 4-8 fail, skip to the next step.
Includes:	Error Message Use Case Reporter Use Case Authentication Use Case Enumerate Local Groups Use Case Enumerate Local Group Members Use Case Enumerate Local User Information Use Case Enumerate Local Accounts Policy Use Case Enumerate Doman Trusts Use Case Enumerate Hosts with Services
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	Do not enumerate the same host twice.
Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to perform automated host enumeration, and can specify some targeting options (i.e. supply host(s) to enumerate etc) through command line argument. Steps 6-8 cannot be specified by the user so cannot be skipped in this use case.

2.21 Manual Enumeration

Use Case ID:	UC15		
Use Case Name:	User		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	User
Description:	Perform a single enumeration exercise based on user input
Trigger:	
Preconditions:	Network connection and IP addresses configured.
Postconditions:	Success guarantees: Discovered information reported in output file. Program exits.
Normal Flow:	1. The user directs the system to perform one of several enumerations. 2. The appropriate Use Case for the enumeration is called: • Enumerate Domain Name Use Case • Enumerate Domain Controllers Use Case • Enumerate Domain Groups Use Case • Enumerate Domain Group Members Use Case • Enumerate Domain User Information Use Case • Enumerate Domain Accounts Policy Use Case • Enumerate Local Controllers Use Case • Enumerate Local Groups Use Case • Enumerate Local Group Members Use Case • Enumerate Local User Information Use Case • Enumerate Local Accounts Policy Use Case • Enumerate Doman Trusts Use Case • Enumerate Hosts with Services 3. The system creates a file with the enumerated information (Reporter Use Case).
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	If insufficient user input I provided, display an error (Error Message Use Case). If the enumeration fails do not create an output file.

Includes:	Error Message Use Case Reporter Use Case Enumerate Local Groups Use Case Enumerate Local Group Members Use Case Enumerate Local User Information Use Case Enumerate Local Accounts Policy Use Case Enumerate Doman Trusts Use Case Enumerate Hosts with Services
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	User must specify pre requisite input for the enumeration.
Assumptions:	The application is being run in a Windows domain environment.
Notes and Issues:	User instructs the application to perform a specific enumeration activity, and specifies the perquisite information (e.g. supply host(s) to enumerate) through command line argument.

2.22 Reporter

Use Case ID:	UC16		
Use Case Name:	Reporter		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case
Description:	Output results of enumeration to file
Trigger:	
Preconditions:	Enumeration complete.
Postconditions:	Success guarantees: If results exist they have been written to a file. Program exits.
Normal Flow:	1. Reporter is called and passed information to write to file. 2. XML structure is created containing the information. 3. XML structure is written to a file in a human readable form.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	
Exceptions:	If reporter is not passed any information to enumerate an error will be returned to the user (Error Message Use Case).
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	
Assumptions:	Sufficient space on hard disk for output file.
Notes and Issues:	

2.23 Authentication

Use Case ID:	UC17		
Use Case Name:	Authentication		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case Enumerate Domain Name Use Case Enumerate Domain Controllers Use Case Enumerate Domain Groups Use Case Enumerate Domain Group Members Use Case Enumerate Domain User Information Use Case Enumerate Domain Accounts Policy Use Case Enumerate Local Groups Use Case Enumerate Local Group Members Use Case Enumerate Local User Information Use Case Enumerate Local Accounts Policy Use Case Enumerate Doman Trusts Use Case Enumerate Hosts with Services
Description:	Authenticate to specified host
Trigger:	
Preconditions:	Network connection and IP addresses configured. Host to authenticate to and credentials are known.
Postconditions:	Success guarantees: Session established with specified host. Program exits.
Normal Flow:	1. Check if already authenticated to the host. 2. If not authenticated, authenticate to host with given credentials
Alternative Flows: [Alternative Flow 1 – Not in Network]	If already authenticated to host, do nothing.
Exceptions:	If there is an error authenticating to the host, output an error message to the user (Error Message Use Case)
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	
Assumptions:	
Notes and Issues:	

2.24 Deauthentication

Use Case ID:	UC18		
Use Case Name:	Deauthentication		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

Actors:	Manual Enumeration Automated Use Case Enumerate Domain Name Use Case Enumerate Domain Controllers Use Case Enumerate Domain Groups Use Case Enumerate Domain Group Members Use Case Enumerate Domain User Information Use Case Enumerate Domain Accounts Policy Use Case Enumerate Local Groups Use Case Enumerate Local Group Members Use Case Enumerate Local User Information Use Case Enumerate Local Accounts Policy Use Case Enumerate Doman Trusts Use Case Enumerate Hosts with Services
Description:	deauthenticate from specified host
Trigger:	
Preconditions:	Network connection and IP addresses configured. Host known. Host authenticated to.
Postconditions:	Success guarantees: Session terminated with specified host. Program exits.
Normal Flow:	1. Check if already authenticated to the host. 2. If authenticated, deauthenticate from host
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	If already deauthenticated to host, do nothing.
Exceptions:	If there is an error deauthenticating to the host, output an error message to the user (Error Message Use Case)
Includes:	Error Message Use Case
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	
Assumptions:	
Notes and Issues:	

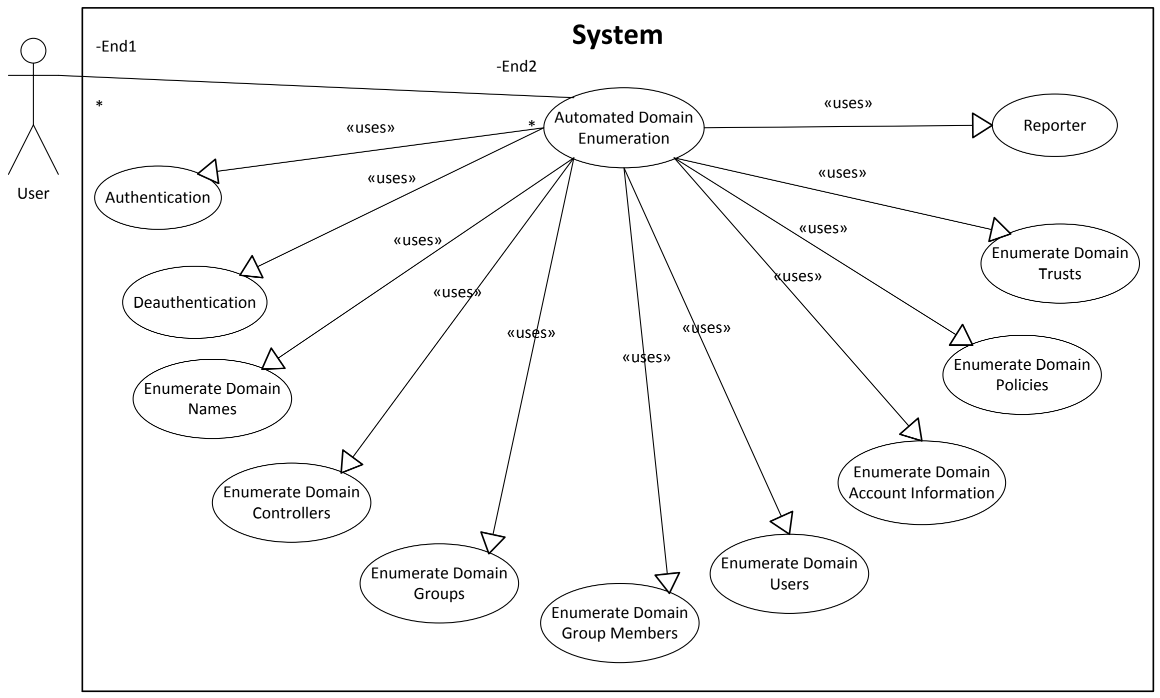
2.25 Error Message

Use Case ID:	UC18		
Use Case Name:	Error Message		
Created By:	Oliver Morton	**Last Updated By:**	
Date Created:	9/12/2013	**Last Revision Date:**	

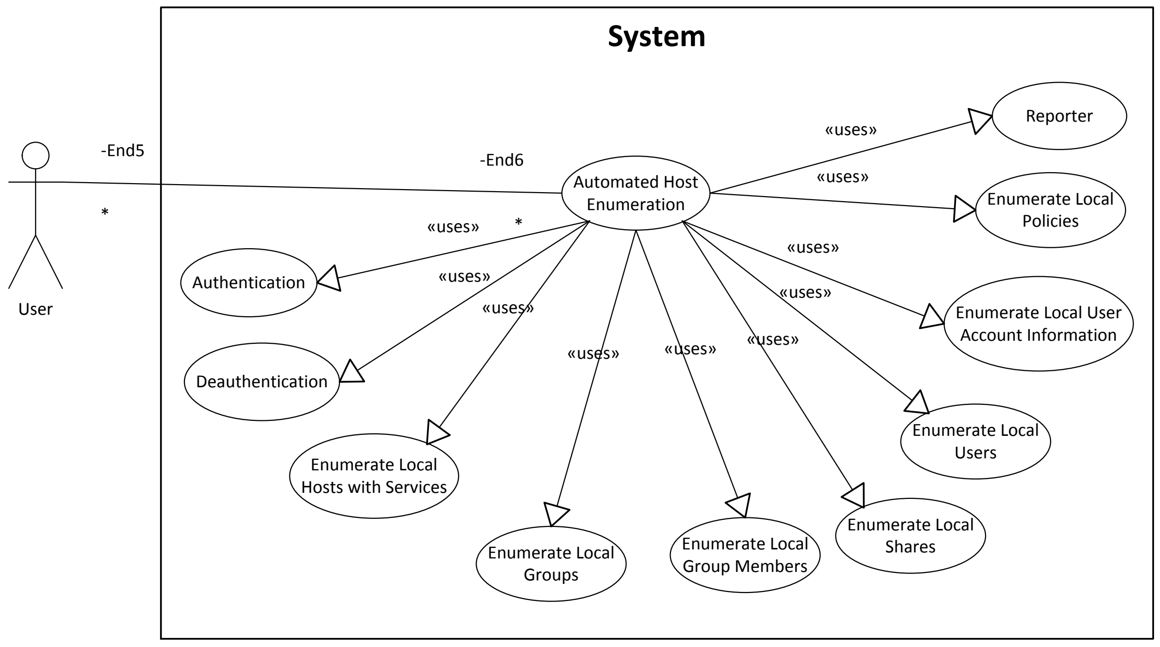
Actors:	Manual Enumeration Automated Use Case Enumerate Domain Name Use Case Enumerate Domain Controllers Use Case Enumerate Domain Groups Use Case Enumerate Domain Group Members Use Case Enumerate Domain User Information Use Case Enumerate Domain Accounts Policy Use Case Enumerate Local Groups Use Case Enumerate Local Group Members Use Case Enumerate Local User Information Use Case Enumerate Local Accounts Policy Use Case Enumerate Doman Trusts Use Case Enumerate Hosts with Services
Description:	Print error message to user
Trigger:	
Preconditions:	Message known.
Postconditions:	Success guarantees: Message written to terminal. Program exits.
Normal Flow:	1. Write message to terminal.
Alternative Flows: [Alternative Flow 1 – Not in Network]	
Exceptions:	
Includes:	
Frequency of Use:	Potentially every instance of product use.
Special Requirements:	
Assumptions:	
Notes and Issues:	

2. Use Case Diagrams

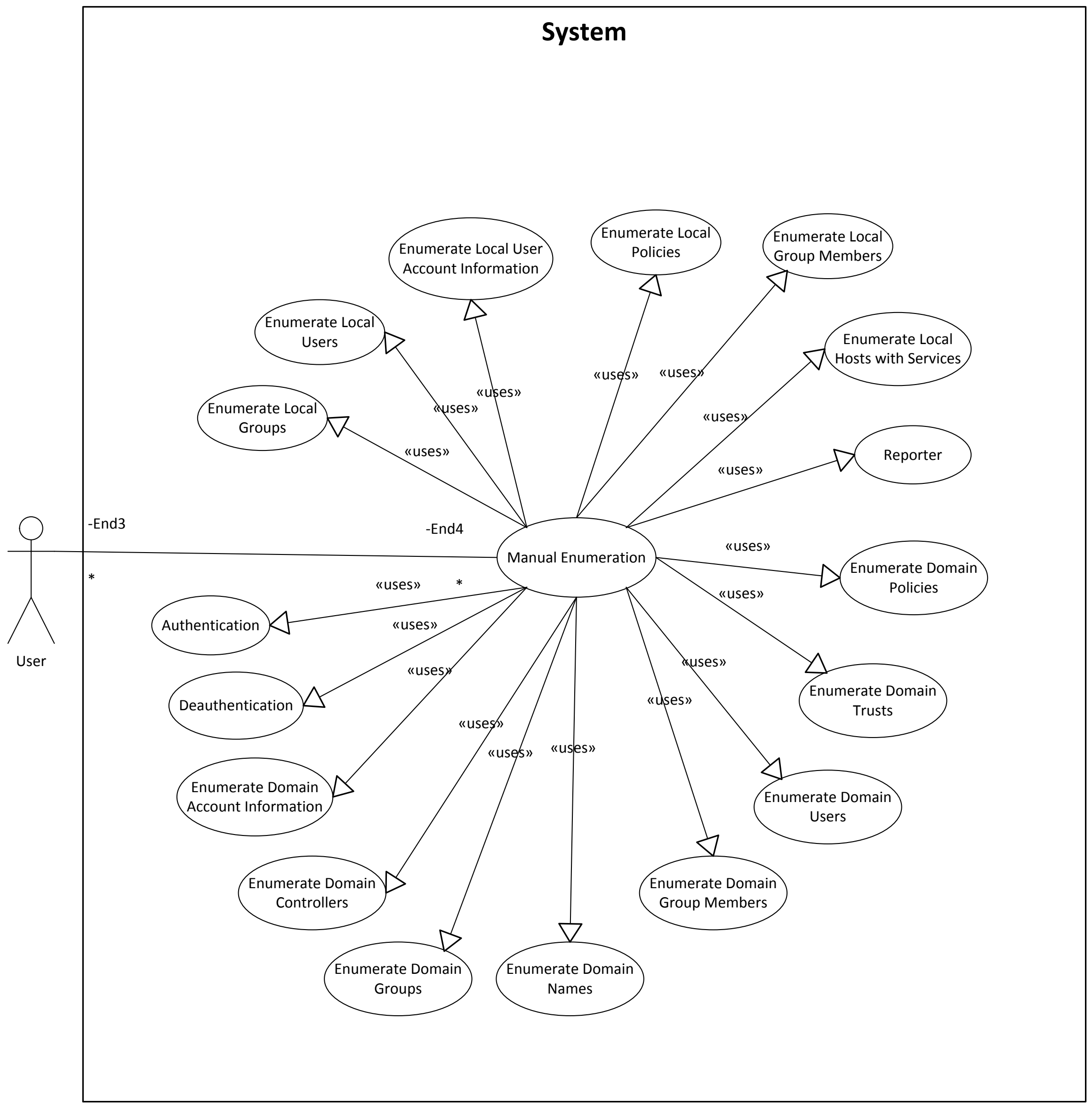
2.26 Automated Domain Enumeration



2.27 Automated Local Enumeration

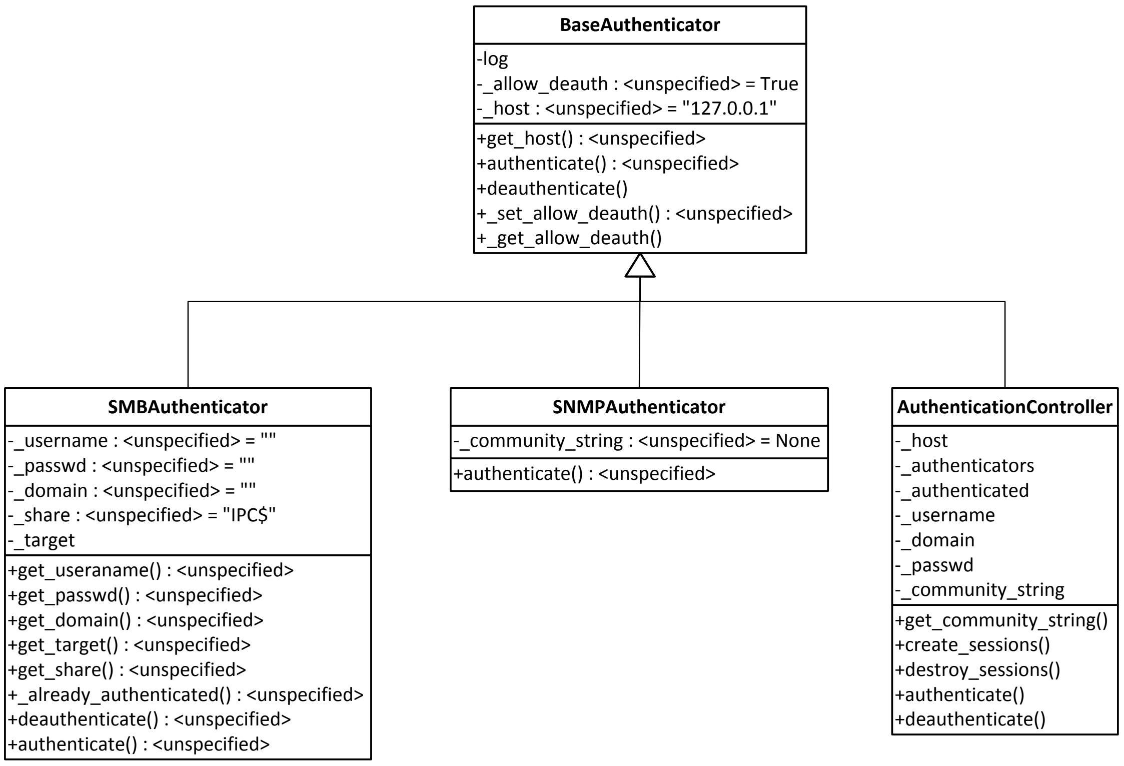


2.28 Manual Enumeration

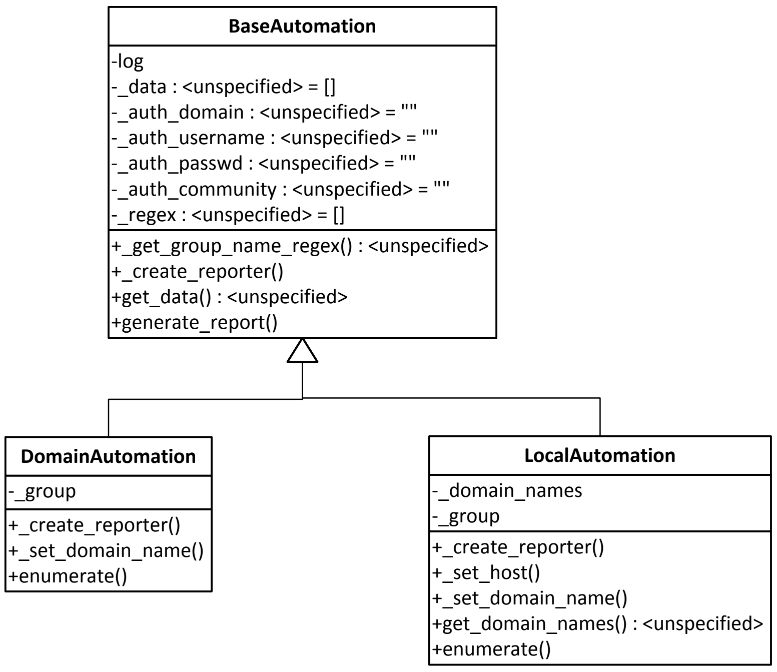

System
User
-End3
*
-End4
*
Manual Enumeration
Enumerate Local User Account Information
Enumerate Local Policies
Enumerate Local Group Members
Enumerate Local Users
Enumerate Local Hosts with Services
Enumerate Local Groups
Reporter
Enumerate Domain Policies
Authentication
Deauthentication
Enumerate Domain Trusts
Enumerate Domain Account Information
Enumerate Domain Users
Enumerate Domain Controllers
Enumerate Domain Group Members
Enumerate Domain Groups
Enumerate Domain Names
«uses»

