

Advanced Security And Network Forensics



Outline Details

Module Leader:	Prof Bill Buchanan
Module number:	CSN10102
Email:	w.buchanan@napier.ac.uk
Telephone:	X2759
Web page:	http://buchananweb.co.uk/index_asfc_napier.html
Within ProfSIMS:	Adv Security and Network Forensics (link)
MSN Messenger:	w_j_buchanan@hotmail.com
Version:	Semester 2, 2010/2011

Associated Software

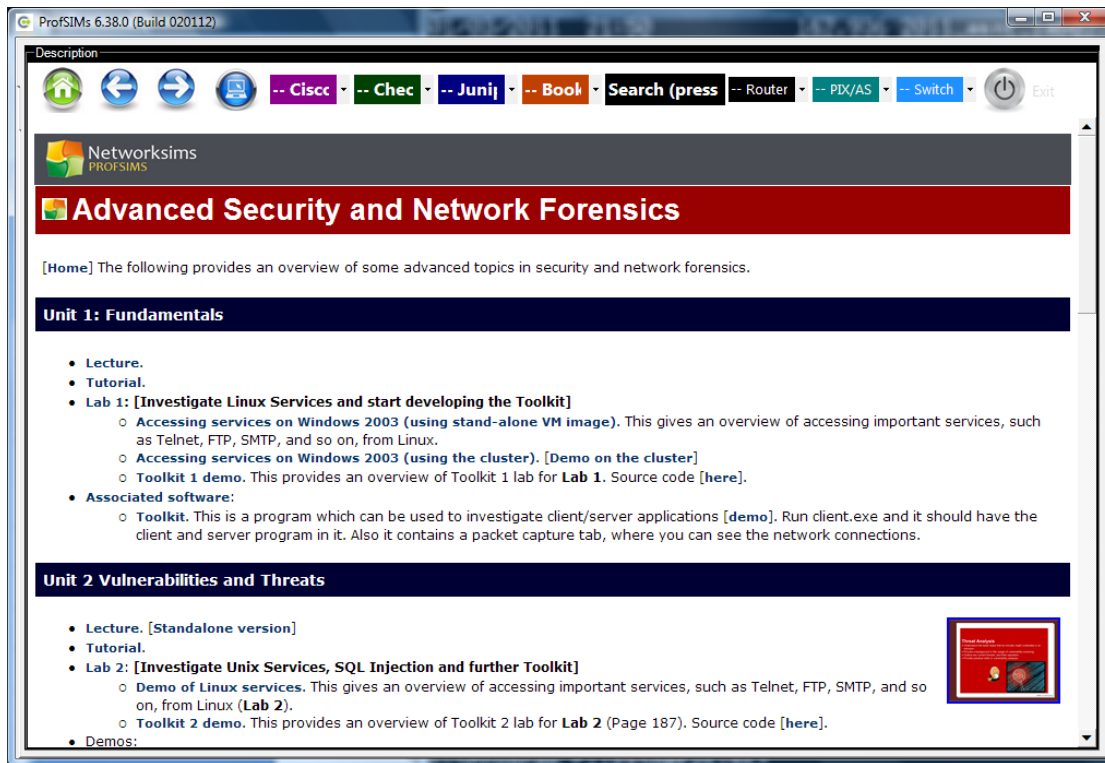
1. **NetworkSims.com ProfSIMS (install and register with your Napier email address)**
<http://www.dcs.napier.ac.uk/~bill/napier.zip>
2. **Toolkit**
<http://buchananweb.co.uk/dotnetclientserver.zip>
3. **WinPcap**
See Web-CT
4. **Snort (Windows)**
See Web-CT
5. **Visual Studio**
Please register onto the MSDN AA, and download Visual Studio 2008. Otherwise download it from <http://www.dreamspark.com> (use Athens to authenticate yourself).

NetworkSims ProfSIMS

The NetworkSims ProfSIMS package can be downloaded from:

<http://www.dcs.napier.ac.uk/~bill/napier.zip>

and contains an up-to-date version of the tests for each of the chapters. The follow graphic shows the main user interface. From this screen you should be able to access a range of related material.



Connecting to the cluster

From an Internet Explorer browser, connect to: **146.176.166.54** (or **Im2003.napier.ac.uk**), and accept the exceptions for the digital certification. Next connect to the Advanced Security and Network Forensics workspace (see Figure 1).