3. Class Diagrams

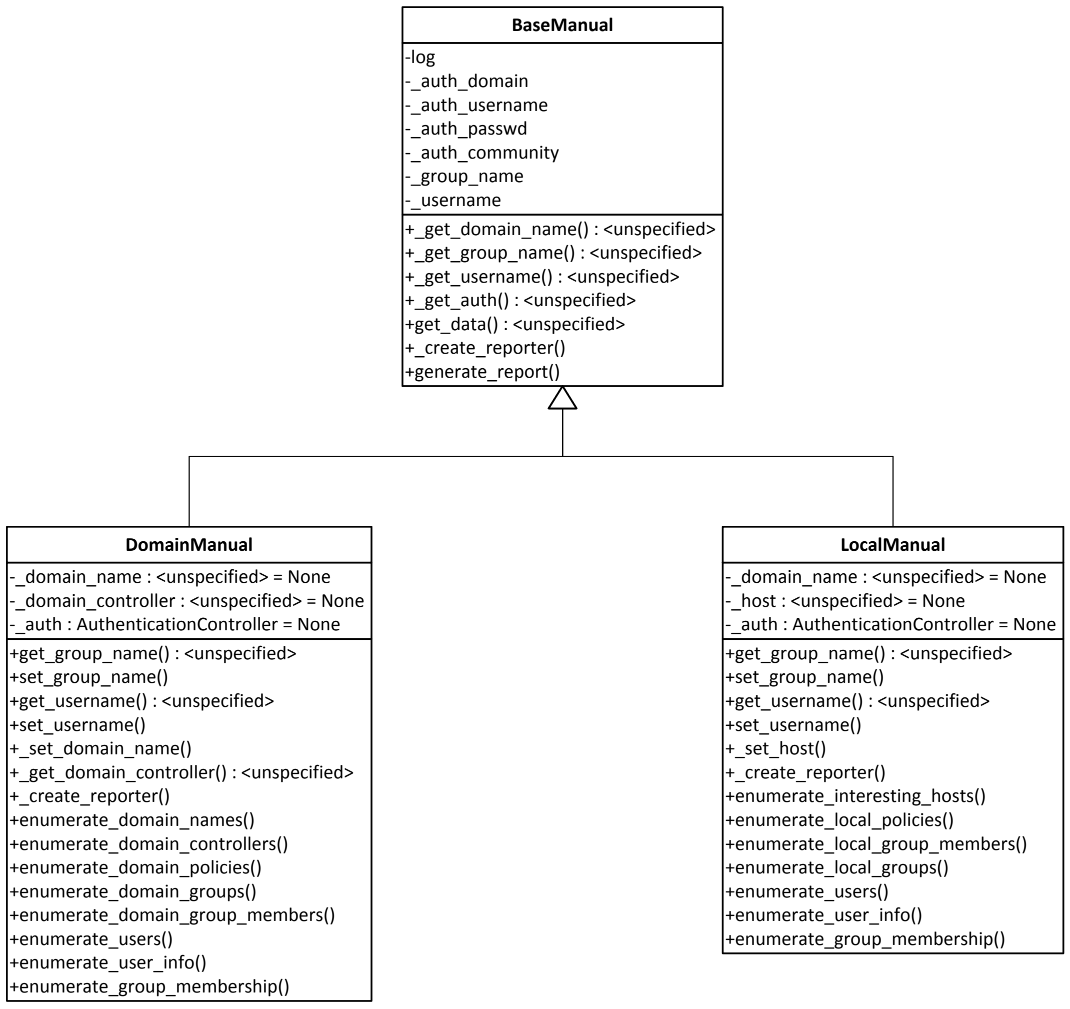
2.29 Authenticators


BaseAuthenticator
-log
-_allow_deauth : <unspecified> = True
-_host : <unspecified> = "127.0.0.1"
+get_host() : <unspecified>
+authenticate() : <unspecified>
+deauthenticate()
+_set_allow_deauth() : <unspecified>
+_get_allow_deauth()
SMBAuthenticator
-_username : <unspecified> = ""
-_passwd : <unspecified> = ""
-_domain : <unspecified> = ""
-_share : <unspecified> = "IPC$"
-_target
+get_useraname() : <unspecified>
+get_passwd() : <unspecified>
+get_domain() : <unspecified>
+get_target() : <unspecified>
+get_share() : <unspecified>
+_already_authenticated() : <unspecified>
+deauthenticate() : <unspecified>
+authenticate() : <unspecified>
SNMPAuthenticator
-_community_string : <unspecified> = None
+authenticate() : <unspecified>
AuthenticationController
-_host
-_authenticators
-_authenticated
-_username
-_domain
-_passwd
-_community_string
+get_community_string()
+create_sessions()
+destroy_sessions()
+authenticate()
+deauthenticate()

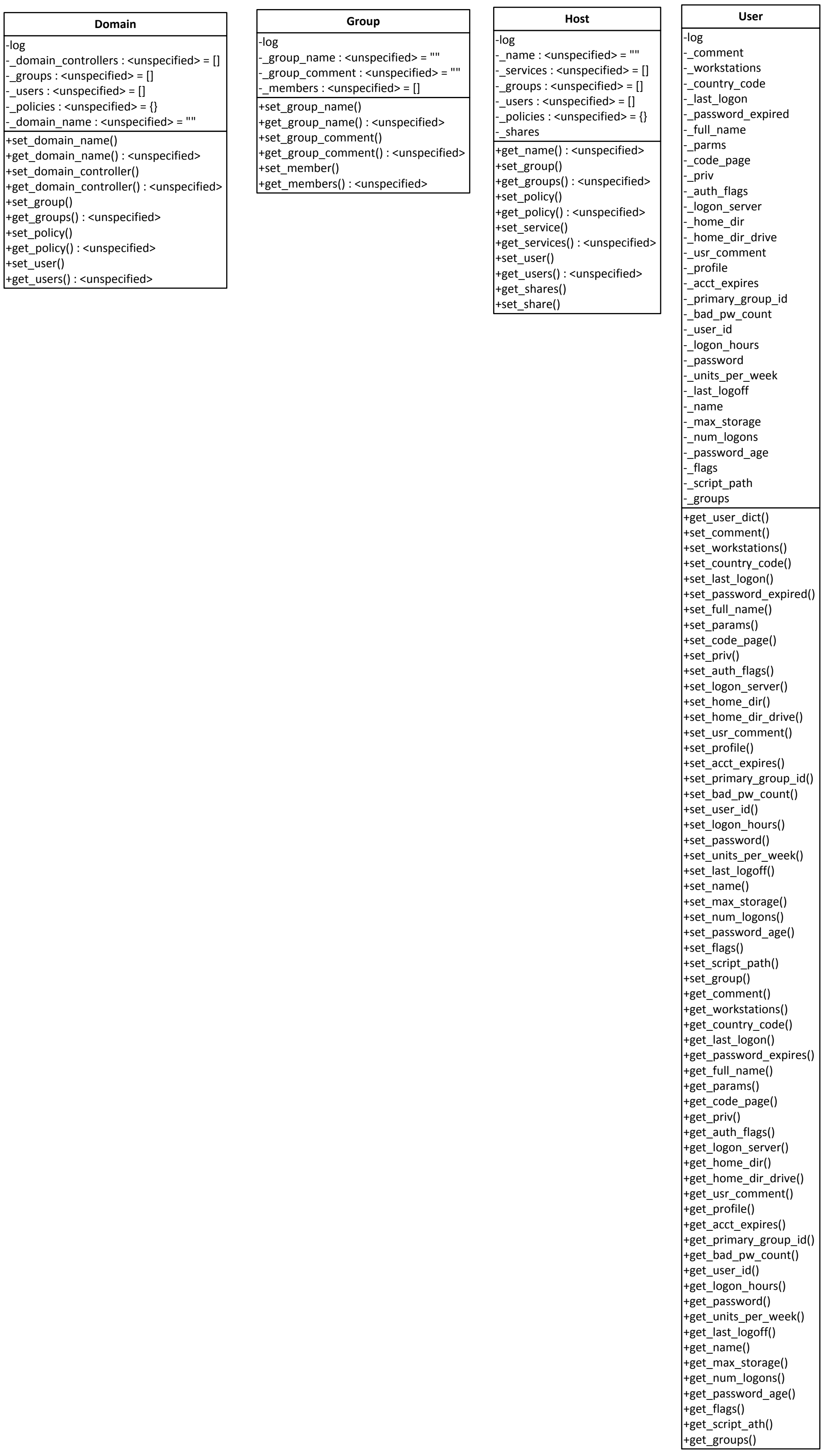
2.30 Automation


BaseAutomation
-log
-_data : <unspecified> = []
-_auth_domain : <unspecified> = ""
-_auth_username : <unspecified> = ""
-_auth_passwd : <unspecified> = ""
-_auth_community : <unspecified> = ""
-_regex : <unspecified> = []
+_get_group_name_regex() : <unspecified>
+_create_reporter()
+get_data() : <unspecified>
+generate_report()
DomainAutomation
-_group
+_create_reporter()
+_set_domain_name()
+enumerate()
LocalAutomation
-_domain_names
-_group
+_create_reporter()
+_set_host()
+_set_domain_name()
+get_domain_names() : <unspecified>
+enumerate()

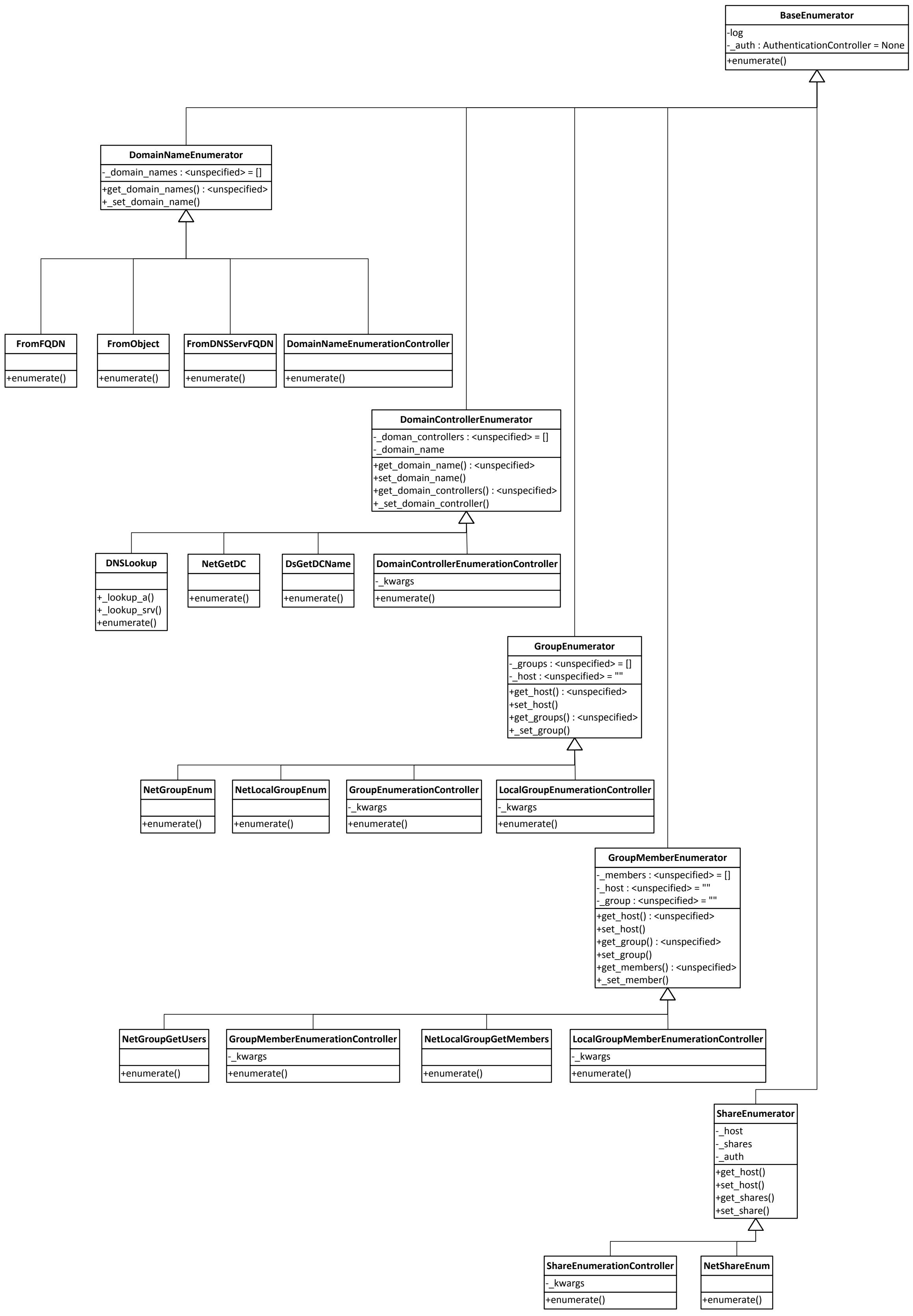
2.31 Manual

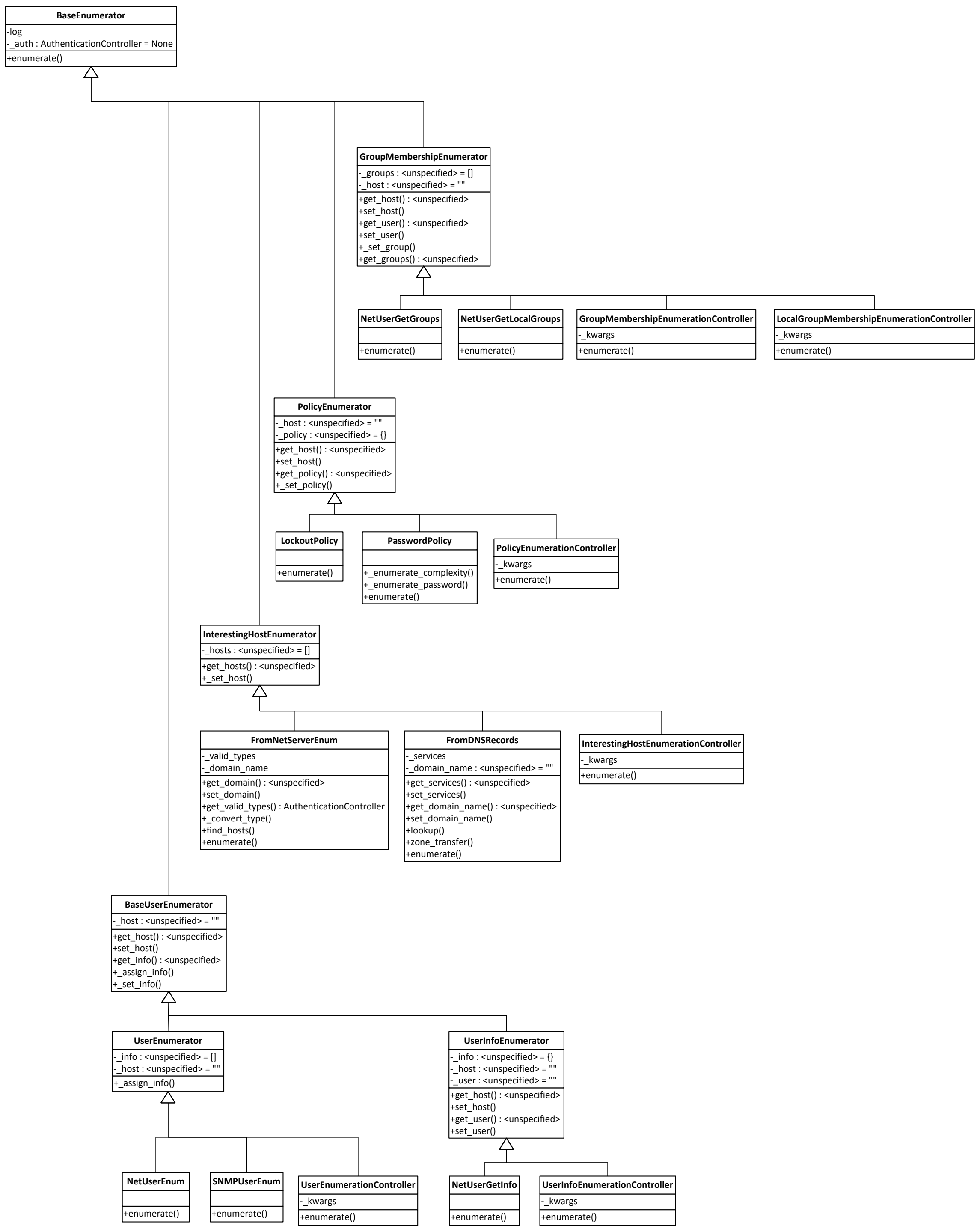

BaseManual
-log
-_auth_domain
-_auth_username
-_auth_passwd
-_auth_community
-_group_name
-_username
+_get_domain_name() : <unspecified>
+_get_group_name() : <unspecified>
+_get_username() : <unspecified>
+_get_auth() : <unspecified>
+get_data() : <unspecified>
+_create_reporter()
+generate_report()
DomainManual
-_domain_name : <unspecified> = None
-_domain_controller : <unspecified> = None
-_auth : AuthenticationController = None
+get_group_name() : <unspecified>
+set_group_name()
+get_username() : <unspecified>
+set_username()
+_set_domain_name()
+_get_domain_controller() : <unspecified>
+_create_reporter()
+enumerate_domain_names()
+enumerate_domain_controllers()
+enumerate_domain_policies()
+enumerate_domain_groups()
+enumerate_domain_group_members()
+enumerate_users()
+enumerate_user_info()
+enumerate_group_membership()
LocalManual
-_domain_name : <unspecified> = None
-_host : <unspecified> = None
-_auth : AuthenticationController = None
+get_group_name() : <unspecified>
+set_group_name()
+get_username() : <unspecified>
+set_username()
+_set_host()
+_create_reporter()
+enumerate_interesting_hosts()
+enumerate_local_policies()
+enumerate_local_group_members()
+enumerate_local_groups()
+enumerate_users()
+enumerate_user_info()
+enumerate_group_membership()

2.32 Data

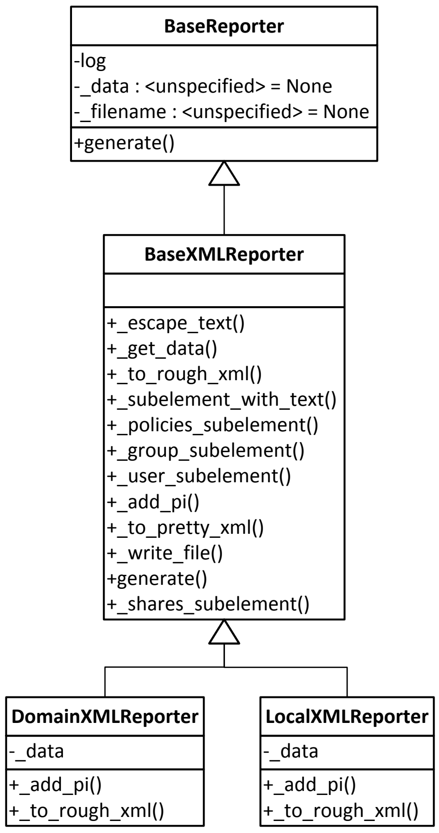

Domain
-log
-_domain_controllers : <unspecified> = []
-_groups : <unspecified> = []
-_users : <unspecified> = []
-_policies : <unspecified> = {}
-_domain_name : <unspecified> = ""
+set_domain_name()
+get_domain_name() : <unspecified>
+set_domain_controller()
+get_domain_controller() : <unspecified>
+set_group()
+get_groups() : <unspecified>
+set_policy()
+get_policy() : <unspecified>
+set_user()
+get_users() : <unspecified>
Group
-log
-_group_name : <unspecified> = ""
-_group_comment : <unspecified> = ""
-_members : <unspecified> = []
+set_group_name()
+get_group_name() : <unspecified>
+set_group_comment()
+get_group_comment() : <unspecified>
+set_member()
+get_members() : <unspecified>
Host
-log
-_name : <unspecified> = ""
-_services : <unspecified> = []
-_groups : <unspecified> = []
-_users : <unspecified> = []
-_policies : <unspecified> = {}
-_shares
+get_name() : <unspecified>
+set_group()
+get_groups() : <unspecified>
+set_policy()
+get_policy() : <unspecified>
+set_service()
+get_services() : <unspecified>
+set_user()
+get_users() : <unspecified>
+get_shares()
+set_share()
User
-log
-_comment
-_workstations
-_country_code
-_last_logon
-_password_expired
-_full_name
-_parms
-_code_page
-_priv
-_auth_flags
-_logon_server
-_home_dir
-_home_dir_drive
-_usr_comment
-_profile
-_acct_expires
-_primary_group_id
-_bad_pw_count
-_user_id
-_logon_hours
-_password
-_units_per_week
-_last_logoff
-_name
-_max_storage
-_num_logons
-_password_age
-_flags
-_script_path
-_groups
+get_user_dict()
+set_comment()
+set_workstations()
+set_country_code()
+set_last_logon()
+set_password_expired()
+set_full_name()
+set_params()
+set_code_page()
+set_priv()
+set_auth_flags()
+set_logon_server()
+set_home_dir()
+set_home_dir_drive()
+set_usr_comment()
+set_profile()
+set_acct_expires()
+set_primary_group_id()
+set_bad_pw_count()
+set_user_id()
+set_logon_hours()
+set_password()
+set_units_per_week()
+set_last_logoff()
+set_name()
+set_max_storage()
+set_num_logons()
+set_password_age()
+set_flags()
+set_script_path()
+set_group()
+get_comment()
+get_workstations()
+get_country_code()
+get_last_logon()
+get_password_expires()
+get_full_name()
+get_params()
+get_code_page()
+get_priv()
+get_auth_flags()
+get_logon_server()
+get_home_dir()
+get_home_dir_drive()
+get_usr_comment()
+get_profile()
+get_acct_expires()
+get_primary_group_id()
+get_bad_pw_count()
+get_user_id()
+get_logon_hours()
+get_password()
+get_units_per_week()
+get_last_logoff()
+get_name()
+get_max_storage()
+get_num_logons()
+get_password_age()
+get_flags()
+get_script_ath()
+get_groups()

2.33 Enumerators

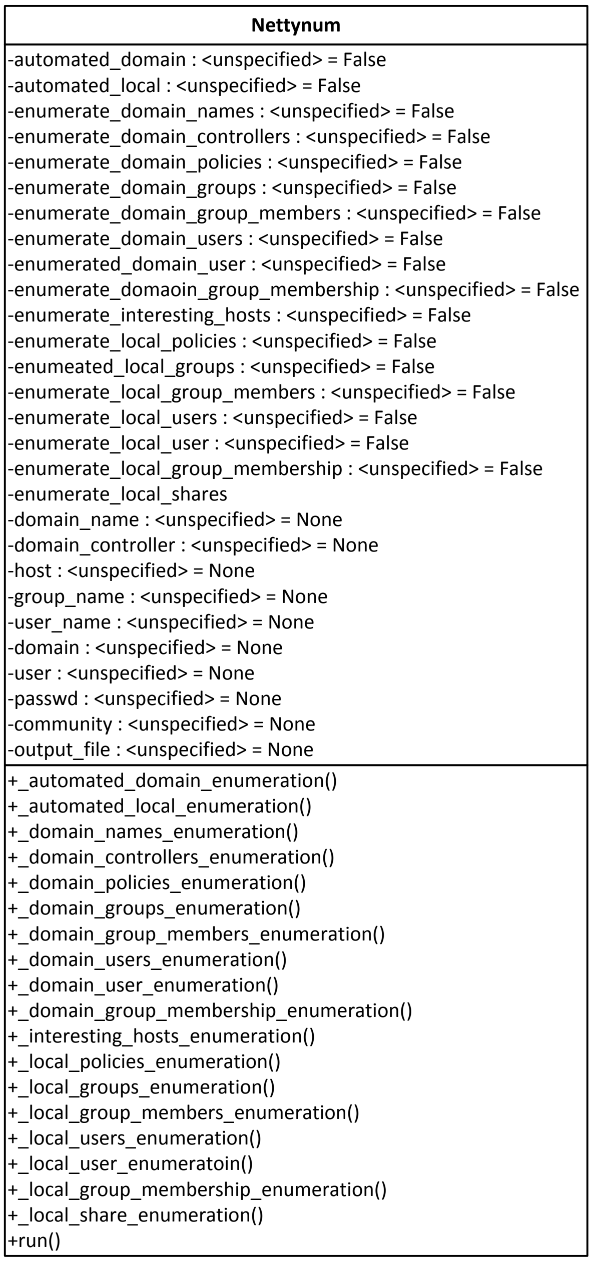

BaseEnumerator
-log
-_auth : AuthenticationController = None
+enumerate()
DomainNameEnumerator
-_domain_names : <unspecified> = []
+get_domain_names() : <unspecified>
+_set_domain_name()
FromFQDN
+enumerate()
FromObject
+enumerate()
FromDNSServFQDN
+enumerate()
DomainNameEnumerationController
+enumerate()
DomainControllerEnumerator
-_doman_controllers : <unspecified> = []
-_domain_name
+get_domain_name() : <unspecified>
+set_domain_name()
+get_domain_controllers() : <unspecified>
+_set_domain_controller()
DNSLookup
+_lookup_a()
+_lookup_srv()
+enumerate()
NetGetDC
+enumerate()
DsGetDCName
+enumerate()
DomainControllerEnumerationController
-_kwargs
+enumerate()
GroupEnumerator
-_groups : <unspecified> = []
-_host : <unspecified> = ""
+get_host() : <unspecified>
+set_host()
+get_groups() : <unspecified>
+_set_group()
NetGroupEnum
+enumerate()
NetLocalGroupEnum
+enumerate()
GroupEnumerationController
-_kwargs
+enumerate()
LocalGroupEnumerationController
-_kwargs
+enumerate()
GroupMemberEnumerator
-_members : <unspecified> = []
-_host : <unspecified> = ""
-_group : <unspecified> = ""
+get_host() : <unspecified>
+set_host()
+get_group() : <unspecified>
+set_group()
+get_members() : <unspecified>
+_set_member()
NetGroupGetUsers
+enumerate()
GroupMemberEnumerationController
-_kwargs
+enumerate()
NetLocalGroupGetMembers
+enumerate()
LocalGroupMemberEnumerationController
-_kwargs
+enumerate()
ShareEnumerator
-_host
-_shares
-_auth
+get_host()
+set_host()
+get_shares()
+set_share()
ShareEnumerationController
-_kwargs
+enumerate()
NetShareEnum
+enumerate()


BaseEnumerator
-log
-_auth : AuthenticationController = None
+enumerate()
GroupMembershipEnumerator
-_groups : <unspecified> = []
-_host : <unspecified> = ""
+get_host() : <unspecified>
+set_host()
+get_user() : <unspecified>
+set_user()
+_set_group()
+get_groups() : <unspecified>
NetUserGetGroups
+enumerate()
NetUserGetLocalGroups
+enumerate()
GroupMembershipEnumerationController
-_kwargs
+enumerate()
LocalGroupMembershipEnumerationController
-_kwargs
+enumerate()
PolicyEnumerator
-_host : <unspecified> = ""
-_policy : <unspecified> = {}
+get_host() : <unspecified>
+set_host()
+get_policy() : <unspecified>
+_set_policy()
LockoutPolicy
+enumerate()
PasswordPolicy
+_enumerate_complexity()
+_enumerate_password()
+enumerate()
PolicyEnumerationController
-_kwargs
+enumerate()
InterestingHostEnumerator
-_hosts : <unspecified> = []
+get_hosts() : <unspecified>
+_set_host()
FromNetServerEnum
-_valid_types
-_domain_name
+get_domain() : <unspecified>
+set_domain()
+get_valid_types() : AuthenticationController
+_convert_type()
+find_hosts()
+enumerate()
FromDNSRecords
-_services
-_domain_name : <unspecified> = ""
+get_services() : <unspecified>
+set_services()
+get_domain_name() : <unspecified>
+set_domain_name()
+lookup()
+zone_transfer()
+enumerate()
InterestingHostEnumerationController
-_kwargs
+enumerate()
BaseUserEnumerator
-_host : <unspecified> = ""
+get_host() : <unspecified>
+set_host()
+get_info() : <unspecified>
+_assign_info()
+_set_info()
UserEnumerator
-_info : <unspecified> = []
-_host : <unspecified> = ""
+_assign_info()
UserInfoEnumerator
-_info : <unspecified> = {}
-_host : <unspecified> = ""
-_user : <unspecified> = ""
+get_host() : <unspecified>
+set_host()
+get_user() : <unspecified>
+set_user()
NetUserEnum
+enumerate()
SNMPUserEnum
+enumerate()
UserEnumerationController
-_kwargs
+enumerate()
NetUserGetInfo
+enumerate()
UserInfoEnumerationController
-_kwargs
+enumerate()

2.34 Reporters



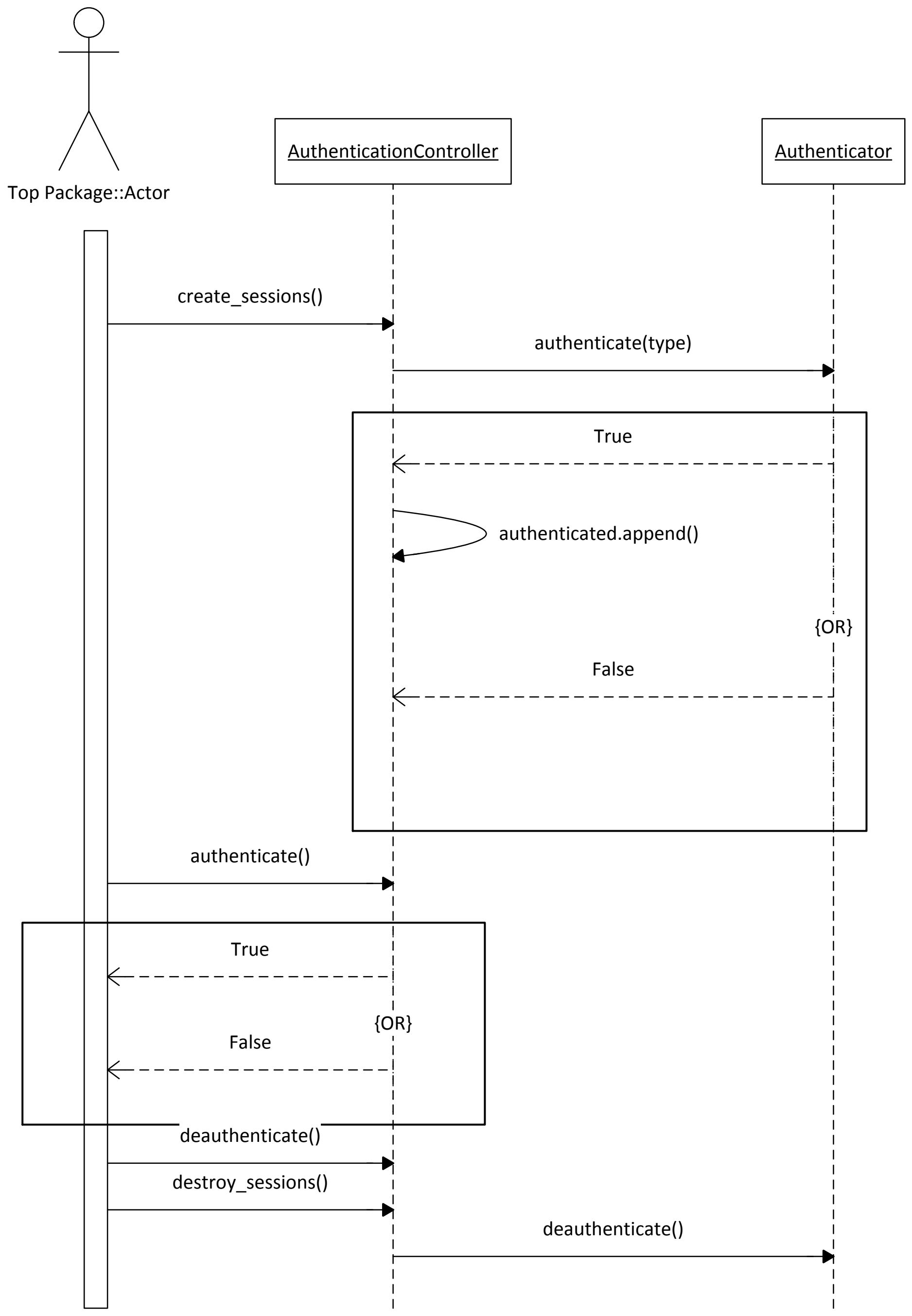
2.35 Nettynum



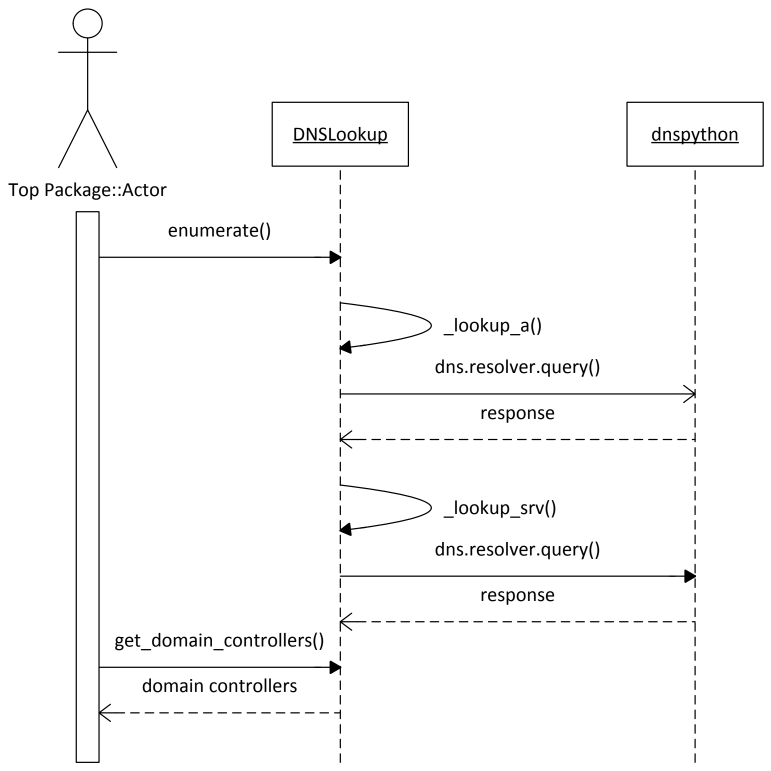
```
Nettynum
-automated_domain : <unspecified> = False
-automated_local : <unspecified> = False
-enumerate_domain_names : <unspecified> = False
-enumerate_domain_controllers : <unspecified> = False
-enumerate_domain_policies : <unspecified> = False
-enumerate_domain_groups : <unspecified> = False
-enumerate_domain_group_members : <unspecified> = False
-enumerate_domain_users : <unspecified> = False
-enumerated_domain_user : <unspecified> = False
-enumerate_domaoin_group_membership : <unspecified> = False
-enumerate_interesting_hosts : <unspecified> = False
-enumerate_local_policies : <unspecified> = False
-enumeated_local_groups : <unspecified> = False
-enumerate_local_group_members : <unspecified> = False
-enumerate_local_users : <unspecified> = False
-enumerate_local_user : <unspecified> = False
-enumerate_local_group_membership : <unspecified> = False
-enumerate_local_shares
-domain_name : <unspecified> = None
-domain_controller : <unspecified> = None
-host : <unspecified> = None
-group_name : <unspecified> = None
-user_name : <unspecified> = None
-domain : <unspecified> = None
-user : <unspecified> = None
-passwd : <unspecified> = None
-community : <unspecified> = None
-output_file : <unspecified> = None
+_automated_domain_enumeration()
+_automated_local_enumeration()
+_domain_names_enumeration()
+_domain_controllers_enumeration()
+_domain_policies_enumeration()
+_domain_groups_enumeration()
+_domain_group_members_enumeration()
+_domain_users_enumeration()
+_domain_user_enumeration()
+_domain_group_membership_enumeration()
+_interesting_hosts_enumeration()
+_local_policies_enumeration()
+_local_groups_enumeration()
+_local_group_members_enumeration()
+_local_users_enumeration()
+_local_user_enumeratoin()
+_local_group_membership_enumeration()
+_local_share_enumeration()
+run()
```