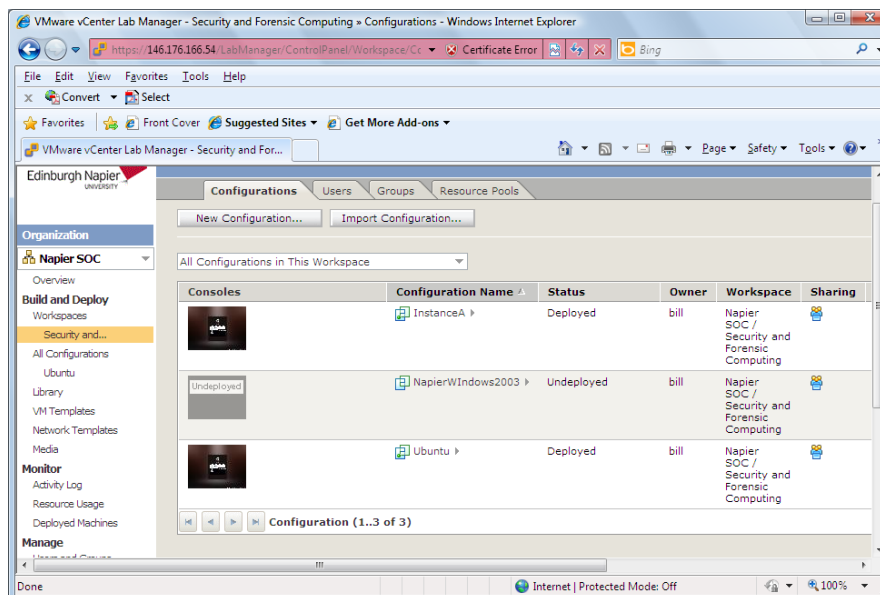


Figure 1: Advanced Security and Network Forensics Workspace

Module Definition

Name:	Advanced Security and Network Forensics
Module Leader:	Prof WJ Buchanan, School of Computing
Contact:	w.buchanan@napier.ac.uk, Room: C.63
MSN Messenger Contact:	w_j_buchanan@hotmail.com
Lectures:	24 hours
Practical:	24 hours
Student Centered Learning:	152 hours
Book:	[1]

Module content

The aim of the module is to develop a deep understanding of advanced areas related to security, digital forensics and next-generation Web-based systems, that will allow graduates to act professionally in the design, analysis, implementation, and reporting of enhanced software systems, security strategies, and in forensic computing investigations. An outline of the main areas includes:

1. Security Threats, Security Models, Security Evaluation, and Mitigation Strategies. This involves an in-depth analysis of a range of current threats.
2. Secure Architectures/Frameworks.
3. Network, Cloud, e-Discovery and Mobile Forensics.
4. Data hiding. Data hiding methods, Covert Channel Analysis, Stenography.
5. Cloud/grid computing. Principles, distributed architectures, and layered approaches. Data grid and Semantic Web. Cloud/grid applications.
6. Service-oriented architectures (SOAs). Interfacing to SOAs, middleware infrastructures, and discovering services.
7. Web-service/Remoting infrastructures and Service Platforms. Example Clouds (Amazon EC2, Google, and so on), Web security (Focus on Web-based infrastructures, XAML and WS-*, SAML), Next Generation Web-based Systems.
8. Web Security, Authentication infrastructures. Principles, Kerberos, enhanced and scalable authentication infrastructures.
9. Virtualization methodologies. Software-/Hardware-as-a-Service (SaaS/HaaS), Fault tolerance and reliability, Distributed data storage/Clustering. High-performance computing.

Week	Date	Academic	Assessment	Lab/Tutorial
2	24/01/12	1: Introduction		Lab 1: Windows Services/ Toolkit 1 (Cmds)
3	31/01/12	2: Threats		Lab 2: Linux Services/ Toolkit 2 (WinDump)
4	07/02/12	3: Network Forensics		Lab 3: Vulnerability Analysis/ Toolkit 3 (Snort/Nmap)
5	14/02/12	4: Data Hiding		Lab 4: Network Forensics/ Toolkit 4 (Packet Capture)
6	21/02/12	Revision		Lab 5: Data Hiding/ Toolkit 5 (Analysis)
7	28/02/12		MCQ Test 1	Lab 6: Secure Connections/ Toolkit 6 (Data Hiding)
8	06/03/12	5: Web Infrastructures		Lab 7: SAML/ Toolkit 7 (URL cache)
9	13/03/12	6: Cloud/grid computing		Lab 8: Using AWS (Web, Database, Telnet and FTP)
10	20/03/12	7: Disk and Live Forensics		Lab 9: Using AWS (LAMP)
11	27/03/12	Professional Certification (CEH)		Lab 10: Integrated Forensics
12	03/04/12	Professional Certification (CEH)		
13	24/04/12		MCQ Test 2/ C/W hand-in	
14	01/05/12			
15	08/05/12			

1 Introduction

🔗 On-line lecture: <http://buchananweb.co.uk/adv/unit01.html>

🔗 Tutorial: NetworkSims

1.1 Objectives

The key objectives of this unit are to:

- Provide an outline of risk, and the terminology used.
- Provide an outline to a range of threats.
- Understand the usage of client/server connections.
- Outline the usage of services on Windows and Linux, and provide an introduction to service-oriented infrastructures.
- Provide a practical background in Windows and Linux for services, logging and auditing.

1.2 Introduction

ISO 27001/2 is a key standard and defines best practice related to information security. An important element of this standard is the assessment of risks, and their associated threats. This chapter outlines some of the key classifications of threats, and Unit 2 discusses how vulnerabilities can be assessed. It also aims to define some of the terminology that can be used to define a security incident, and outlines a formal taxonomy which defines common definitions for key terms. The ISO 27001/2 standard started life as “Information Security Code of Practice” from the UK (DTI), and was published in the 1990. It recently changed from ISO/IEC 17799 to ISO/IEC 27001/2 and provides a benchmark for most areas of security. Overall it defines 12 main areas, which are defined in Appendix A.

The key focus of this chapter is to provide a background to key areas, including the integration of services within computing environments, as this will provide a core to the module. It thus covers the key principles of client-server communications, where a service is advertised at a known Internet address, on a defined TCP port. A client can thus easily find the service, if it knows both these parameters. This type of communication is the standard one used on the Internet, and is also the basis of service-oriented architecture, where services are distributed around a network. In this way it becomes easier to provide robustness and maintenance, while allowing devices of limited capacity access to services which they might not be able to normally support on a local basis. Much of modern security and network forensics are also build around the usage of services, thus they will provide a key focus of the rest of the module.

1.3 Security Taxonomy

Along with defining an ontology to allow a wide range of professionals to use a standardized method of defining risk, there needs to be clear classifications for

threats and their mitigation procedures. Abbas (2006), for example, defines a taxonomy for Internet security with a number of attack strategies:

- **Manual Penetrating the System and/or Individual Privacy.** This includes methods and techniques that define the manual system penetration and includes password cracking, social engineering, and masquerading.
- **Data Interception, Interruption, and Replaying.** This includes methods of intercepting information or communication processes, and includes the tampering of transmitted data.
- **Biometrics and Physical Token.** This includes attacks related to physical and biometrical methods, such as for copying a biometric scan, and their associated processes.
- **Defeating Mechanisms and Policy.** This includes comprising the authentication, authorization, and access control infrastructure and their associated policies.
- **Malicious Code.** This includes malicious software, viruses, malfeasant code, bugs, coding problems, and so on.
- **Distributed Communication Systems.** This includes methods to attack that uses network communication protocols, such as for Distributed Denial of Service (DDoS).

A fairly abstract representation is provided by Hollingworth (2009) which classifies a security incident in terms of threats, attack tools, vulnerabilities, access methods, results and objectives. For example (Figure 1.1):

A Threat:

- Hacker.
- Spies
- Terrorists.
- Corporate Raiders.
- Professional Criminals.
- Vandals.
- Military Forces.

is achieved with **Attack Tools:**

- User command.
- Script or program.
- Autonomous Agent.
- Toolkit
- Distributed Tool.
- Data Tap.

for **Vulnerabilities:**

- Implementation vulnerability.
- Design vulnerability.
- Configuration vulnerability.

with **Access:**

- Unauthorized Access of Processes for:
- Unauthorized Use of Processes for:
 - Files.

- Data in transit.
- Objects in Transit.
- Invocations in Transit.

which **Results** in:

- Corruption of Information.
- Disclosure of Information.
- Theft of Service.
- Denial-of-Service.

for **Objectives**:

- Challenge/Status.
- Political Gain.
- Financial Gain.
- Damage.
- Destruction of an Enemy.

Thus for the Russian attack on Estonia with a denial-of-service attack might be defined with:

[Government Agents] used [Autonomous Agents] to exploit a [Design Vulnerability] for [Unauthorized Use] of [Processes] for [Invocations in Transit] which resulted in a [Denial-of-Service] for [Political Gain]

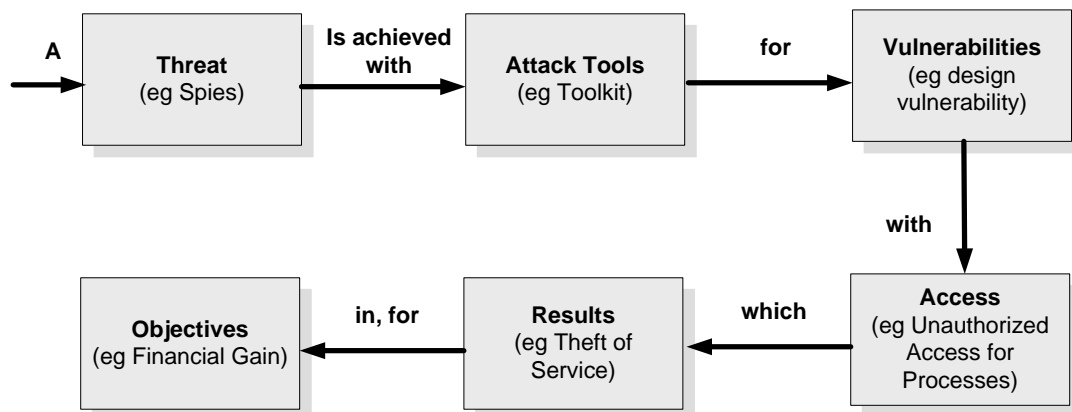


Figure 1.1 Security incident taxonomy

1.4 Threats

Kent (2003) defines that a threat relates to an adversary motivated and capable of exploiting a vulnerability. From now on, an adversary will be known as an intruder. There are various properties which can relate to this threat, including:

- **Accessibility/Opportunity.**
- **Asset Financial Value.**
- **Asset Security Value.**
- **Attacker Sophistication/knowledge.**
- **Coordination/synchronisation.** This might involve the amount of co-ordination or synchronisation required to exploit the vulnerability, such as related to the co-

ordination of bots in a Distributed Denial-of-Service attack.