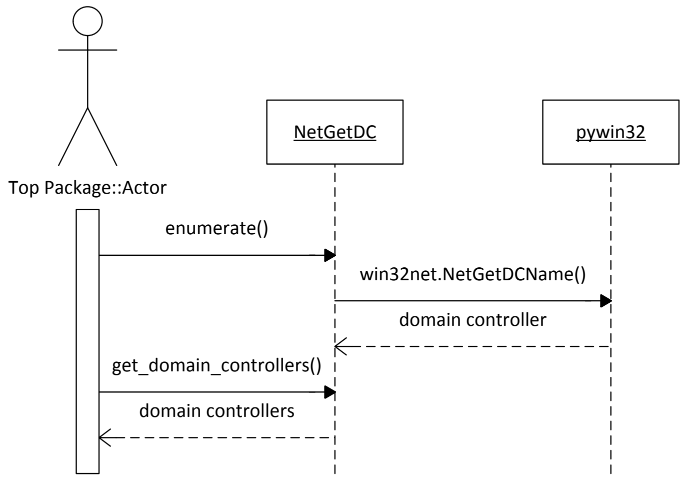
3 Sequence Diagrams

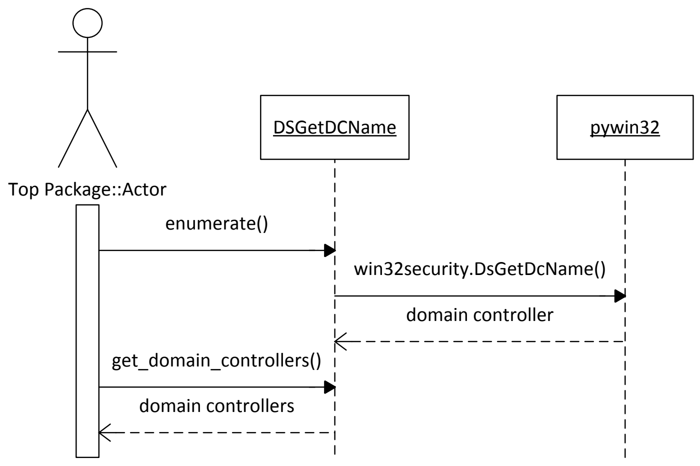
3.1 AuthenticationController



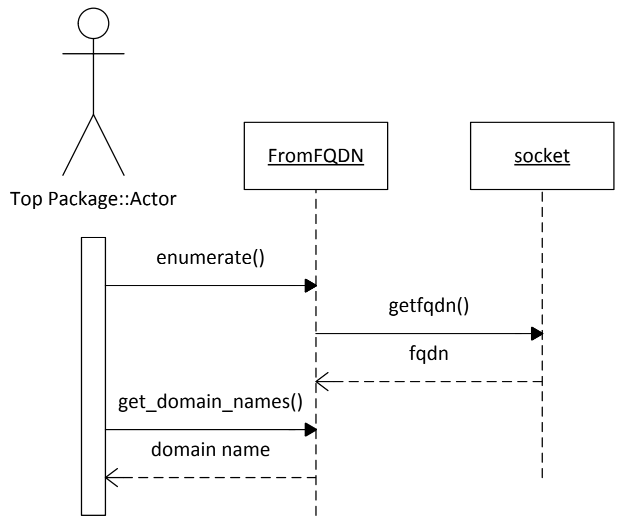
3.2 DomainControllerEnumeration

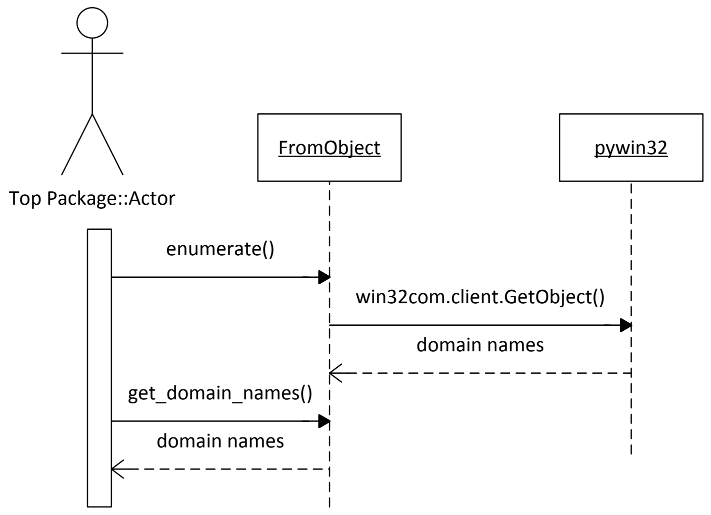


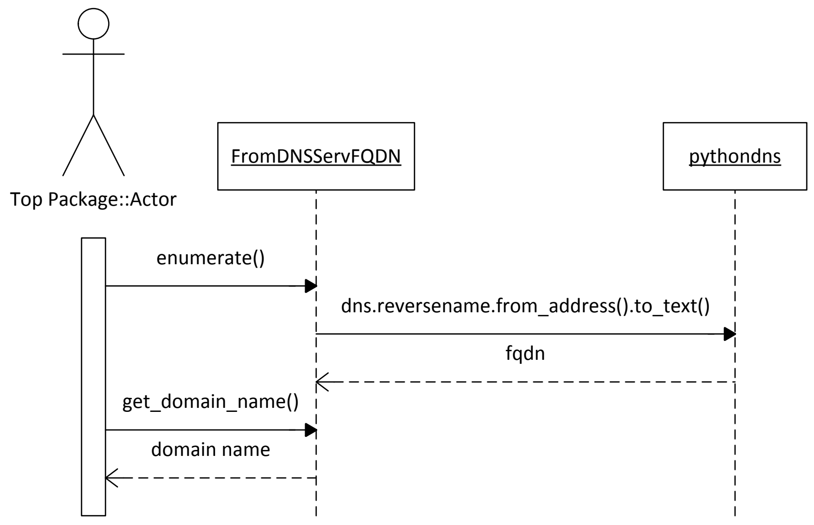




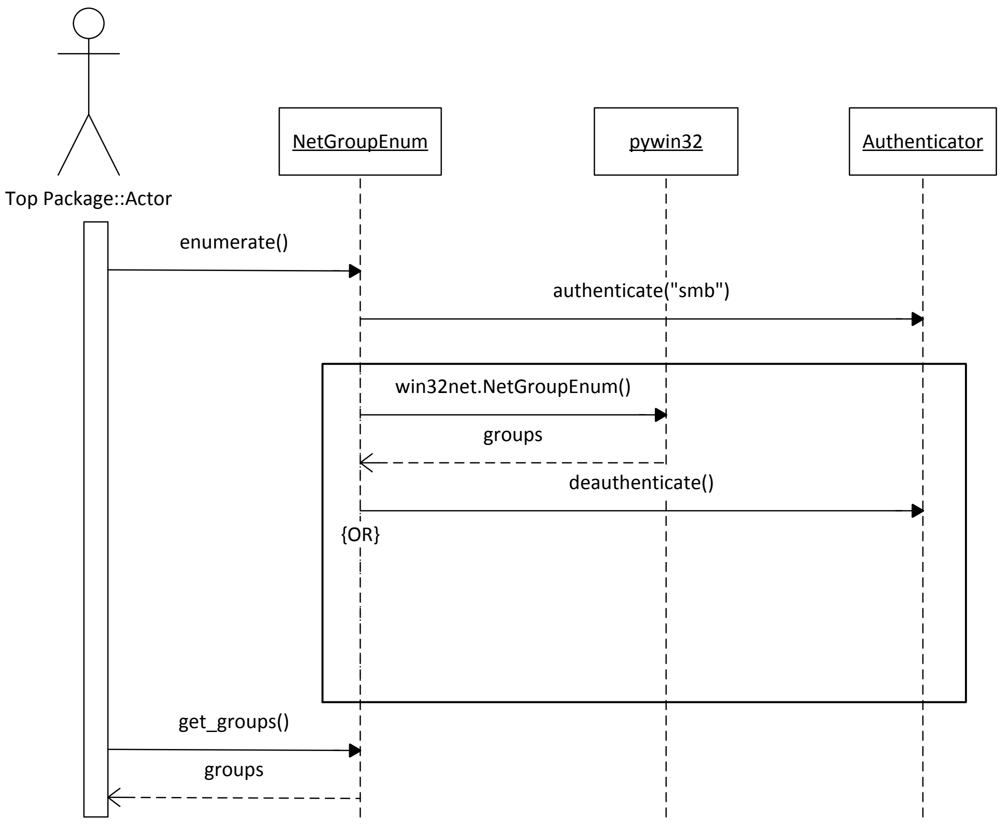
3.3 DomainNameEnumeration

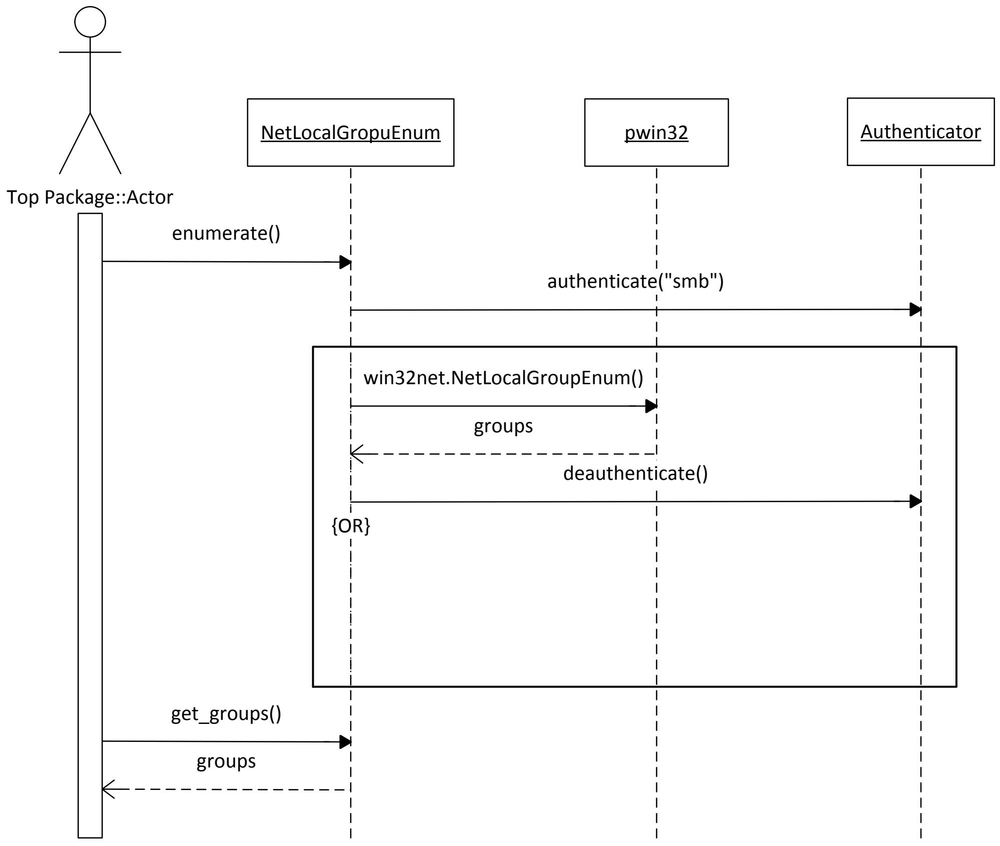




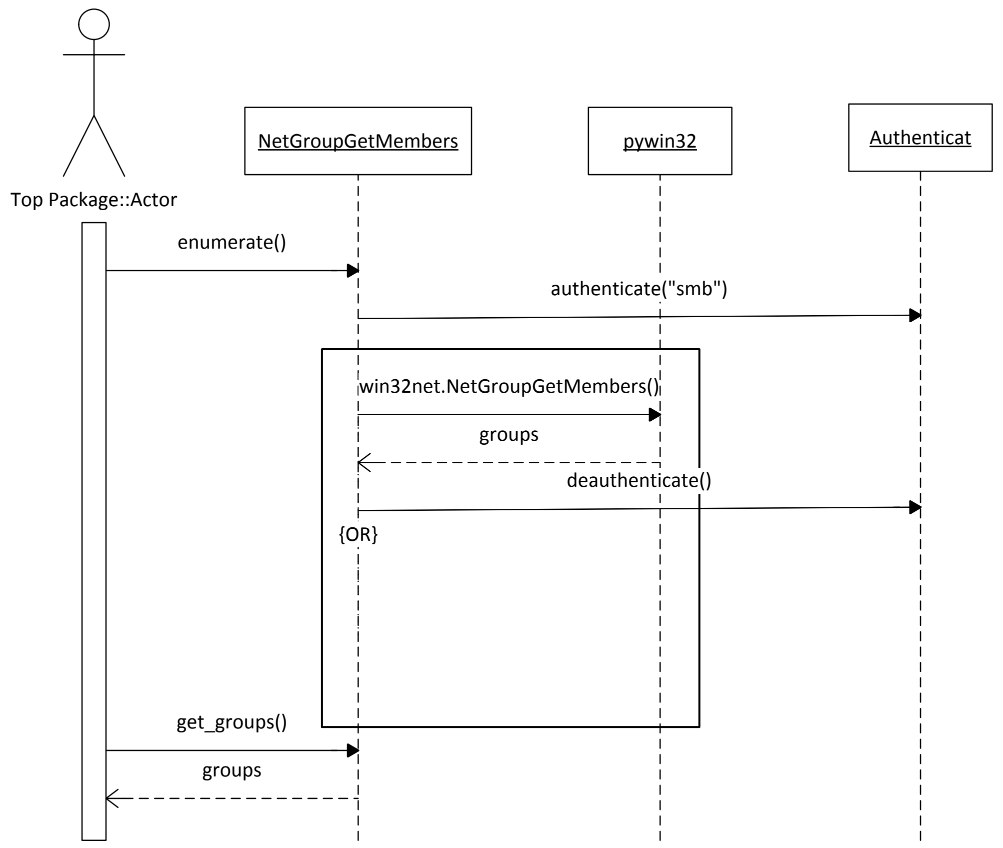


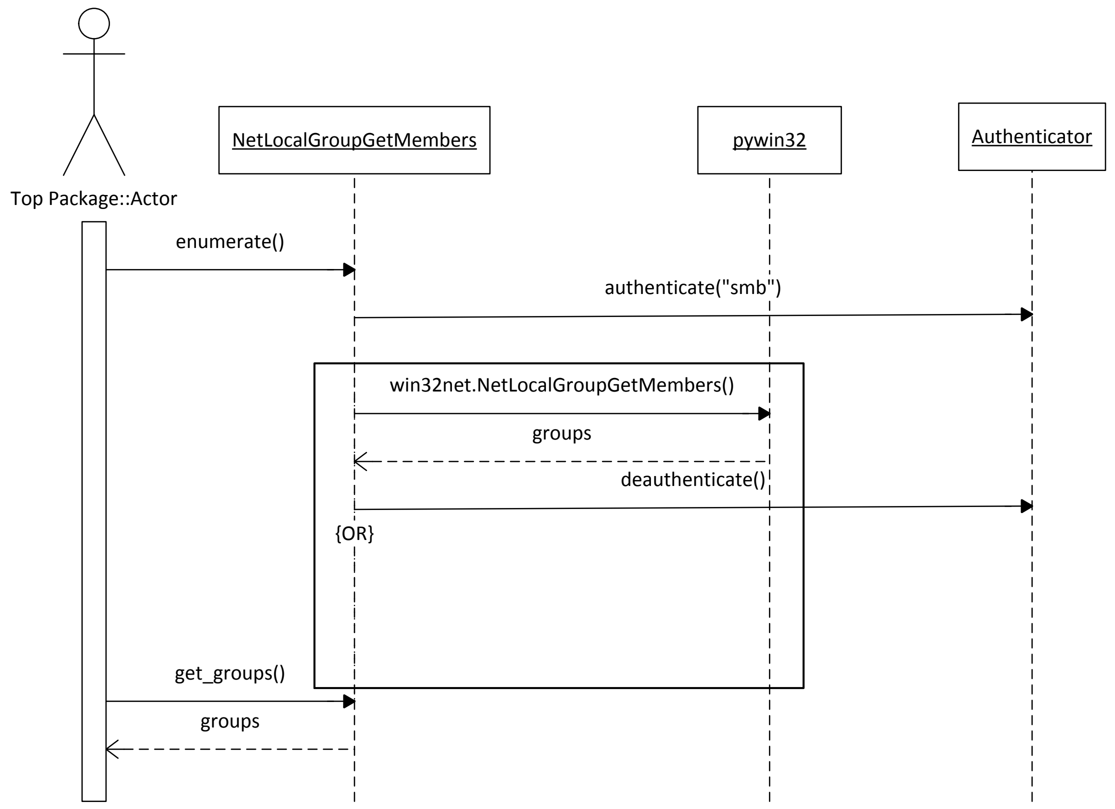
3.4 GroupEnumeration



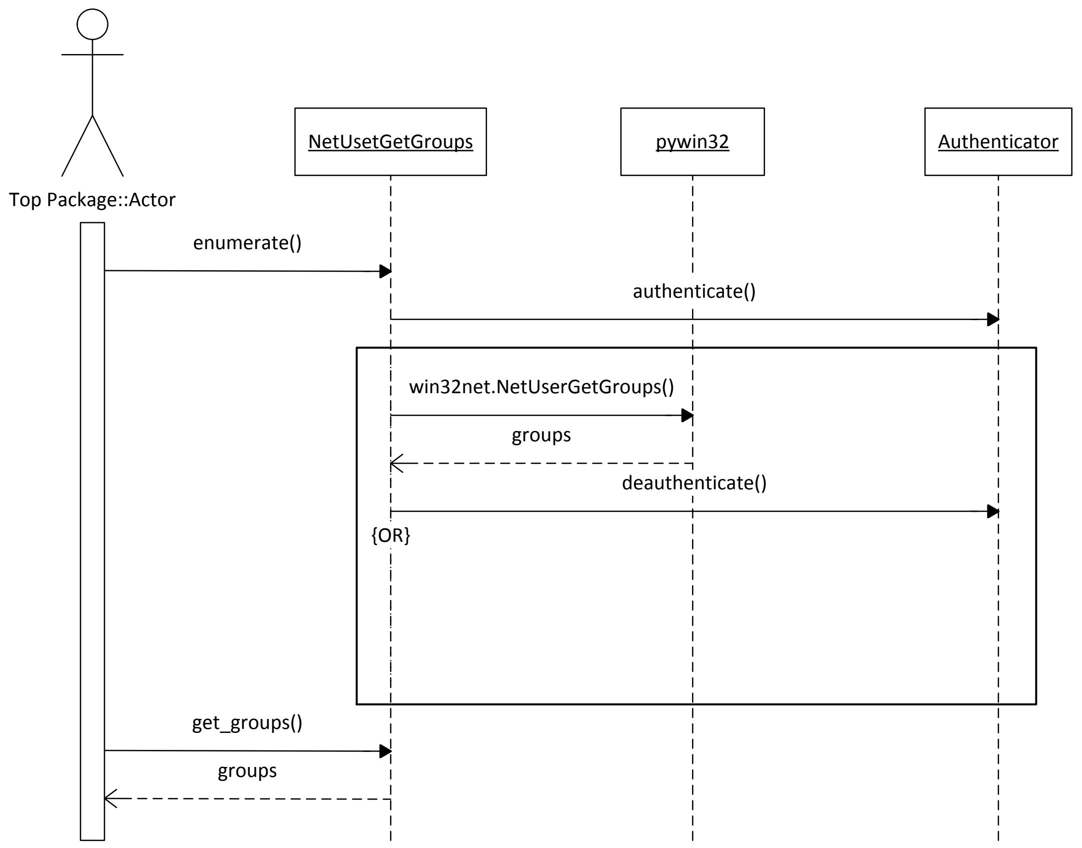


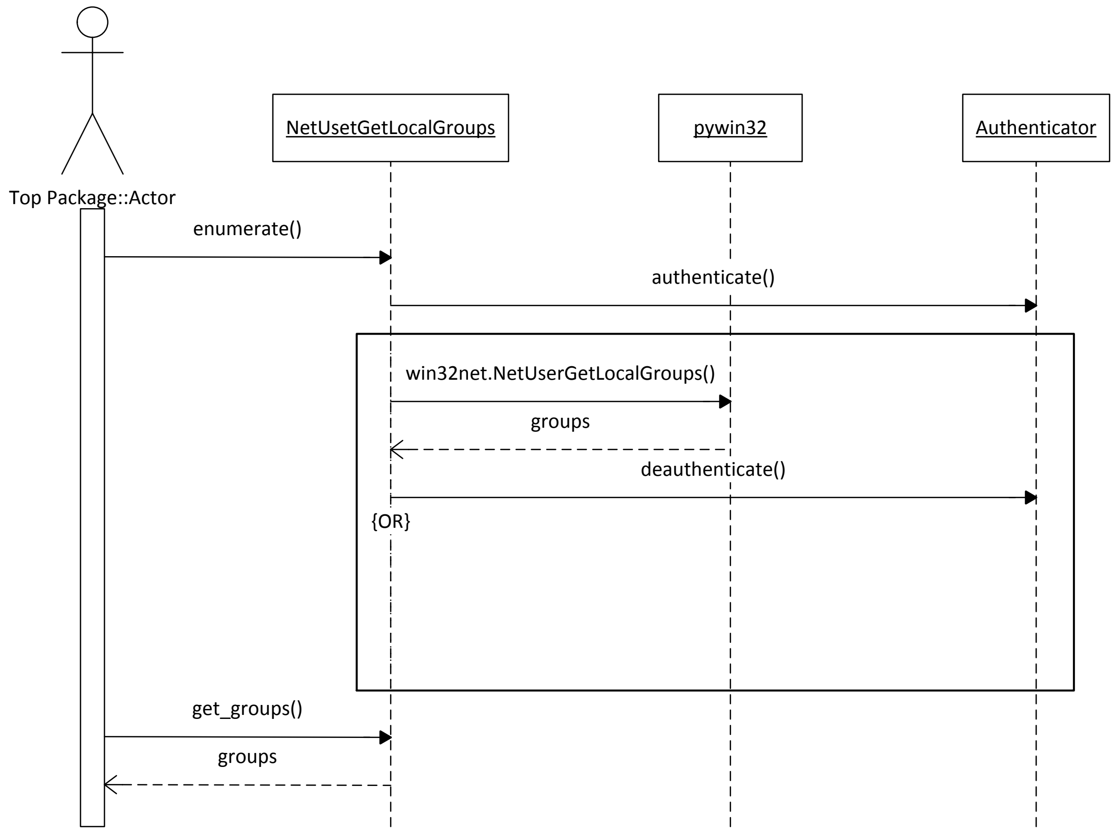
3.5 GroupMembersEnumeration



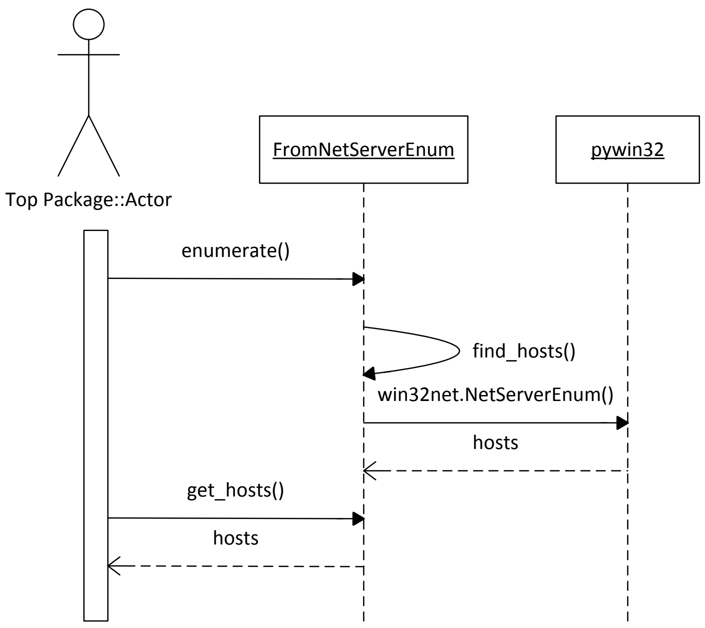


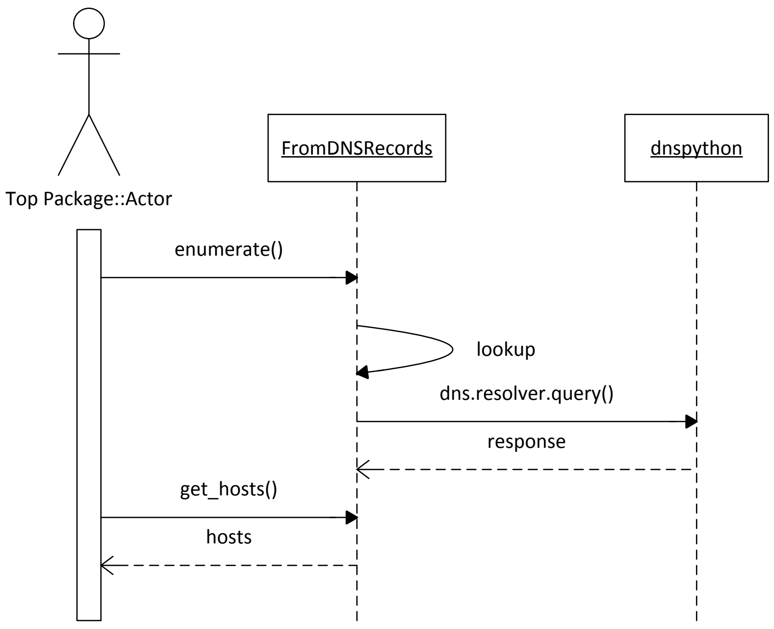
3.6 GroupMembershipEnumeration



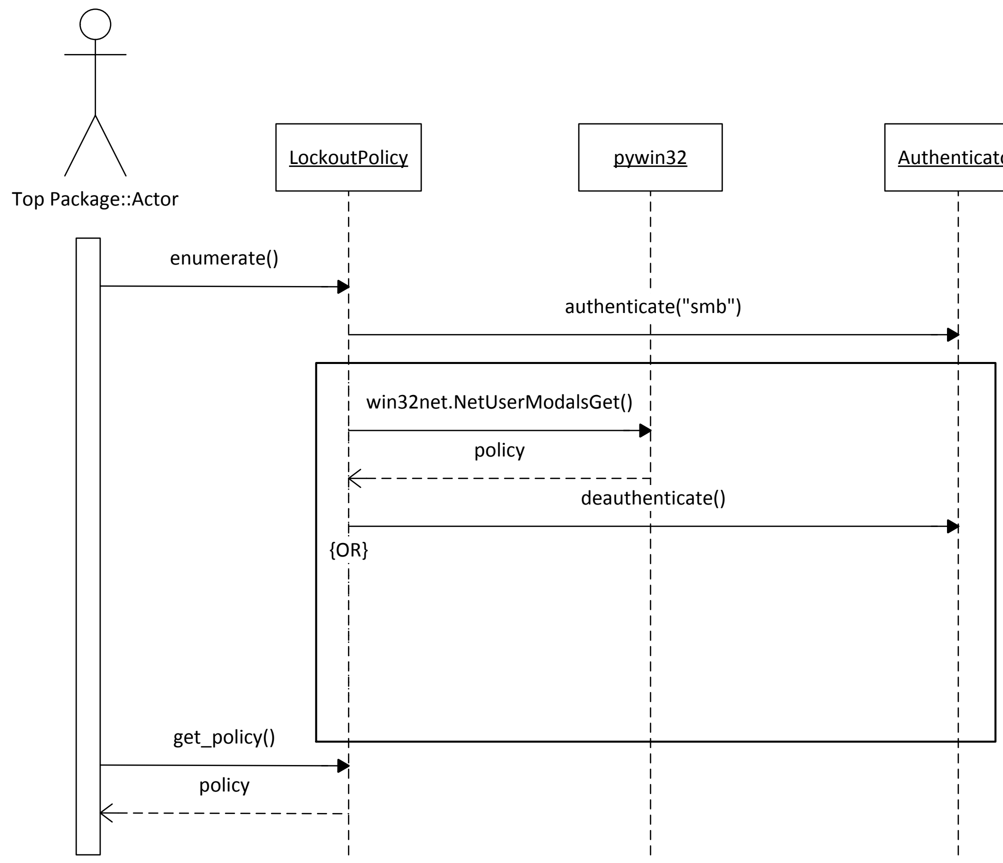