- **Knowledge.** This might relate to the amount of knowledge that an intruder requires for exploit the vulnerability.
- **Progression.** This might relate to the setup of backdoors and Trojans.
- **Threat characteristics.**

The following defines some relevant threats.

Hardware misuse

There are many forms of hardware misuse including:

- **Logical scavenging.** This involves scavenging through discarded media (Figure 1.2).
- **Eavesdropping.** This involves intercepting communications (Figure 1.2).
- **Interference.** This involves the actual interference of communications, such as in jamming communications, or modifying it in some way (Figure 1.3).
- **Physical attacks.** This involves an actual physical attack on the hardware (Figure 1.3).
- **Physical removal.** This involves the actual physical removal of hardware.

Andy Jones and Prof Andrew Blyth from the University in Glamorgan, in 2008/2009, found that an analysis of 300 discarded disk showed that 34% of the disks still had personal information, which included health care and banking information (including a 50 billion Euro currency exchange). One of the disks even contained details of the test launch procedures for the THAAD (Terminal High Altitude Area Defence) ground-to-air missile defence system, which is a system designed to destroy long-range intercontinental missiles. As much as possible organisations should have a predefined plan on how they discard of their equipment, especially for electronic media (normally USB sticks) and for hard disks. The term used in the US for logical scavenging is dumpster diving.

Physical attacks and removal can cause major problems with security, and thus organisations require a strong policy on protecting hardware, including padlocking equipment within locked cabinets, within restricted zones.

Environmental risks

Computer equipment is fairly sensitive to changes in environmental conditions, especially related to temperature and humidity changes. This could involve risks related to ventilation, humidity control, and system cooling. Computer equipment can also be affected by power supply changes, including power spikes and surges, power blackouts, and reduced power levels (sags and brownouts). Often these environmental risks are caused by operating conditions (especially the weather and local supplies) or human/process failure, but may be part of a computer environment attack from an intruder.

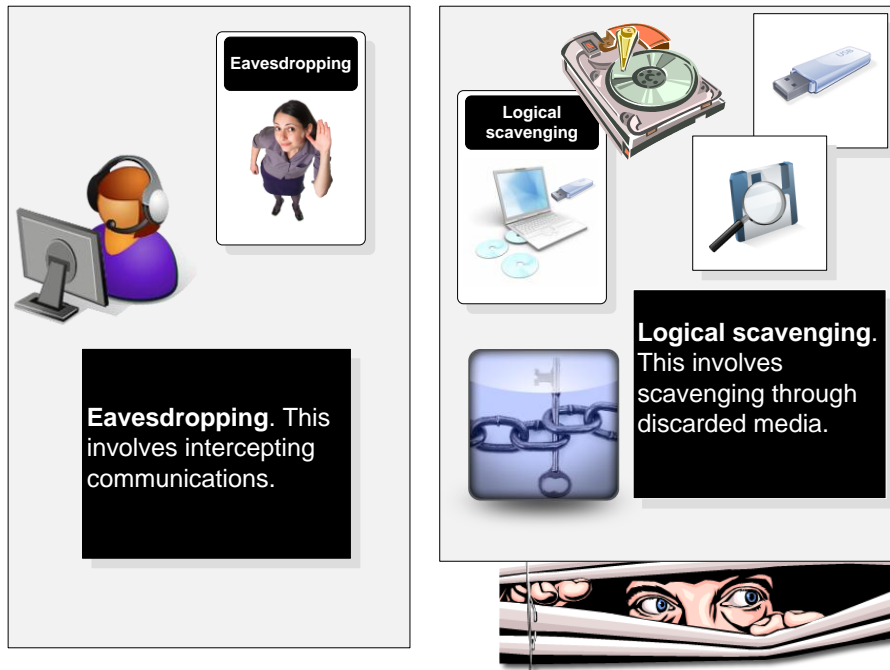


Figure 1.2 Eavesdropping and logical scavenging

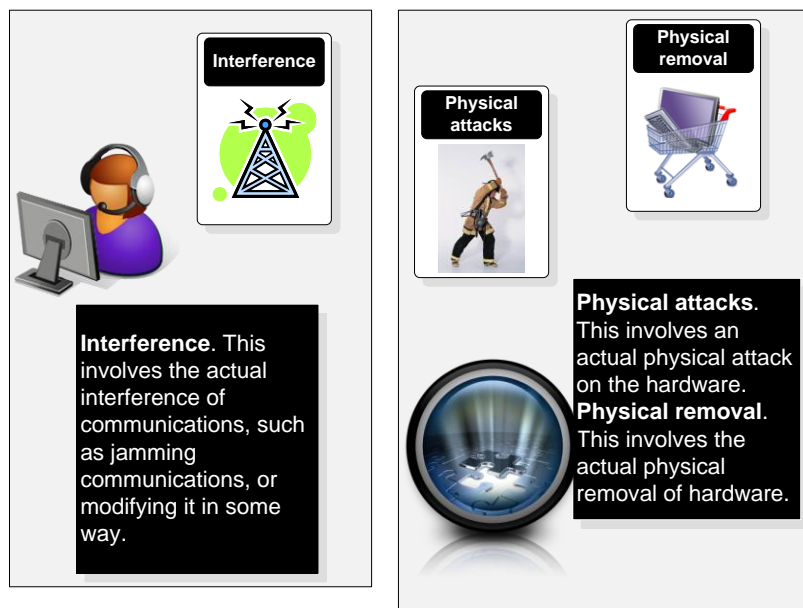


Figure 1.3 Interference and physical attacks

External misuse

External misuse involves some form external interference with internal users within an organization, normally involving some form of social engineering attack. This might include:

- **Visual spying.** This is the actual physical viewing a user's activities, such as their keystrokes or mouse clicks (Figure 1.4).
- **Misrepresentation.** This involves the actual deception of users and system operators, and is a form of social engineering (Figure 1.4).

Often misrepresentation occurs through a telephone call, a meeting in-person, an observation of the person, or through email. Typical methods include gaining a background knowledge of an individual, such as through dumpster diving or on-line searching (such as Facebook), and then focusing on them to mine them for information. For example if an intruder targets a certain user, a search on the Internet might give them their mother's maiden name. They could then phone the *target*, saying they were their mobile phone company, and saying there was a problem with their direct debit, and thus gain the bank account of the user. Armed with this, and a knowledge of the target, could allow them to gain access to privileged information.

Typical method used to mitigate against social engineering attacks are to:

- Improve training for users.
- Define strict policies about the kind of information that is allowed to be given over communications channels. For example a bank will never ask for a full password, and will typically only ask for two or three characters from a passphrase.
- Enforce a system of badges and ID cards.

In many cases, it is people who are the weakest link in a security infrastructure, and not the actual technological design/implementation.

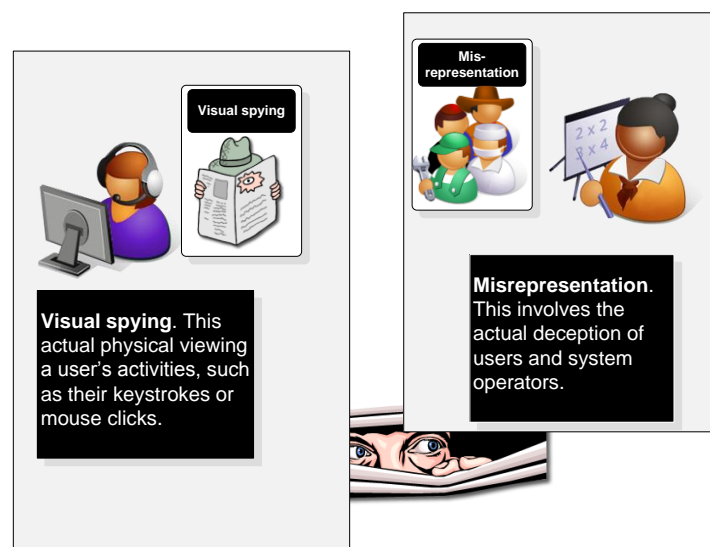


Figure 1.4 Visual spying and misrepresentation

Pests

Pest are malicious pieces of software which have been created for the purpose of exploiting a host machine. These types of threat are often detected using a signature scanning program, such as for a virus scanner or an intrusion detection system. Examples include:

- **Trojan horses.** This involves users running programs which look valid, but install an illicit program which will typically do damage to the host (Figure 1.5).

- **Logic bombs.** This involves the installation of a program which will trigger at some time in the future based at a given or pre-defined event, such as for a certain user logging-in, or a message from a remote system (Figure 1.5).
- **Malevolent worms.** This involves a worm program which mutates in a given way which will eventually reduce the quality-of-service of the network system, such as using up CPU resources, or taking up network band-width (Figure 1.6).
- **Viruses.** This involves malicious programs attaching themselves and which self replicate themselves (Figure 1.6).