3.7 InterestingHostEnumeration

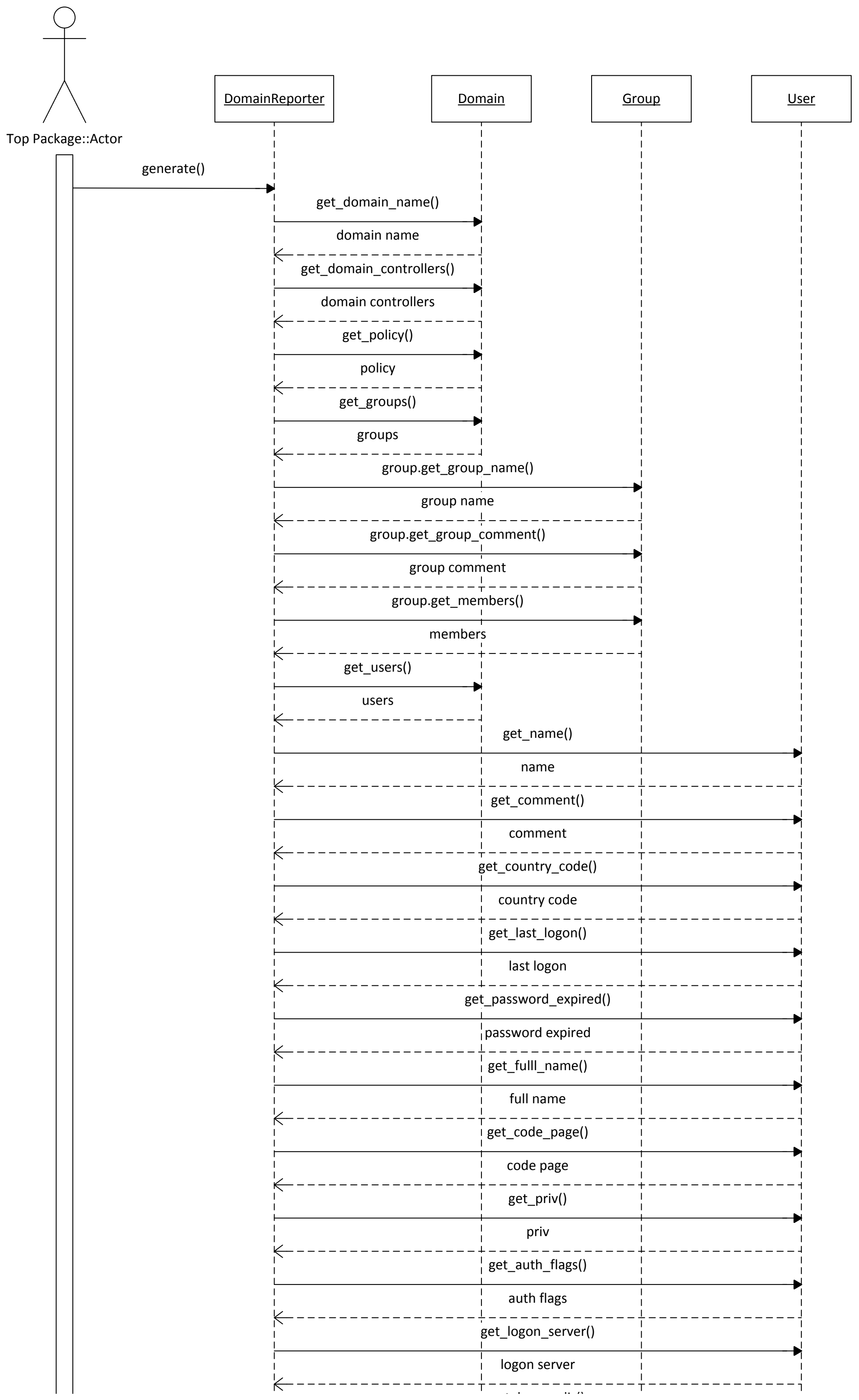


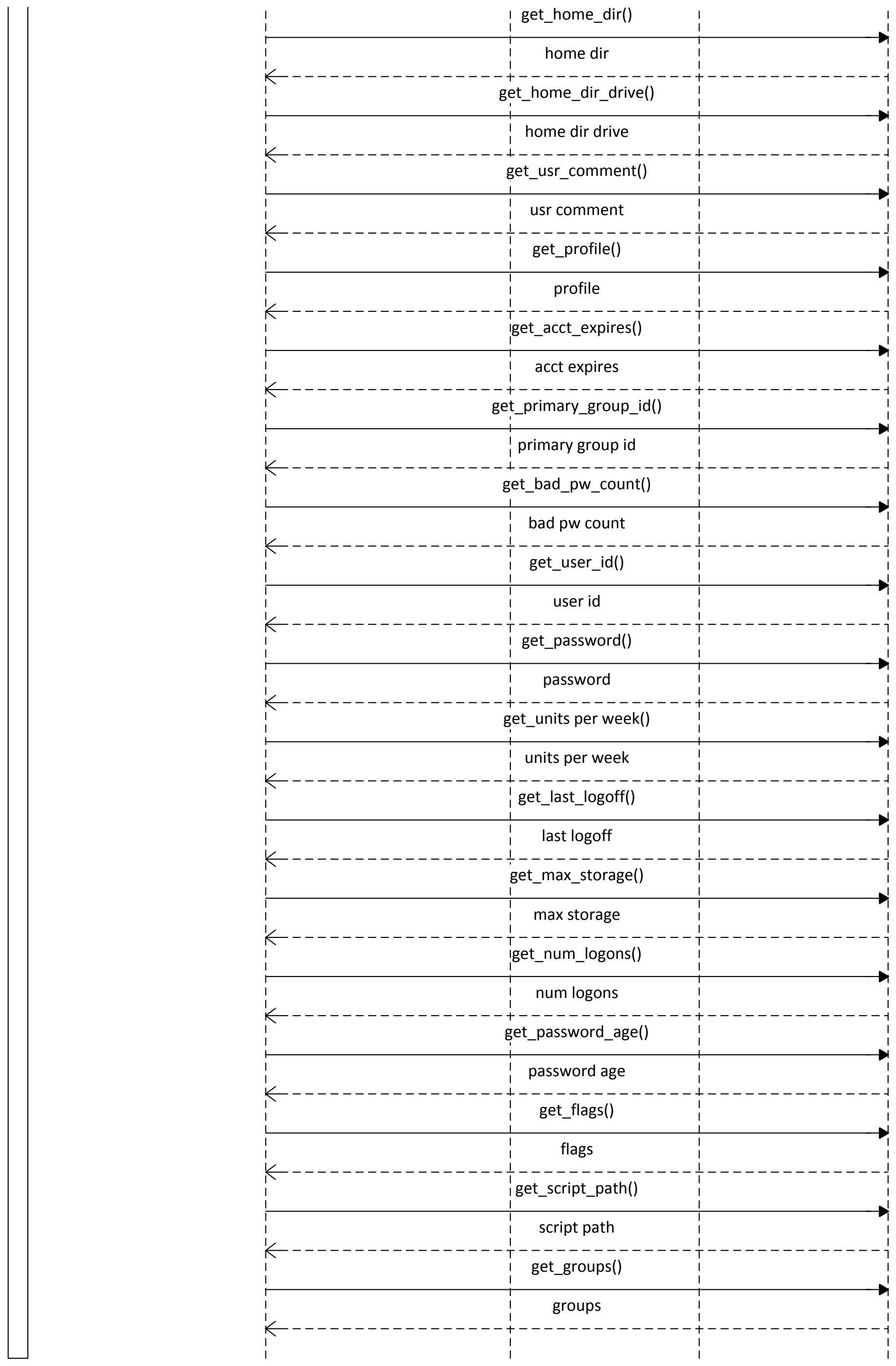


3.8 LockoutPolicyEnumeration

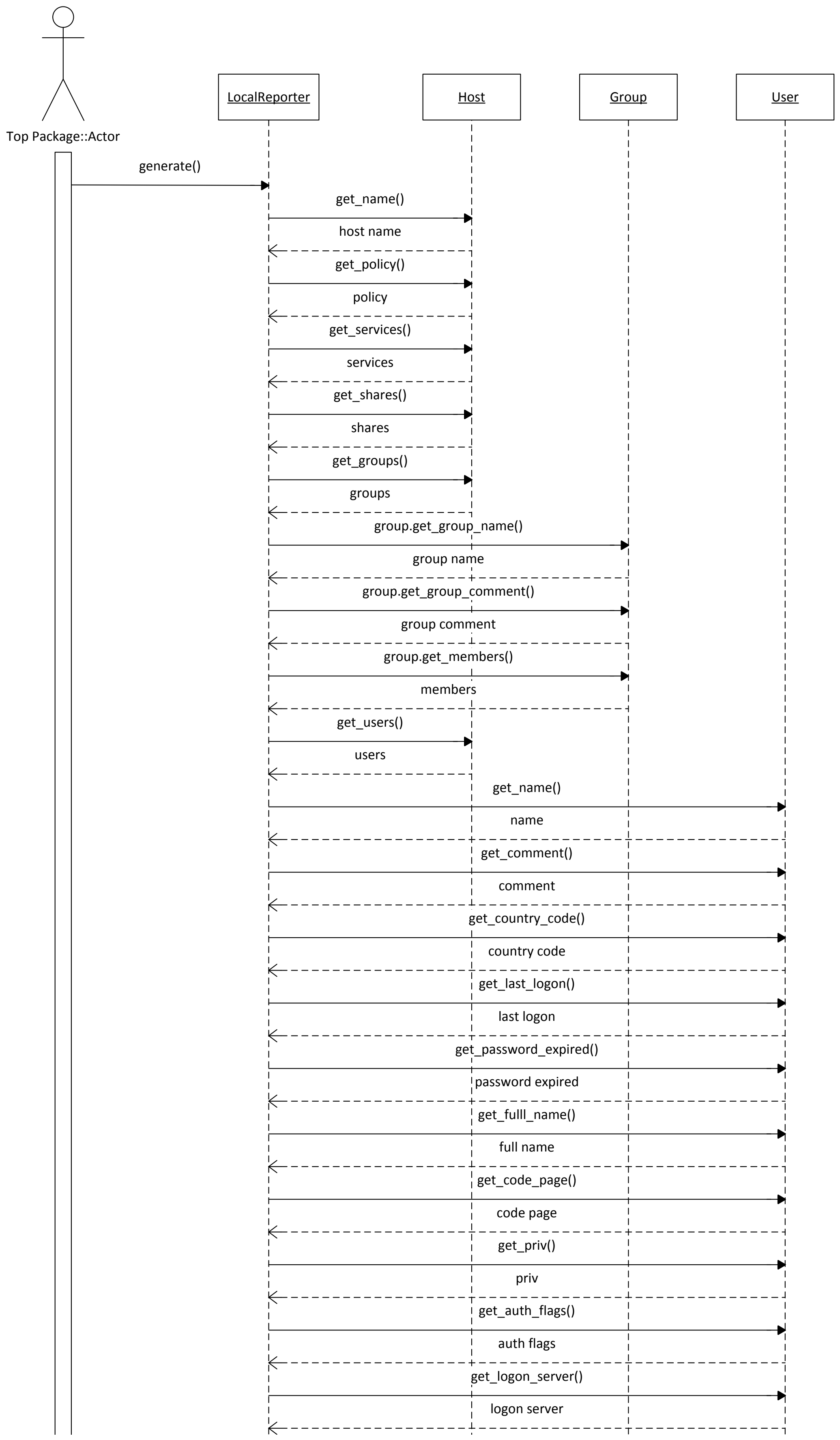


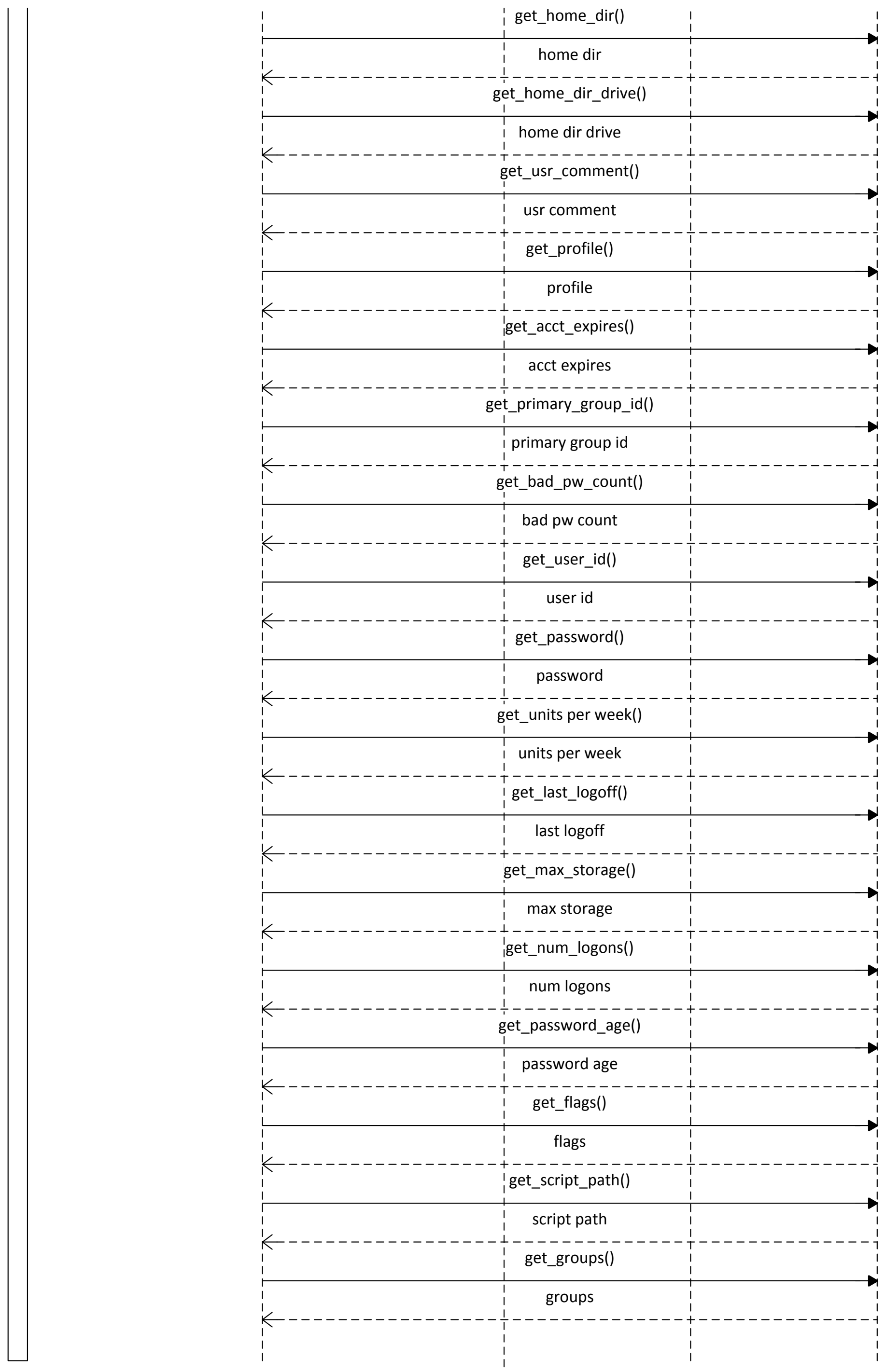
3.9 DomainReporter



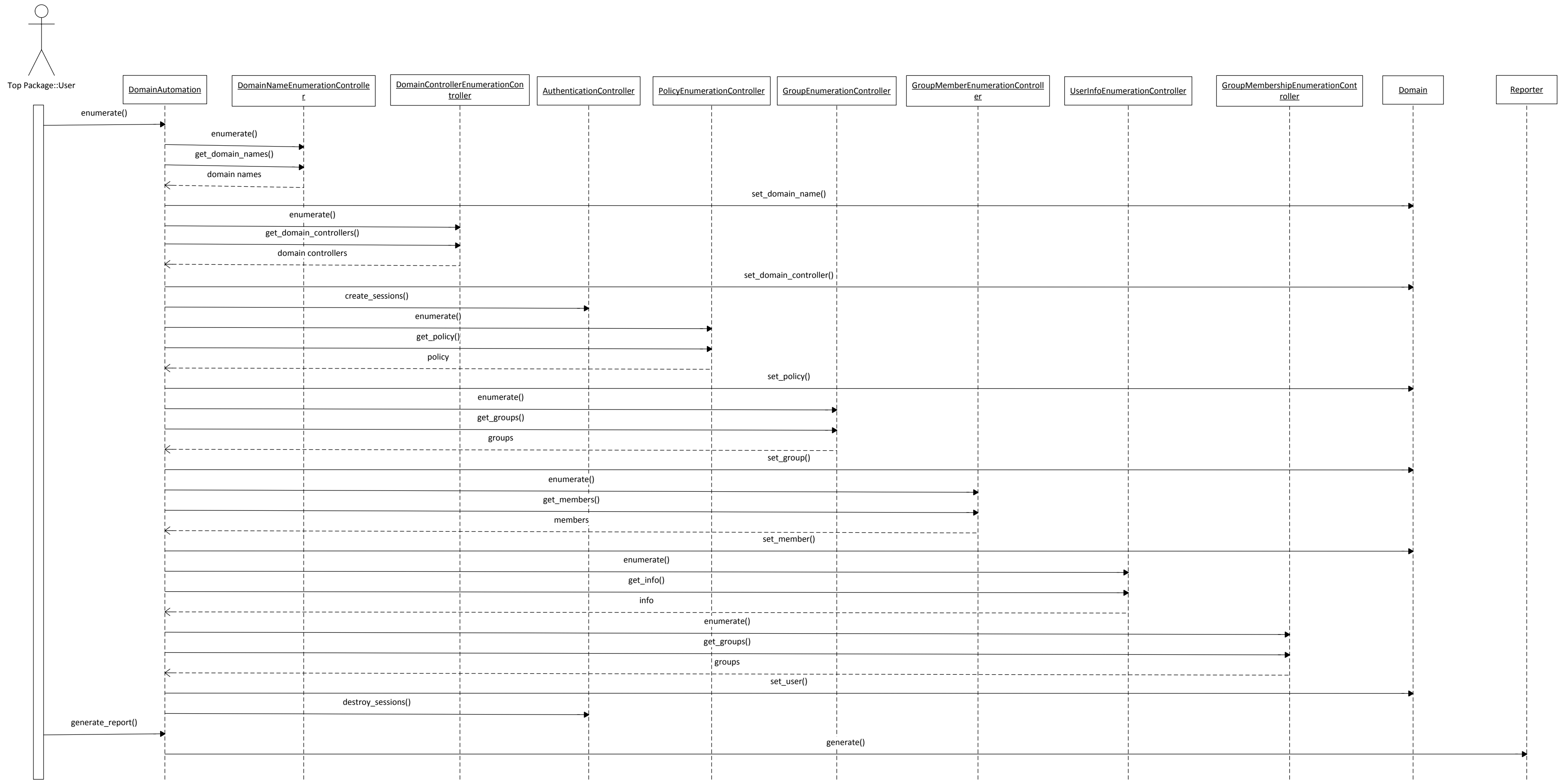

get_home_dir()
home dir
get_home_dir_drive()
home dir drive
get_usr_comment()
usr comment
get_profile()
profile
get_acct_expires()
acct expires
get_primary_group_id()
primary group id
get_bad_pw_count()
bad pw count
get_user_id()
user id
get_password()
password
get_units per week()
units per week
get_last_logoff()
last logoff
get_max_storage()
max storage
get_num_logons()
num logons
get_password_age()
password age
get_flags()
flags
get_script_path()
script path
get_groups()
groups