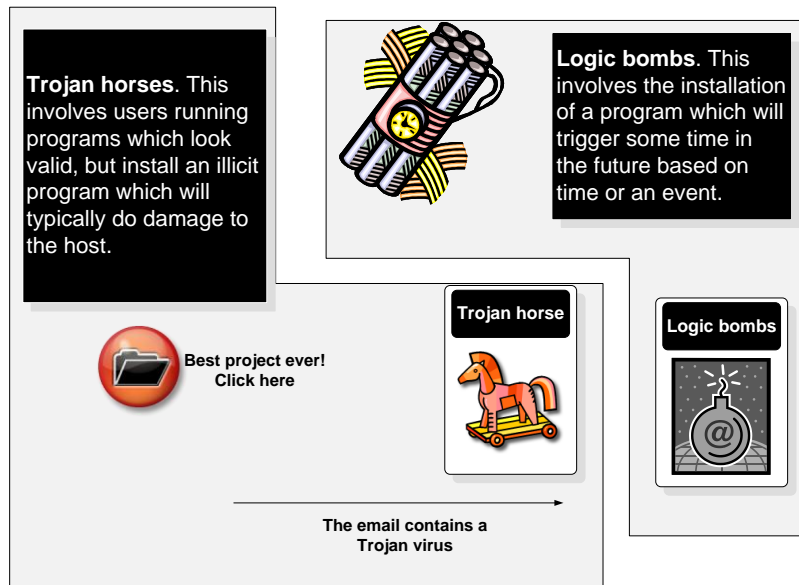


Figure 1.5 Trojan horses and logic bombs

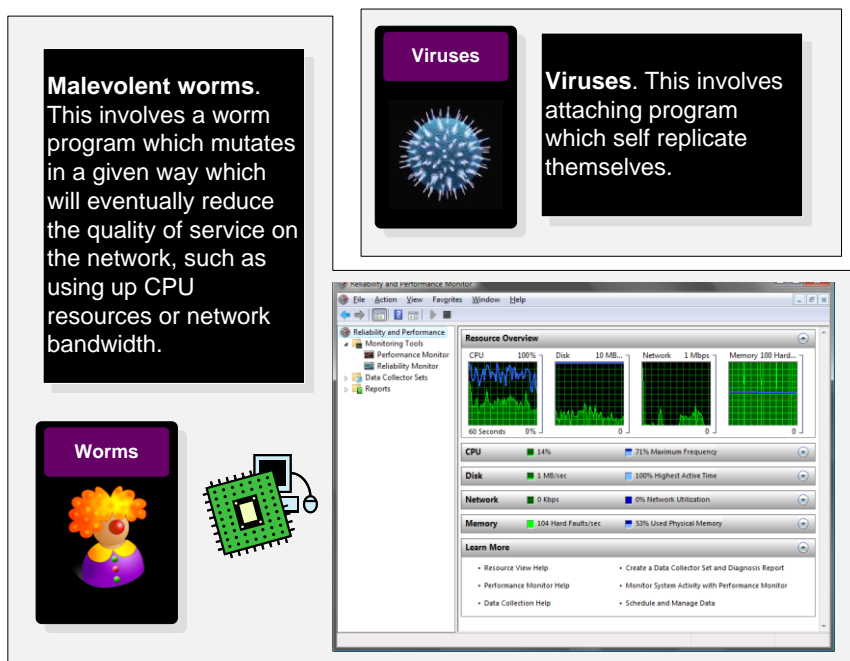


Figure 1.6 Malevolent worms and viruses

Active misuse

An active misuse threat normally involves some form of proactive method, such as for:

- **Incremental attack.** This involves damaging a system using an incremental approach.
- **Denial-of-service (DoS).** This involves attacking a host with continual requests for services, which eventually reduces its performance (Figure 1.7). With a distributed DoS (DDoS), remote hosts make continual accesses to hosts, such as continually requesting a connection (SYN flood).
- **Active attack.** This typically involves entering incorrect data with the intention to do damage to the system (Figure 1.7). An example threat with this involves buffer overflows, where the intruder intentionally enters data into a program, which overfills the memory area reserved for input variables. This overflowing can affect other data, or can even affect the actual program code. The major problem involves older programming languages such as C and C++, which allocates defined areas for their variables, and does not typically check for the size of the data input. Modern languages such as Java and C# use dynamic variable sizes, where more checking is done on the input data, in which one variable cannot affect another one.

Passive misuse

Passive misuse is a threat where an intruder attacks a system using methods that look fairly passive in their scope, but where they try to exploit a weakness in protocols/methods. This includes:

- **Browsing.** This issues random and/or selective searches for information.
- **Interference/aggression.** This involves exploiting database weaknesses using inferences (Figure 1.8). In the example in Figure 1.8, the user is not allowed to see the individual marks of students, but is allowed to see the average of a number of students. It can be seen that for three students, and three queries for an average mark of each of each group of two students, results in the inference of their individual mark. Inference is difficult to defend against, as there are an almost infinite number of ways that someone may view data, and the only way to overcome it is to make sure that the queries allowed on a system is limited to valid ones.
- **Covert channels.** This involves hiding data in valid communications, such as with network traffic or by passing information through the creation of timed events (Figure 1.9). Many of the modern Internet protocols were created at a time where security was not a major factor. They are thus often flawed, and there are many fields where data can be hidden. For example, the IP header contains a TTL field, which is ignored by more hosts, and can be used to hide data.

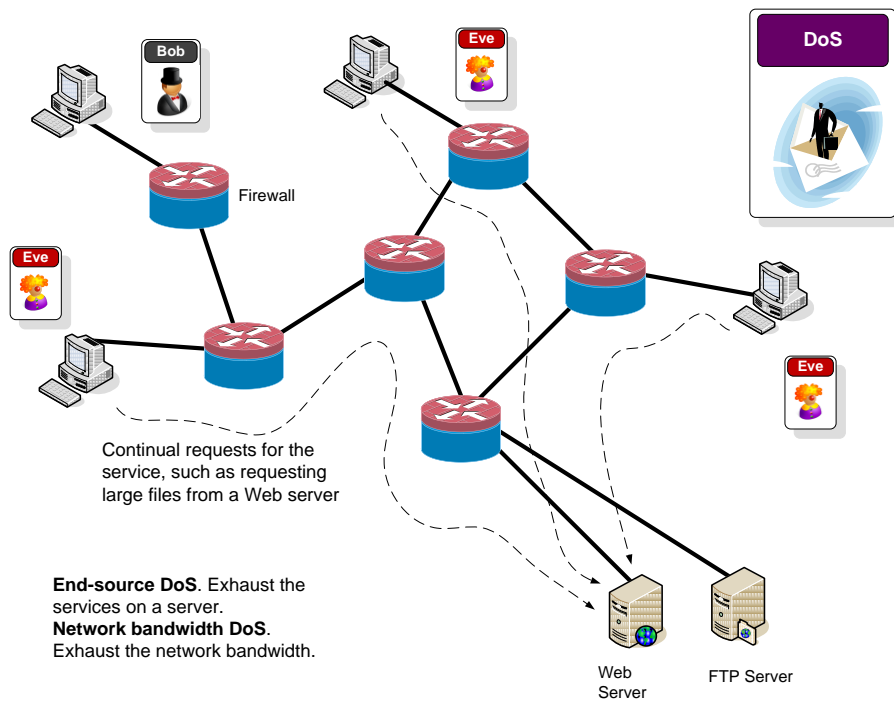


Figure 1.7 DDoS

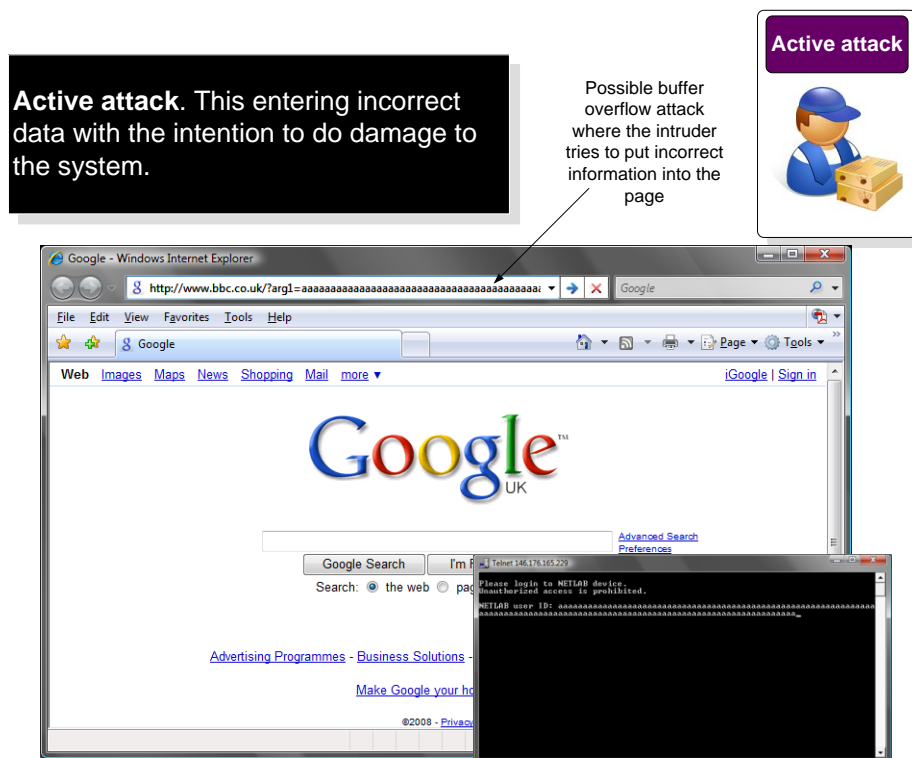


Figure 1.8 Active attack

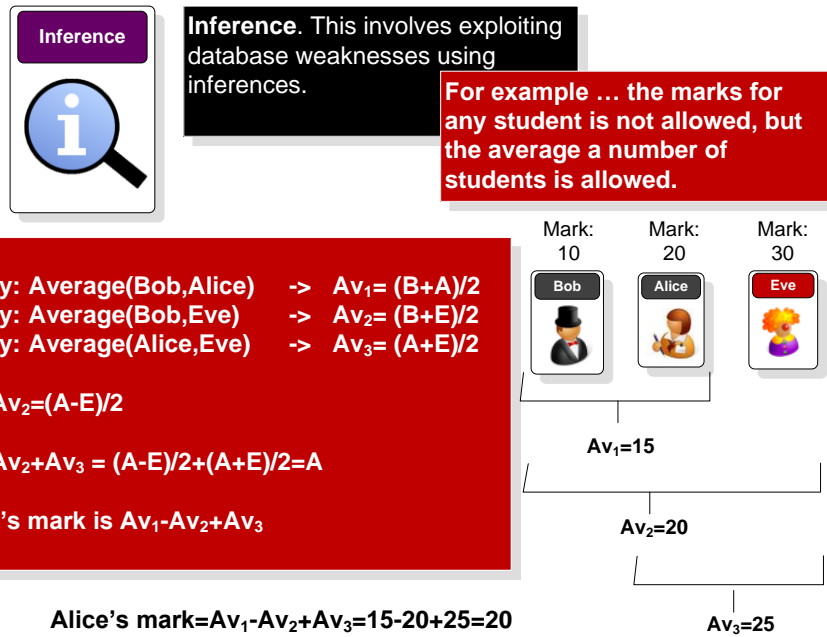


Figure 1.9 Inference

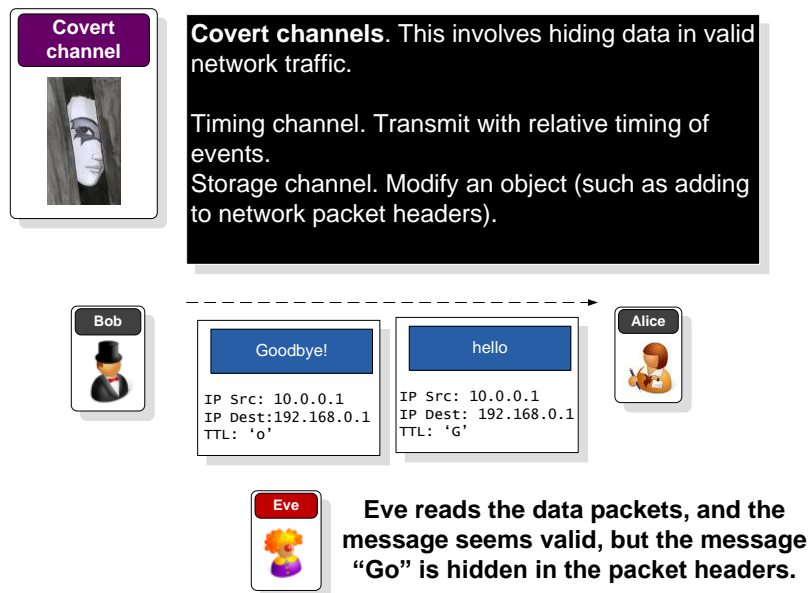


Figure 1.10 Covert channels

Masquerading/spoofing

These types of threats normally involve some form of impersonation of a device or a user, or involve obfuscation of the source of the threat. Examples of this include:

- **Impersonation.** This involves the impersonation of a user/device (Figure 1.11).