3.10 LocalReporter

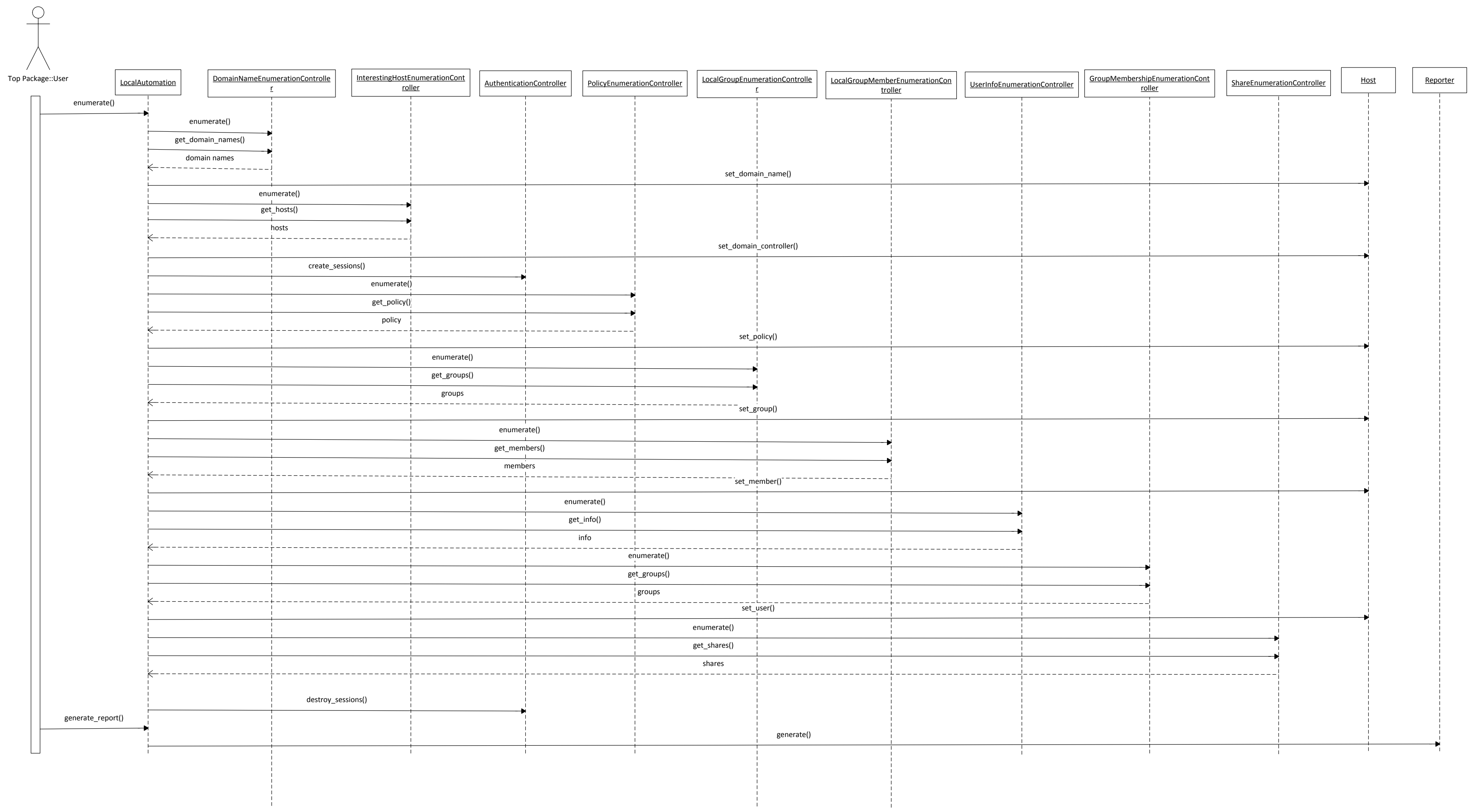



get_home_dir()
home dir
get_home_dir_drive()
home dir drive
get_usr_comment()
usr comment
get_profile()
profile
get_acct_expires()
acct expires
get_primary_group_id()
primary group id
get_bad_pw_count()
bad pw count
get_user_id()
user id
get_password()
password
get_units per week()
units per week
get_last_logoff()
last logoff
get_max_storage()
max storage
get_num_logons()
num logons
get_password_age()
password age
get_flags()
flags
get_script_path()
script path
get_groups()
groups

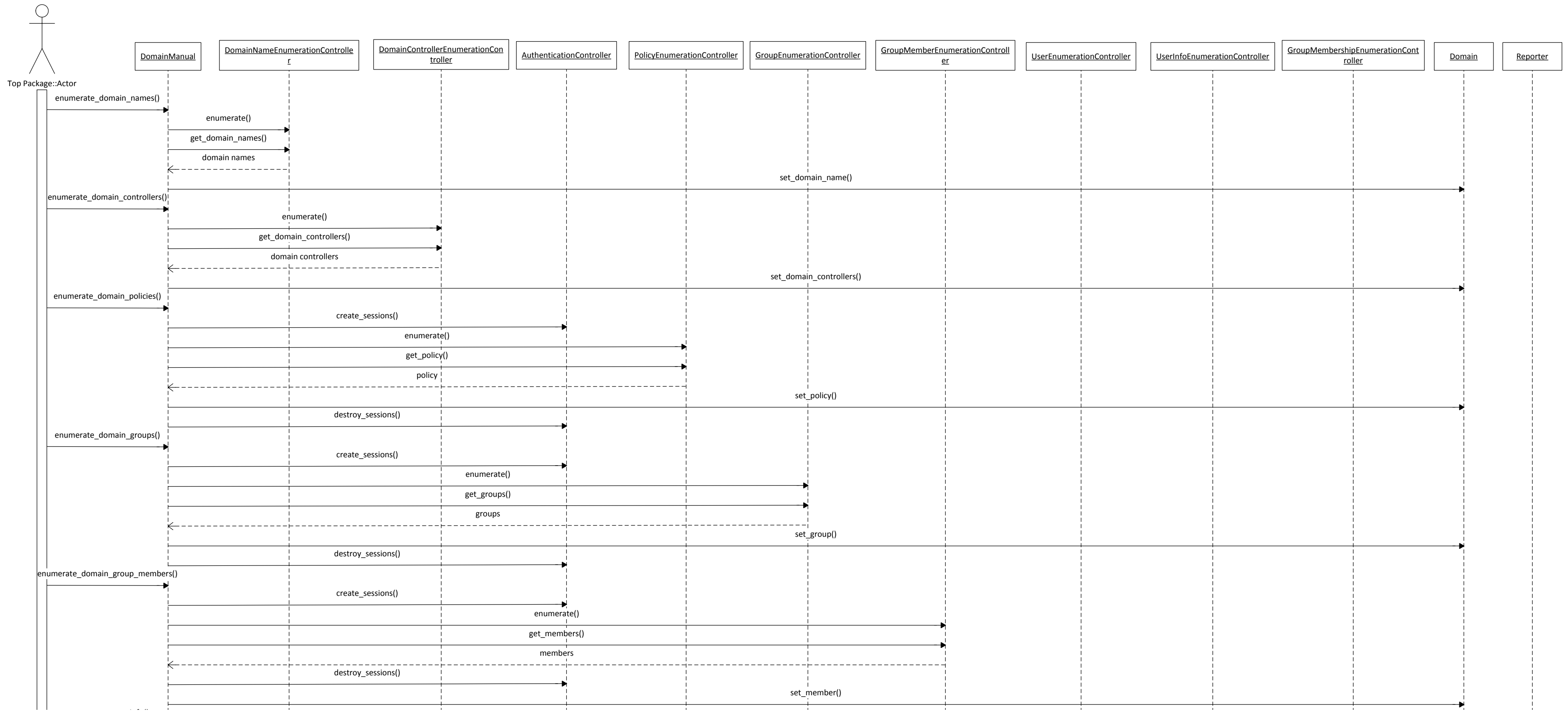
3.11 AutomatedDomainEnumeration

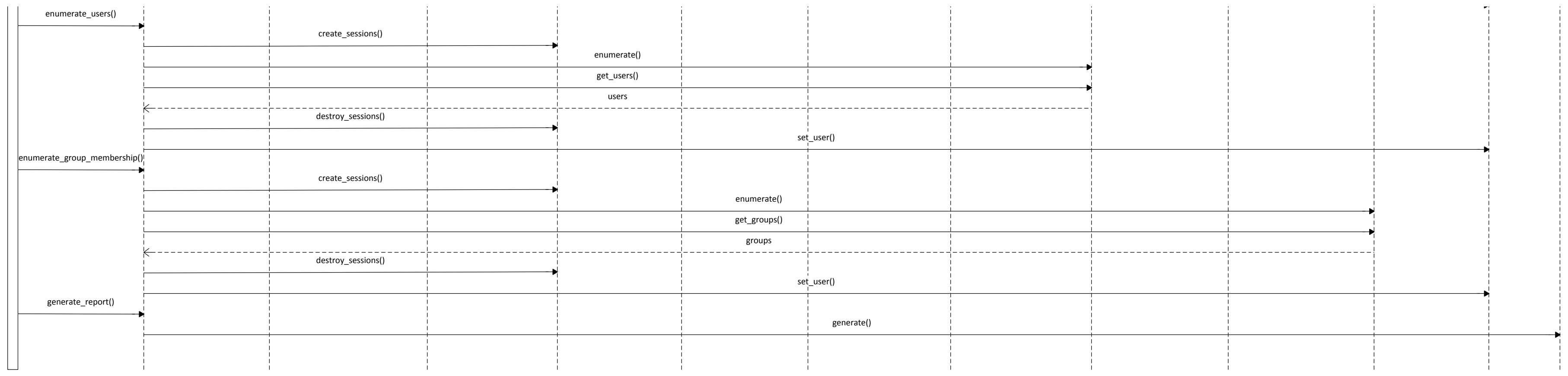


3.12 AutomatedLocalEnumeration



3.13 ManualEnumeration




enumerate_users()
create_sessions()
enumerate()
get_users()
users
destroy_sessions()
set_user()
enumerate_group_membership()
create_sessions()
enumerate()
get_groups()
groups
destroy_sessions()
set_user()
generate_report()
generate()

Appendix D: Test Bed Configuration

The testing environment was configured as follows:

Two Windows Servers (2012 and 2008 R2) with the static IP addresses 10.10.10.1 and 10.10.10.2 respectively were setup each with Active Directory, DNS, WINS DHCP installed. DHCP was configured to serve addresses in the range 10.0.0.0/8 range (excluding the server's addresses) and advertise 10.10.10.1 and 10.10.10.2 as DNS servers. WINS was configured on the 2008r2 and replicated to the 2012 server. Active Directory was setup using the domain name 'testdomain.local' and allowed to configure the DNS service appropriately. This gave these two servers the role of Domain Controller within the 'testdomain.local' domain.

The default accounts policies were left unchanged:

```
Force user logoff how long after time expires?:    Never
Minimum password age (days):                       1
Maximum password age (days):                       42
Minimum password length:                           7
Length of password history maintained:             24
Lockout threshold:                                 Never
Lockout duration (minutes):                        30
Lockout observation window (minutes):              30
```

Along with the standard domain users and groups 50 users and 50 groups were created within the Active Directory. Each of the 50 groups were assigned 10 users as group members. This resulted in a total of 61 groups with a varied number of group members and a total of 52 user accounts and 2 machine accounts in the domain. The table below show the group membership within active directory. The "Domain Users" group contains all user accounts within the domain.

Domain Groups		
Group Name	**Group Comment**	**Group Members**
DnsUpdateProxy	DNS clients who are permitted to perform dynamic updates on behalf of some other clients (such as DHCP servers).	
Domain Admins	Designated administrators of the domain	Administrator
Domain Computers	All workstations and servers joined to the domain	
Domain Controllers	All domain controllers in the domain	WIN-52VR2HTADOK$, WIN-CKU3D6EPQKA$
Domain Guests	All domain guests	

Domain Users	All domain users	Administrator, krbtgt, user1, user2, user3, user4, user5, user6, user7, user8, user9, user10, user11, user12, user13, user14, user15, user16, user17, user18, user19, user20, user21, user22, user23, user24, user25, user26, user27, user28, user29, user30, user31, user32, user33, user34, user35, user36, user37, user38, user39, user40, user41, user42, user43, user44, user45, user46, user47, user48, user49, user50
Enterprise Admins	Designated administrators of the enterprise	Administrator
Enterprise Read-only Domain Controllers	Members of this group are Read-Only Domain Controllers in the enterprise	
Group Policy Creator Owners	Members in this group can modify group policy for the domain	Administrator
Read-only Domain Controllers	Members of this group are read-only domain controllers in the domain	
Schema Admins	Designated administrators of the schema	Administrator
group1	test group 1	user1, user2, user3, user4, user5, user6, user7, user8, user9, user10
group2	test group 2	user2, user3, user4, user5, user6, user7, user8, user9, user10, user11
group3	test group 3	user3, user4, user5, user6, user7, user8, user9, user10, user11, user12
group4	test group 4	user4, user5, user6, user7, user8, user9, user10,

		user11, user12, user13
group5	test group 5	user5, user6, user7, user8, user9, user10, user11, user12, user13, user14
group6	test group 6	user6, user7, user8, user9, user10, user11, user12, user13, user14, user15
group7	test group 7	user7, user8, user9, user10, user11, user12, user13, user14, user15, user16
group8	test group 8	user8, user9, user10, user11, user12, user13, user14, user15, user16, user17
group9	test group 9	user9, user10, user11, user12, user13, user14, user15, user16, user17, user18
group10	test group 10	user10, user11, user12, user13, user14, user15, user16, user17, user18, user19
group11	test group 11	user11, user12, user13, user14, user15, user16, user17, user18, user19, user20
group12	test group 12	user12, user13, user14, user15, user16, user17, user18, user19, user20, user21
group13	test group 13	user13, user14, user15, user16, user17, user18, user19, user20, user21, user22
group14	test group 14	user14, user15, user16, user17, user18, user19, user20, user21, user22, user23
group15	test group 15	user15, user16, user17, user18, user19, user20,

		user21, user22, user23, user24
group16	test group 16	user16, user17, user18, user19, user20, user21, user22, user23, user24, user25
group17	test group 17	user17, user18, user19, user20, user21, user22, user23, user24, user25, user26
group18	test group 18	user18, user19, user20, user21, user22, user23, user24, user25, user26, user27
group19	test group 19	user19, user20, user21, user22, user23, user24, user25, user26, user27, user28
group20	test group 20	user20, user21, user22, user23, user24, user25, user26, user27, user28, user29
group21	test group 21	user21, user22, user23, user24, user25, user26, user27, user28, user29, user30
group22	test group 22	user22, user23, user24, user25, user26, user27, user28, user29, user30, user31
group23	test group 23	user23, user24, user25, user26, user27, user28, user29, user30, user31, user32
group24	test group 24	user24, user25, user26, user27, user28, user29, user30, user31, user32, user33
group25	test group 25	user25, user26, user27, user28, user29, user30, user31, user32, user33,

		user34
group26	test group 26	user26, user27, user28, user29, user30, user31, user32, user33, user34, user35
group27	test group 27	user27, user28, user29, user30, user31, user32, user33, user34, user35, user36
group28	test group 28	user28, user29, user30, user31, user32, user33, user34, user35, user36, user37
group29	test group 29	user29, user30, user31, user32, user33, user34, user35, user36, user37, user38
group30	test group 30	user30, user31, user32, user33, user34, user35, user36, user37, user38, user39
group31	test group 31	user31, user32, user33, user34, user35, user36, user37, user38, user39, user40
group32	test group 32	user32, user33, user34, user35, user36, user37, user38, user39, user40, user41
group33	test group 33	user33, user34, user35, user36, user37, user38, user39, user40, user41, user42
group34	test group 34	user34, user35, user36, user37, user38, user39, user40, user41, user42, user43
group35	test group 35	user35, user36, user37, user38, user39, user40, user41, user42, user43, user44

group36	test group 36	user36, user37, user38, user39, user40, user41, user42, user43, user44, user45
group37	test group 37	user37, user38, user39, user40, user41, user42, user43, user44, user45, user46
group38	test group 38	user38, user39, user40, user41, user42, user43, user44, user45, user46, user47
group39	test group 39	user39, user40, user41, user42, user43, user44, user45, user46, user47, user48
group40	test group 40	user40, user41, user42, user43, user44, user45, user46, user47, user48, user49
group41	test group 41	user41, user42, user43, user44, user45, user46, user47, user48, user49, user50
group42	test group 42	user42, user43, user44, user45, user46, user47, user48, user49, user50, user1
group43	test group 43	user43, user44, user45, user46, user47, user48, user49, user50, user1, user2
group44	test group 44	user44, user45, user46, user47, user48, user49, user50, user1, user2, user3
group45	test group 45	user45, user46, user47, user48, user49, user50, user1, user2, user3, user4
group46	test group 46	user46, user47, user48, user49, user50, user1,

		user2, user3, user4, user5
group47	test group 47	user47, user48, user49, user50, user1, user2, user3, user4, user5, user6
group48	test group 48	user48, user49, user50, user1, user2, user3, user4, user5, user6, user7
group49	test group 49	user49, user50, user1, user2, user3, user4, user5, user6, user7, user8
group50	test group 50	user50, user1, user2, user3, user4, user5, user6, user7, user8, user9

The server 10.10.10.1 was used for local enumeration with the default users and groups. Note, since this host is a domain controller the 50 users that were created also exist as local users. The table below shows the groups and membership that exist.

Group Name	Group Comment	Group Members
Account Operators	Members can administer domain user and group accounts	
Administrators	Administrators have complete and unrestricted access to the computer/domain	Administrator, Domain Admins, Enterprise Admins
Allowed RODC Password Replication Group	Members in this group can have their passwords replicated to all read-only domain controllers in the domain	
Backup Operators	Backup Operators can override security restrictions for the sole purpose of backing up and restoring files	
Cert Publishers	Members of this group are permitted to publish certificates to the directory	
Certificate Service DCOM Access	Members of this group are allowed to connect to Certification Authorities in the enterprise	

Cryptographic Operators	Members are authorized to perform cryptographic operations	
Denied RODC Password Replication Group	Members in this group cannot have their passwords replicated to any read-only domain controllers in the domain	Cert Publishers, Domain Admins, Domain Controllers, Enterprise Admins, Group Policy Creator Owners, krbtgt, Read-only Domain Controllers, Schema Admins
DHCP Administrators	Members who have administrative access to the DHCP Service	
DHCP Users	Members who have view-only access to the DHCP service	
Distributed COM Users	Members are allowed to launch, activate and use Distributed COM objects on this machine.	
DnsAdmins	DNS Administrators Group	
Event Log Readers	Members of this group can read event logs from local machine	
Guests	Guests have the same access as members of the Users group by default, except for the Guest account which is further restricted	Domain Guests, Guest
IIS_IUSRS	Built-in group used by Internet Information Services	ISUR
Incoming Forest Trust Builders	Members of this group can create incoming, on-way trusts to this forest	
Network Configuration Operators	Members in this group can have some administrative privileges to manage configuration of networking features	
Performance Log Users	Members of this group may schedule logging of performance counters, enable trace providers, and collect event traces both locally and via	

	remote access to this computer	
Performance Monitor Users	Members of this group can access performance counter data locally and remotely	
Pre-Windows 2000 Compatible Access	A backward compatibility group which allows read access on all users and groups in the domain	Authenticated Users
Print Operators	Members can administer domain printers	
RAS and IAS Servers	Servers in this group can access remote access properties of users	
Remote Desktop Users	Members in this group are granted the right to logon remotely	
Replicator	Supports file replication in a domain	
Server Operators	Members can administer domain servers	
Terminal Server License Servers	Members of this group can update user accounts in Active Directory with information about license issuance, for the purpose of tracking and reporting TS Per User CAL usage	
Users	Users are prevented from making accidental or intentional system-wide changes and can run most applications	Domain Users, Authenticated Users, INTERACTIVE
Windows Authorization Access Group	Members of this group have access to the computed tokenGroupsGlobalAndUniversal attribute on User objects	ENTERPRISE DOMAIN CONTROLLERS

Appendix E: Testing Results

1. Unit Testing

1.1. Authenticators

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.1.1.	CredsSMBAuthenticaitonTestCase				P
1.1.1.1.	test_valid Create SMBAuthenticator with valid credentials and call authenticate method.	Authenticate returns True	Authenticate returns True		P
1.1.1.2.	test_invalid_password Create SMBAuthenticator with valid credentials except password and call authenticate method.	Authenticate returns False	Authenticate returns False		P
1.1.1.3.	test_invalid_user Create SMBAuthenticator with valid credentials except user name and call authenticate method.	Authenticate returns False	Authenticate returns False		P
1.1.2.	InvalidSMBAuthenticationTestCase				P
1.1.2.1.	test_invalid_host Create SMBAuthenticator with valid details except host	Exception raised.	Exception raised.		P
1.1.2.2.	test_invalid_share Create SMBAuthenticator with valid details except share	Exception raised.	Exception raised.		P
1.1.2.3.	test_invalid_user Create SMBAuthenticator with valid details except user name	Exception raised.	Exception raised.		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.1.2.4.	test_invalid_passwd Create SMBAuthenticator with valid details except password	Exception raised.	Exception raised.		P
1.1.2.5.	test_invalid_domain Create SMBAuthenticator with valid details except domain	Exception raised.	Exception raised.		P
1.1.3.	SMBDeauthenticationTestCase				P
1.1.3.1.	test_valid Call deauthenticate method of SMBAuthenticator	No exception raised	No exception raised		P
1.1.4.	CredsSNMPAuthenticationTestCase				P
1.1.4.1.	test_valid Create SMBAuthenticator with valid credentials and call authenticate method.	Authenticate returns True	Authenticate returns True		P
1.1.4.2.	test_invalid_community_string Create SMBAuthenticator with valid credentials except community string and call authenticate method.	Authenticate returns False	Authenticate returns False		P
1.1.5.	InvalidSNMPAuthenticationTestCase				P
1.1.5.1.	test_invalid_host Create SMBAuthenticator with valid details except host	Exception raised.	Exception raised.		P
1.1.5.2.	test_invalid_community_string Create SMBAuthenticator with valid details except community string	Exception raised.	Exception raised.		P

1.2. Enumerators

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.1.	LocalGroupSMBEnumerationTestCase				P
1.2.1.1.	test_valid Create NetLocalGroupEnum with valid details and call enumerate method.	Return non-empty list.	Return non-empty list.		P
1.2.1.2.	test_invalid_auth Create NetLocalGroupEnum with valid details except auth	Exception raised	Exception raised		P
1.2.1.3.	test_invalid_host Create NetLocalGroupEnum with valid details except host	Exception raised	Exception raised		P
1.2.2.	LocalGroupMembersSMBEnumerationTestCase				P
1.2.2.1.	test_valid Create NetLocalGroupGetMembers with valid details and call enumerate method.	Return non-empty list	Return non-empty list		P
1.2.2.2.	test_invalid_auth Create NetLocalGroupGetMembers with valid details except auth.	Exception raised	Exception raised		P
1.2.2.3.	test_invalid_host Create NetLocalGroupGetMembers with valid details except host.	Exception raised	Exception raised		P
1.2.2.4.	test_invalid_group Create NetLocalGroupGetMembers with valid details except group.	Exception raised	Exception raised		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.2.5.	test_nonexistant_group Create NetLocalGroupGetMembers with valid details except non-existant group name supplied, and call enumerate method.	Empty list returned.	Empty list returned.		P
1.2.3.	LocalUsersSMBEnumerationTestCase				P
1.2.3.1.	test_valid Create NetUserEnum with valid details and call enumerate method.	Return non-empty list	Return non-empty list		P
1.2.3.2.	test_invalid_auth Create NetUserEnum with valid details except auth.	Exception raised	Exception raised		P
1.2.3.3.	test_invalid_host Create NetUserEnum with valid details except host.	Exception raised	Exception raised		P
1.2.4.	LocalUserInfoSMBEnumerationTestCase				P
1.2.4.1.	test_valid Create NetUserGetInfo with valid details and call enumerate method.	Return non-empty dictionary	Return non-empty dictionary		P
1.2.4.2.	test_invalid_auth Create NetUserGetInfo with valid details except auth.	Exception raised	Exception raised		P
1.2.4.3.	test_invalid_host Create NetUserGetInfo with valid details except host.	Exception raised	Exception raised		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.4.4.	test_invalid_user Create NetUserGetInfo with valid details except user.	Exception raised	Exception raised		P
1.2.4.5.	test_nonexistant_user Create NetUserEnum with valid details except non-existent user specified and call enumerate method.	Empty dictionary returned.	Empty dictionary returned.		P
1.2.5.	LocalGroupMembershipSMBEnumerationTestCase				P
1.2.5.1.	test_valid Create NetUserGetLocalGroups with valid details and call enumerate method.	Return non-empty list	Return non-empty list		P
1.2.5.2.	test_invalid_auth Create NetUserGetLocalGroups with valid details except auth.	Exception raised	Exception raised		P
1.2.5.3.	test_invalid_host Create NetUserGetLocalGroups with valid details except host.	Exception raised	Exception raised		P
1.2.5.4.	test_invalid_user Create NetUserGetLocalGroups with valid details except user.	Exception raised	Exception raised		P
1.2.5.5.	test_nonexistant_user Create NetUserGetLocalGroups with valid details except non-existent user specified and call enumerate method.	Empty list returned.	Empty list returned.		P
1.2.6.	LocalPasswordPolicySMBEnumerationTestCase				P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.6.1.	test_valid Create PasswordPolicy with valid details and call enumerate method.	Return dictionary not equal to default value.	Return dictionary not equal to default value.		P
1.2.6.2.	test_invalid_auth Create PasswordPolicy with valid details except auth.	Exception raised	Exception raised		P
1.2.6.3.	test_invalid_host Create PasswordPolicy with valid details except host.	Exception raised	Exception raised		P
1.2.7.	LocalLockoutPolicySMBEnumerationTestCase				P
1.2.7.1.	test_valid Create LockoutPolicy with valid details and call enumerate method.	Return dictionary not equal to default value.	Return dictionary not equal to default value.		P
1.2.7.2.	test_invalid_auth Create LockoutPolicy with valid details except auth.	Exception raised	Exception raised		P
1.2.7.3.	test_invalid_host Create LockoutPolicy with valid details except host.	Exception raised	Exception raised		P
1.2.8.	DomainGroupSMBEnumerationTestCase				P
1.2.8.1.	test_valid Create NetGroupEnum with valid details and call enumerate method.	Return non-empty list.	Return non-empty list.		P
1.2.8.2.	test_invalid_auth Create NetGroupEnum with valid details except auth.	Exception raised	Exception raised		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.8.3.	test_invalid_host Create NetGroupEnum with valid details except host.	Exception raised	Exception raised		P
1.2.9.	DomainGroupMembersSMBEnumerationTestCase				P
1.2.9.1.	test_valid Create NetGroupGetUsers with valid details and call enumerate method.	Return non-empty list.	Return non-empty list.		P
1.2.9.2.	test_invalid_auth Create NetGroupGetUsers with valid details except auth.	Exception raised	Exception raised		P
1.2.9.3.	test_invalid_host Create NetGroupGetUsers with valid details except host.	Exception raised	Exception raised		P
1.2.9.4.	test_invalid_group Create NetGroupGetUsers with valid details except group.	Exception raised	Exception raised		P
1.2.9.5.	test_nonexistant_group Create NetGroupGetUsers with valid details except non-existant group name supplied, and call enumerate method.	Empty list returned.	Empty list returned.		P
1.2.10.	DomainUsersSMBEnumerationTestCase				P
1.2.10.1.	test_valid Create NetUserEnum with valid details and call enumerate method.	Return non-empty list.	Return non-empty list.		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.10.2.	test_invalid_auth Create NetUserEnum with valid details except auth.	Exception raised	Exception raised		P
1.2.10.3.	test_invalid_host Create NetUserEnum with valid details except host.	Exception raised	Exception raised		P
1.2.11.	DomainUserInfoSMBEnumerationTestCase				
1.2.11.1.	test_valid Create NetUserGetInfo with valid details and call enumerate method.	Return non-empty dictionary	Return non-empty dictionary		P
1.2.11.2.	test_invalid_auth Create NetUserGetInfo with valid details except auth.	Exception raised	Exception raised		P
1.2.11.3.	test_invalid_host Create NetUserGetInfo with valid details except host.	Exception raised	Exception raised		P
1.2.11.4.	test_invalid_user Create NetUserGetInfo with valid details except user.	Exception raised	Exception raised		P
1.2.11.5.	test_nonexistant_user Create NetUserEnum with valid details except non-existent user specified and call enumerate method.	Empty dictionary returned.	Empty dictionary returned.		P
1.2.12.	DomainGroupMembershipSMBEnumerationTestCase				P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.12.1.	test_valid Create NetUserGetGroups with valid details and call enumerate method.	Return non-empty list	Return non-empty list		P
1.2.12.2.	test_invalid_auth Create NetUserGetGroups with valid details except auth.	Exception raised	Exception raised		P
1.2.12.3.	test_invalid_host Create NetUserGetGroups with valid details except host.	Exception raised	Exception raised		P
1.2.12.4.	test_invalid_user Create NetUserGetGroups with valid details except user.	Exception raised	Exception raised		P
1.2.12.5.	test_nonexistant_user Create NetUserGetGroups with valid details except non-existent user specified and call enumerate method.	Empty list returned.	Empty list returned.		P
1.2.13.	DomainPasswordPolicySMBEnumerationTestCase				P
1.2.13.1.	test_valid Create PasswordPolicy with valid details and call enumerate method.	Return dictionary not equal to default value.	Return dictionary not equal to default value.		P
1.2.13.2.	test_invalid_auth Create PasswordPolicy with valid details except auth.	Exception raised	Exception raised		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.13.3.	test_invalid_host Create PasswordPolicy with valid details except host.	Exception raised	Exception raised		P
1.2.14.	DomainLockoutPolicySMBEnumerationTestCase				P
1.2.14.1.	test_valid Create LockoutPolicy with valid details and call enumerate method.	Return dictionary not equal to default value.	Return dictionary not equal to default value.		P
1.2.14.2.	test_invalid_auth Create LockoutPolicy with valid details except auth.	Exception raised	Exception raised		P
1.2.14.3.	test_invalid_host Create LockoutPolicy with valid details except host.	Exception raised	Exception raised		P
1.2.15.	UserSNMPEnumerationTestCase				P
1.2.15.1.	test_valid Create SNMPUserEnum with valid details and call enumerate method.	Return non-empty list.	Return non-empty list.		P
1.2.15.2.	test_invalid_auth Create SNMPUserEnum with valid details except auth.	Exception raised	Exception raised		P
1.2.15.3.	test_invalid_host Create SNMPUserEnum with valid details except host.	Exception raised	Exception raised		P
1.2.16.	FromFQDNDomainNameEnumerationTestCase				P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.16.1.	test_valid Create FromFQDN object and call enumerate method	Return non-empty list	Return non-empty list		P
1.2.17.	FromObjectDomainNameEnumerationTestCase				P
1.2.17.1.	test_valid Create FromObject object and call enumerate method	Return non-empty list	Return non-empty list		P
1.2.18.	FromDNSServFQDNDomainNameEnumerationTestCase				P
1.2.18.1.	test_valid Create FromDNSServFQDN object and call enumerate method	Return non-empty list	Return non-empty list		P
1.2.19.	FromMasterDomainNameEnumerationTestCase				
1.2.19.1.	test_valid Create FromMaster object and call enumerate method	Return non-empty list	Return non-empty list	From Master not implemented	F
1.2.20.	DNSLookupDomainControllerEnumerationTestCase				P
1.2.20.1.	test_valid Create DNSLookup object with valid domain name and call enumerate method.	Return non-empty list	Return non-empty list		P
1.2.20.2.	test_invalid_domain_name Create DNSLookup object with invalid domain name and call enumerate method.	Exception raised	Exception raised		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.2.20.3.	test_nonexistent_domain_name Create DNSLookup object with non-existent domain name and call enumerate method.	Return empty list	Return empty list		P
1.2.21.	NetGetDCDomainControllerEnumerationTestCase				P
1.2.21.1.	test_valid Create NetGetDC object with valid domain name and call enumerate method.	Return non-empty list	Return non-empty list		P
1.2.21.2.	test_invalid_domain_name Create NetGetDC object with invalid domain name and call enumerate method.	Exception raised	Exception raised		P
1.2.21.3.	test_nonexistent_domain_name Create NetGetDC object with non-existent domain name and call enumerate method.	Return empty list	Return empty list		P
1.2.22.	DsGetDCNameDomainControllerEnumerationTestCase				P
1.2.22.1.	test_valid Create DsGetDCName object with valid domain name and call enumerate method.	Return non-empty list	Return non-empty list		P
1.2.22.2.	test_invalid_domain_name Create DsGetDCName object with invalid domain name and call enumerate method.	Exception raised	Exception raised		P
1.2.22.3.	test_nonexistent_domain_name Create DsGetDCName object with non-existent domain name and call enumerate method.	Return empty list	Return empty list		P

1.3. Reporters

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.3.1.	LocalReporterTestCase				P
1.3.1.1.	test_valid_data Create LocalXMLReporter providing an instance of Host data structure in a list.	None empty file created.	None empty file created.		P
1.3.1.2.	test_invalid_data1 Create LocalXMLReporter providing an instance of Domain data structure in a list.	Exception raised	Exception raised		P
1.3.1.3.	test_invalid_data2 Create LocalXMLReporter providing an instance of Host data structure.	Exception raised	Exception raised		P
1.3.1.4.	test_malformed_data1 Create LocalXMLReporter providing an instance of Host data structure (in a list) with malformed data as the top level variables (e.g. '_users=None').	None empty file created.	None empty file created.		P
1.3.1.5.	test_malforded_data2 Create LocalXMLReporter providing an instance of Host data structure (in a list) with malformed data as the second level variables (e.g. '_users' is a list of User instances with variables set to 'None').	None empty file created.	None empty file created.		P
1.3.2.	DomainReporterTestCase				P
1.3.2.1.	test_valid_data Create DomainXMLReporter providing an instance of Host data structure in a list.	None empty file created.	None empty file created.		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
1.3.2.2.	test_invalid_data1 Create DomainXMLReporter providing an instance of Domain data structure in a list.	Exception raised	Exception raised		P
1.3.2.3.	test_invalid_data2 Create DomainXMLReporter providing an instance of Host data structure.	Exception raised	Exception raised		P
1.3.2.4.	test_malformed_data1 Create DomainXMLReporter providing an instance of Host data structure (in a list) with malformed data as the top level variables (e.g. '_users=None').	None empty file created.	None empty file created.		P
1.3.2.5.	test_malforded_data2 Create DomainXMLReporter providing an instance of Host data structure (in a list) with malformed data as the second level variables (e.g. '_users' is a list of User instances with variables set to 'None').	None empty file created.	None empty file created.		P

2. Integration Testing

2.1. Automation

Test #	Case	Expected	Actual	Comments	Pass/Fail
2.1.1.	Domain Automation User specifies domain automation. System enumerates all information and generates a report.	File created containing enumerated information.	File created containing enumerated information.		P
2.1.2.	Local Automation User specifies local automation. System enumerates all information and generates a report.	File created containing enumerated information.	File created containing enumerated information.		P

2.2. Manual

Test #	Case	Expected	Actual	Comments	Pass/Fail
2.2.1.	Local				P
2.2.1.1.	enumerate_interesting_hosts User specifies enumerating a list of interesting hosts. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.1.2.	enumerate_local_policies User specifies enumerating a local policy. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
2.2.1.3.	enumerate_local_groups User specifies enumerating a local groups. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.1.4.	enumerate_local_group_members User specifies enumerating a local group members. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.1.5.	enumerate_local_users User specifies enumerating a local users. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.1.6.	enumerate_local_user User specifies enumerating a local user information. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.1.7.	enumerate_group_membership User specifies enumerating a local user group membership. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.2.	Domain				P

Test #	Case	Expected	Actual	Comments	Pass/Fail
2.2.2.1.	enumerate_domain_names User specifies enumerating a list of domain names. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.2.2.	enumerate_domain_controllers User specifies enumerating a list of domain controllers. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.2.3.	enumerate_domain_policies User specifies enumerating a domain policies. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.2.4.	enumerate_domain_groups User specifies enumerating domain groups. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.2.5.	enumerate_domain_group_members User specifies enumerating domain group members. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.2.6.	enumerate_domain_users User specifies enumerating domain users. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
2.2.2.7.	enumerate_domain_user User specifies enumerating domain user information. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P
2.2.2.8.	enumerate_group_membership User specifies enumerating domain user group membership. System enumerates the information and generates a report.	File containing enumerated information.	File containing enumerated information.		P

3. System Testing

3.1. Requirements

Test #	Case	Expected	Actual	Comments	Pass/Fail
3.1.1.	Essential				P
3.1.1.1.	Run on a machine that is not a member of a domain	Enumeration functions.	Enumeration functions.		P
3.1.1.2.	Run on a Microsoft Windows machine	Enumeration functions.	Enumeration functions on Windows 7 x64.	Other versions of windows were not tested.	P
3.1.1.3.	Identify domain names on a local area network	List of domain names enumerated.	List of domain names enumerated.		P
3.1.1.4.	Identify domain controllers	List of domain controllers enumerated.	List of domain controllers enumerated.		P
3.1.1.5.	Retrieve a list of users from the active directory	List of users enumerated.	List of users enumerated.		P
3.1.1.6.	Retrieve a list of groups from the active directory	List of groups enumerated.	List of groups enumerated.		P
3.1.1.7.	Retrieve a list of group members from the active directory	List of group members enumerated.	List of group members enumerated.		P
3.1.1.8.	Retrieve user account information from the active directory	User account information enumerated.	User account information enumerated.		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
3.1.1.9.	Retrieve the domain accounts policy (Lockout: duration, threshold, observation window. Password: length, character set)	Domain accounts policy enumerated.	Domain accounts policy enumerated.		P
3.1.1.10.	Perform automated enumeration	Automated operation.	Automated operation.		P
3.1.1.11.	Output in a parseable and human readable format	Output format parseable and readable.	XML output with XSLT.	XML is parseable and readable in its raw form. When viewed in a web browser the XSLT generates a HTML page which is easier to read than XML.	P
3.1.1.12.	Provide a help / usage guide	Output usage when incorrect parameters entered at the command line. Output usage when '-h' specified. Output help text when '--help' specified.	Output usage when incorrect parameters entered at the command line. Output usage when '-h' specified. Output help text when '--help' specified.		P
3.1.2.	Desirable				
3.1.2.1.	Perform directed enumeration	Directed enumeration options.	Directed enumeration options.		P

Test #	Case	Expected	Actual	Comments	Pass/Fail
3.1.2.2.	Identify services running on hosts	Identify a list of services running on a host.			P
3.1.2.3.	Identify domain trusts	List the trust relationships between domains.	None	Not implemented because function not available in `pywin32`	F
3.1.2.4.	Retrieve a list of users from a local host	List of users enumerated.	List of users enumerated.		P
3.1.2.5.	Retrieve a list of groups from a local host	List of groups enumerated.	List of groups enumerated.		P
3.1.2.6.	Retrieve a list of group members a local host	List of group members enumerated.	List of group members enumerated.		P
3.1.2.7.	Retrieve information user account information from a local host	User account information enumerated.	User account information enumerated.		P
3.1.2.8.	Retrieve the local accounts policy (Lockout: duration, threshold, observation window. Password: length, character set)	Local accounts policy enumerated.	Local accounts policy enumerated.		P
3.1.2.9.	Retrieve a list of shares from a local host	List of shares available.	List of shares available with remark and type.		P

3.2. Documentation

Test #	Case	Expected	Actual	Comments	Pass/Fail
3.2.1.	Install stated requirements and run examples in documentation.	No additional requirements, examples function.	No additional requirements, examples function.		P

3.3. Precision and Recall

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.1.	Local						
3.3.1.1.	enumerate_interesting_hosts for testdomain.local	2	2	2		1	1
3.3.1.2.	enumerate_local_policies	8	8	8	Note: Times are displayed in seconds in nettynum's output.	1	1
3.3.1.3.	enumerate_local_groups	28	28	28		1	1
3.3.1.4.	enumerate_local_group_members						
3.3.1.4.1.	Account Operators	0	0	0		1	1
3.3.1.4.2.	Administrators	4	4	4		1	1
3.3.1.4.3.	Allowed RODC Password Replication Group	0	0	0		1	1
3.3.1.4.4.	Backup Operators	0	0	0		1	1
3.3.1.4.5.	Cert Publishers	0	0	0		1	1
3.3.1.4.6.	Certificate Service DCOM Access	0	0	0		1	1
3.3.1.4.7.	Cryptographic Operators	0	0	0		1	1
3.3.1.4.8.	Denied RODC Password Replication Group	8	8	8		1	1
3.3.1.4.9.	DHCP Administrators	0	0	0		1	1
3.3.1.4.10.	DHCP Users	0	0	0		1	1
3.3.1.4.11.	Distributed COM Users	0	0	0		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.1.4.12.	DnsAdmins	0	0	0		1	1
3.3.1.4.13.	Event Log Readers	0	0	0		1	1
3.3.1.4.14.	Guests	2	2	2		1	1
3.3.1.4.15.	IIS_IUSRS	1	1	1		1	1
3.3.1.4.16.	Incoming Forest Trust Builders	0	0	0		1	1
3.3.1.4.17.	Network Configuration Operators	0	0	0		1	1
3.3.1.4.18.	Performance Log Users	0	0	0		1	1
3.3.1.4.19.	Performance Monitor Users	0	0	0		1	1
3.3.1.4.20.	Pre-Windows 2000 Compatible Access	1	1	1		1	1
3.3.1.4.21.	Print Operators	0	0	0		1	1
3.3.1.4.22.	RAS and IAS Servers	0	0	0		1	1
3.3.1.4.23.	Remote Desktop Users	0	0	0		1	1
3.3.1.4.24.	Replicator	0	0	0		1	1
3.3.1.4.25.	Server Operators	0	0	0		1	1
3.3.1.4.26.	Terminal Server License Servers	0	0	0		1	1
3.3.1.4.27.	Users	3	3	3		1	1
3.3.1.4.28.	Windows Authorization Access Group	1	1	1		1	1
3.3.1.5.	enumerate_users	53	53	53		1	1
3.3.1.6.	enumerate_user_info						

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.1.6.1.	Administrator	29	27	27	Params and logon_hours not returned.	1	0.931
3.3.1.6.2.	Guest	29	27	27		1	0.931
3.3.1.6.3.	krbtgt	29	27	27		1	0.931
3.3.1.6.4.	user1	29	27	27		1	0.931
3.3.1.6.5.	user2	29	27	27		1	0.931
3.3.1.6.6.	user3	29	27	27		1	0.931
3.3.1.6.7.	user4	29	27	27		1	0.931
3.3.1.6.8.	user5	29	27	27		1	0.931
3.3.1.6.9.	user6	29	27	27		1	0.931
3.3.1.6.10	user7	29	27	27		1	0.931
3.3.1.6.11	user8	29	27	27		1	0.931
3.3.1.6.12	user9	29	27	27		1	0.931
3.3.1.6.13	user10	29	27	27		1	0.931
3.3.1.6.14	user11	29	27	27		1	0.931
3.3.1.6.15	user12	29	27	27		1	0.931
3.3.1.6.16	user13	29	27	27		1	0.931
3.3.1.6.17	user14	29	27	27		1	0.931
3.3.1.6.18	user15	29	27	27		1	0.931
3.3.1.6.19	user16	29	27	27		1	0.931
3.3.1.6.20	user17	29	27	27		1	0.931
3.3.1.6.21	user18	29	27	27		1	0.931
3.3.1.6.22	user19	29	27	27		1	0.931
3.3.1.6.23	user20	29	27	27		1	0.931
3.3.1.6.24	user21	29	27	27		1	0.931

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.1.6.25.	user22	29	27	27		1	0.931
3.3.1.6.26.	user23	29	27	27		1	0.931
3.3.1.6.27.	user24	29	27	27		1	0.931
3.3.1.6.28.	user25	29	27	27		1	0.931
3.3.1.6.29.	user26	29	27	27		1	0.931
3.3.1.6.30.	user27	29	27	27		1	0.931
3.3.1.6.31.	user28	29	27	27		1	0.931
3.3.1.6.32.	user29	29	27	27		1	0.931
3.3.1.6.33.	user30	29	27	27		1	0.931
3.3.1.6.34.	user31	29	27	27		1	0.931
3.3.1.6.35.	user32	29	27	27		1	0.931
3.3.1.6.36.	user33	29	27	27		1	0.931
3.3.1.6.37.	user34	29	27	27		1	0.931
3.3.1.6.38.	user35	29	27	27		1	0.931
3.3.1.6.39.	user36	29	27	27		1	0.931
3.3.1.6.40.	user37	29	27	27		1	0.931
3.3.1.6.41.	user38	29	27	27		1	0.931
3.3.1.6.42.	user39	29	27	27		1	0.931
3.3.1.6.43.	user40	29	27	27		1	0.931
3.3.1.6.44.	user41	29	27	27		1	0.931
3.3.1.6.45.	user42	29	27	27		1	0.931
3.3.1.6.46.	user43	29	27	27		1	0.931
3.3.1.6.47.	user44	29	27	27		1	0.931
3.3.1.6.48.	user45	29	27	27		1	0.931

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.1.6.49.	user46	29	27	27		1	0.931
3.3.1.6.50.	user47	29	27	27		1	0.931
3.3.1.6.51.	user48	29	27	27		1	0.931
3.3.1.6.52.	user49	29	27	27		1	0.931
3.3.1.6.53.	user50	29	27	27		1	0.931
3.3.1.7.	enumerate_group_membership						
3.3.1.7.1.	Administrator	3	3	3		1	1
3.3.1.7.2.	Guest	1	1	1		1	1
3.3.1.7.3.	krbtgt	2	2	2		1	1
3.3.1.7.4.	user1	1	1	1		1	1
3.3.1.7.5.	user2	1	1	1		1	1
3.3.1.7.6.	user3	1	1	1		1	1
3.3.1.7.7.	user4	1	1	1		1	1
3.3.1.7.8.	user5	1	1	1		1	1
3.3.1.7.9.	user6	1	1	1		1	1
3.3.1.7.10.	user7	1	1	1		1	1
3.3.1.7.11.	user8	1	1	1		1	1
3.3.1.7.12.	user9	1	1	1		1	1
3.3.1.7.13.	user10	1	1	1		1	1
3.3.1.7.14.	user11	1	1	1		1	1
3.3.1.7.15.	user12	1	1	1		1	1
3.3.1.7.16.	user13	1	1	1		1	1
3.3.1.7.17.	user14	1	1	1		1	1
3.3.1.7.18.	user15	1	1	1		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.1.7.19.	user16	1	1	1		1	1
3.3.1.7.20.	user17	1	1	1		1	1
3.3.1.7.21.	user18	1	1	1		1	1
3.3.1.7.22.	user19	1	1	1		1	1
3.3.1.7.23.	user20	1	1	1		1	1
3.3.1.7.24.	user21	1	1	1		1	1
3.3.1.7.25.	user22	1	1	1		1	1
3.3.1.7.26.	user23	1	1	1		1	1
3.3.1.7.27.	user24	1	1	1		1	1
3.3.1.7.28.	user25	1	1	1		1	1
3.3.1.7.29.	user26	1	1	1		1	1
3.3.1.7.30.	user27	1	1	1		1	1
3.3.1.7.31.	user28	1	1	1		1	1
3.3.1.7.32.	user29	1	1	1		1	1
3.3.1.7.33.	user30	1	1	1		1	1
3.3.1.7.34.	user31	1	1	1		1	1
3.3.1.7.35.	user32	1	1	1		1	1
3.3.1.7.36.	user33	1	1	1		1	1
3.3.1.7.37.	user34	1	1	1		1	1
3.3.1.7.38.	user35	1	1	1		1	1
3.3.1.7.39.	user36	1	1	1		1	1
3.3.1.7.40.	user37	1	1	1		1	1
3.3.1.7.41.	user38	1	1	1		1	1
3.3.1.7.42.	user39	1	1	1		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.1.7.43.	user40	1	1	1		1	1
3.3.1.7.44.	user41	1	1	1		1	1
3.3.1.7.45.	user42	1	1	1		1	1
3.3.1.7.46.	user43	1	1	1		1	1
3.3.1.7.47.	user44	1	1	1		1	1
3.3.1.7.48.	user45	1	1	1		1	1
3.3.1.7.49.	user46	1	1	1		1	1
3.3.1.7.50.	user47	1	1	1		1	1
3.3.1.7.51.	user48	1	1	1		1	1
3.3.1.7.52.	user49	1	1	1		1	1
3.3.1.7.53.	user50	1	1	1		1	1
3.3.2.	Domain						
3.3.2.1.	enumerate_domain_names	2	2	2		1	1
3.3.2.2.	enumerate_domain_controllers	2	2	2		1	1
3.3.2.3.	enumerate_domain_policies	8	8	8		1	1
3.3.2.4.	enumerate_domain_groups	61	61	61		1	1
3.3.2.5.	enumerate_domain_group_members						
3.3.2.5.1.	DnsUpdateProxy	0	0	0		1	1
3.3.2.5.2.	Domain Admins	1	1	1		1	1
3.3.2.5.3.	Domain Computers	0	0	0		1	1
3.3.2.5.4.	Domain Controllers	2	2	2		1	1
3.3.2.5.5.	Domain Guests	0	0	0		1	1
3.3.2.5.6.	Domain Users	52	52	52		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.2.5.7.	Enterprise Admins	1	1	1		1	1
3.3.2.5.8.	Enterprise Read-only Domain Controllers	0	0	0		1	1
3.3.2.5.9.	Group Policy Creator Owners	1	1	1		1	1
3.3.2.5.10.	Read-only Domain Controllers	0	0	0		1	1
3.3.2.5.11.	Schema Admins	1	1	1		1	1
3.3.2.5.12.	group1	10	10	10		1	1
3.3.2.5.13.	group2	10	10	10		1	1
3.3.2.5.14.	group3	10	10	10		1	1
3.3.2.5.15.	group4	10	10	10		1	1
3.3.2.5.16.	group5	10	10	10		1	1
3.3.2.5.17.	group6	10	10	10		1	1
3.3.2.5.18.	group7	10	10	10		1	1
3.3.2.5.19.	group8	10	10	10		1	1
3.3.2.5.20.	group9	10	10	10		1	1
3.3.2.5.21.	group10	10	10	10		1	1
3.3.2.5.22.	group11	10	10	10		1	1
3.3.2.5.23.	group12	10	10	10		1	1
3.3.2.5.24.	group13	10	10	10		1	1
3.3.2.5.25.	group14	10	10	10		1	1
3.3.2.5.26.	group15	10	10	10		1	1
3.3.2.5.27.	group16	10	10	10		1	1
3.3.2.5.28.	group17	10	10	10		1	1
3.3.2.5.29.	group18	10	10	10		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.2.5.30	group19	10	10	10		1	1
3.3.2.5.31	group20	10	10	10		1	1
3.3.2.5.32	group21	10	10	10		1	1
3.3.2.5.33	group22	10	10	10		1	1
3.3.2.5.34	group23	10	10	10		1	1
3.3.2.5.35	group24	10	10	10		1	1
3.3.2.5.36	group25	10	10	10		1	1
3.3.2.5.37	group26	10	10	10		1	1
3.3.2.5.38	group27	10	10	10		1	1
3.3.2.5.39	group28	10	10	10		1	1
3.3.2.5.40	group29	10	10	10		1	1
3.3.2.5.41	group30	10	10	10		1	1
3.3.2.5.42	group31	10	10	10		1	1
3.3.2.5.43	group32	10	10	10		1	1
3.3.2.5.44	group33	10	10	10		1	1
3.3.2.5.45	group34	10	10	10		1	1
3.3.2.5.46	group35	10	10	10		1	1
3.3.2.5.47	group36	10	10	10		1	1
3.3.2.5.48	group37	10	10	10		1	1
3.3.2.5.49	group38	10	10	10		1	1
3.3.2.5.50	group39	10	10	10		1	1
3.3.2.5.51	group40	10	10	10		1	1
3.3.2.5.52	group41	10	10	10		1	1
3.3.2.5.53	group42	10	10	10		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.2.5.54	group43	10	10	10		1	1
3.3.2.5.55	group44	10	10	10		1	1
3.3.2.5.56	group45	10	10	10		1	1
3.3.2.5.57	group46	10	10	10		1	1
3.3.2.5.58	group47	10	10	10		1	1
3.3.2.5.59	group48	10	10	10		1	1
3.3.2.5.60	group49	10	10	10		1	1
3.3.2.5.61	group50	10	10	10		1	1
3.3.2.6.	enumerate_users	53	53	53		1	1
3.3.2.7.	enumerate_user_info					1	
3.3.2.7.1.	Administrator	29	27	27	Params and logon_hours not returned.	1	0.931
3.3.2.7.2.	Guest	29	27	27		1	0.931
3.3.2.7.3.	krbtgt	29	27	27		1	0.931
3.3.2.7.4.	user1	29	27	27		1	0.931
3.3.2.7.5.	user2	29	27	27		1	0.931
3.3.2.7.6.	user3	29	27	27		1	0.931
3.3.2.7.7.	user4	29	27	27		1	0.931
3.3.2.7.8.	user5	29	27	27		1	0.931
3.3.2.7.9.	user6	29	27	27		1	0.931
3.3.2.7.10	user7	29	27	27		1	0.931
3.3.2.7.11	user8	29	27	27		1	0.931
3.3.2.7.12	user9	29	27	27		1	0.931
3.3.2.7.13	user10	29	27	27		1	0.931
3.3.2.7.14	user11	29	27	27		1	0.931

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.2.7.15.	user12	29	27	27		1	0.931
3.3.2.7.16.	user13	29	27	27		1	0.931
3.3.2.7.17.	user14	29	27	27		1	0.931
3.3.2.7.18.	user15	29	27	27		1	0.931
3.3.2.7.19.	user16	29	27	27		1	0.931
3.3.2.7.20.	user17	29	27	27		1	0.931
3.3.2.7.21.	user18	29	27	27		1	0.931
3.3.2.7.22.	user19	29	27	27		1	0.931
3.3.2.7.23.	user20	29	27	27		1	0.931
3.3.2.7.24.	user21	29	27	27		1	0.931
3.3.2.7.25.	user22	29	27	27		1	0.931
3.3.2.7.26.	user23	29	27	27		1	0.931
3.3.2.7.27.	user24	29	27	27		1	0.931
3.3.2.7.28.	user25	29	27	27		1	0.931
3.3.2.7.29.	user26	29	27	27		1	0.931
3.3.2.7.30.	user27	29	27	27		1	0.931
3.3.2.7.31.	user28	29	27	27		1	0.931
3.3.2.7.32.	user29	29	27	27		1	0.931
3.3.2.7.33.	user30	29	27	27		1	0.931
3.3.2.7.34.	user31	29	27	27		1	0.931
3.3.2.7.35.	user32	29	27	27		1	0.931
3.3.2.7.36.	user33	29	27	27		1	0.931
3.3.2.7.37.	user34	29	27	27		1	0.931
3.3.2.7.38.	user35	29	27	27		1	0.931

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.2.7.39.	user36	29	27	27		1	0.931
3.3.2.7.40.	user37	29	27	27		1	0.931
3.3.2.7.41.	user38	29	27	27		1	0.931
3.3.2.7.42.	user39	29	27	27		1	0.931
3.3.2.7.43.	user40	29	27	27		1	0.931
3.3.2.7.44.	user41	29	27	27		1	0.931
3.3.2.7.45.	user42	29	27	27		1	0.931
3.3.2.7.46.	user43	29	27	27		1	0.931
3.3.2.7.47.	user44	29	27	27		1	0.931
3.3.2.7.48.	user45	29	27	27		1	0.931
3.3.2.7.49.	user46	29	27	27		1	0.931
3.3.2.7.50.	user47	29	27	27		1	0.931
3.3.2.7.51.	user48	29	27	27		1	0.931
3.3.2.7.52.	user49	29	27	27		1	0.931
3.3.2.7.53.	user50	29	27	27		1	0.931
3.3.2.8.	enumerate_group_membership						
3.3.2.8.1.	Administrator	5	5	5		1	1
3.3.2.8.2.	Guest	1	1	1		1	1
3.3.2.8.3.	krbtgt	1	1	1		1	1
3.3.2.8.4.	user1	11	11	11		1	1
3.3.2.8.5.	user2	11	11	11		1	1
3.3.2.8.6.	user3	11	11	11		1	1
3.3.2.8.7.	user4	11	11	11		1	1
3.3.2.8.8.	user5	11	11	11		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.2.8.9.	user6	11	11	11		1	1
3.3.2.8.10.	user7	11	11	11		1	1
3.3.2.8.11.	user8	11	11	11		1	1
3.3.2.8.12.	user9	11	11	11		1	1
3.3.2.8.13.	user10	11	11	11		1	1
3.3.2.8.14.	user11	11	11	11		1	1
3.3.2.8.15.	user12	11	11	11		1	1
3.3.2.8.16.	user13	11	11	11		1	1
3.3.2.8.17.	user14	11	11	11		1	1
3.3.2.8.18.	user15	11	11	11		1	1
3.3.2.8.19.	user16	11	11	11		1	1
3.3.2.8.20.	user17	11	11	11		1	1
3.3.2.8.21.	user18	11	11	11		1	1
3.3.2.8.22.	user19	11	11	11		1	1
3.3.2.8.23.	user20	11	11	11		1	1
3.3.2.8.24.	user21	11	11	11		1	1
3.3.2.8.25.	user22	11	11	11		1	1
3.3.2.8.26.	user23	11	11	11		1	1
3.3.2.8.27.	user24	11	11	11		1	1
3.3.2.8.28.	user25	11	11	11		1	1
3.3.2.8.29.	user26	11	11	11		1	1
3.3.2.8.30.	user27	11	11	11		1	1
3.3.2.8.31.	user28	11	11	11		1	1
3.3.2.8.32.	user29	11	11	11		1	1

Test #	Case	Real Positive (RP)	Predicted Positive (PP)	True Positive (TP)	Comment	Precision = TP/PP	Recall = TP/RP
3.3.2.8.33	user30	11	11	11		1	1
3.3.2.8.34	user31	11	11	11		1	1
3.3.2.8.35	user32	11	11	11		1	1
3.3.2.8.36	user33	11	11	11		1	1
3.3.2.8.37	user34	11	11	11		1	1
3.3.2.8.38	user35	11	11	11		1	1
3.3.2.8.39	user36	11	11	11		1	1
3.3.2.8.40	user37	11	11	11		1	1
3.3.2.8.41	user38	11	11	11		1	1
3.3.2.8.42	user39	11	11	11		1	1
3.3.2.8.43	user40	11	11	11		1	1
3.3.2.8.44	user41	11	11	11		1	1
3.3.2.8.45	user42	11	11	11		1	1
3.3.2.8.46	user43	11	11	11		1	1
3.3.2.8.47	user44	11	11	11		1	1
3.3.2.8.48	user45	11	11	11		1	1
3.3.2.8.49	user46	11	11	11		1	1
3.3.2.8.50	user47	11	11	11		1	1
3.3.2.8.51	user48	11	11	11		1	1
3.3.2.8.52	user49	11	11	11		1	1
3.3.2.8.53	user50	11	11	11		1	1

Appendix F: Beta Testing Feedback

The following feedback is indicative of that received from the Beta testers. Note most communication was through informal channels and very little was in a written form.

The nettynum tool has proven to be very useful and practical for internal network penetration testing, since it combines in a novel way many related probes and features that are cumbersome to integrate with existing tools. It would be good to have some Linux portability too, to allow wider integration into penetration testing systems, though the code design lends itself well to this functionality being added at a later stage.

Dr N Blundell (Chief Software Architect, Sec-1 Ltd)

[03/04/2014 14:17:57] Richard Jones: Coupla issues re layout
[03/04/2014 14:18:35] Richard Jones: It may be my browser (IE) but when I click on a + to expand, it keeps jumping to the top of the document
[03/04/2014 14:19:16] Richard Jones: and I think I personally would prefer it in more of a table format, than a list with +'s
…
[03/04/2014 14:25:38] Richard Jones: It does look good though dude, nice work fella!
[03/04/2014 14:25:55] Richard Jones: Think it's gonna be really handy on internals ;)
…
[03/04/2014 14:36:25] Richard Jones: Error handling is the first comment I'd make ;)
…
[03/04/2014 14:41:24] Richard Jones: … it looks pretty nicely laid out and commented, so yeah, lookin good dude!

[03/04/2014 09:58:24] Matthew Hall: for me - needs pywin32 so sort of stopped trying to install it under linux there ;)
…
[03/04/2014 10:05:48] Matthew Hall: ill see if I can get it working in my win vm
[03/04/2014 10:14:56] Matthew Hall: ok - feedback 1 : convert the password ages from integer/decimals to human readable (max password age 7776000 is 36 days?)
…
[03/04/2014 10:18:50] Matthew Hall: pretty useful tool tho olly. have you thought about a greppable output?