- **Spoofing.** This involves the spoofing of devices (Figure 1.11).
- **Piggy back attacks.** This involves adding data onto valid data packets (Figure 1.12).
- **Network weaving.** This involves confusing the system as to the whereabouts of a device, or confusing the routing of data. A typical technique used for this is to hide operations behind NAT (Network Address Translation), or to use a proxy

agent to perform an action (Figure 1.12).

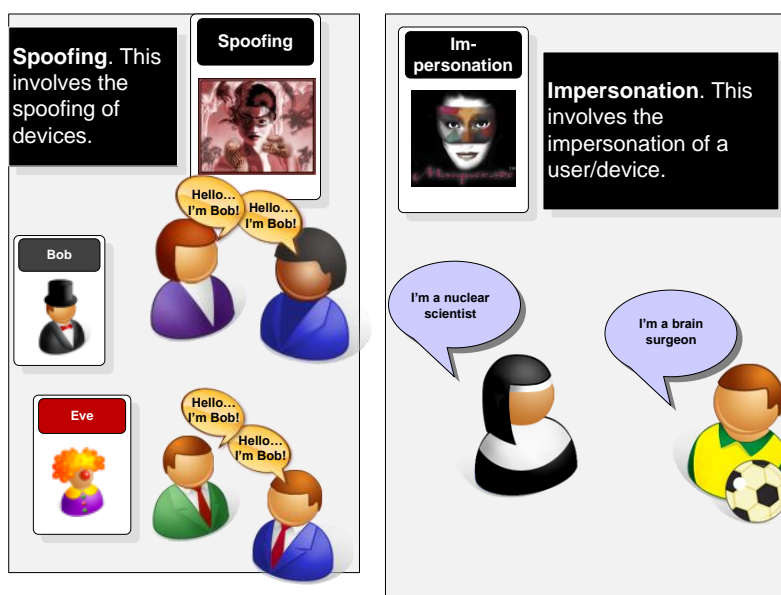


Figure 1.11 Spoofing and impersonation

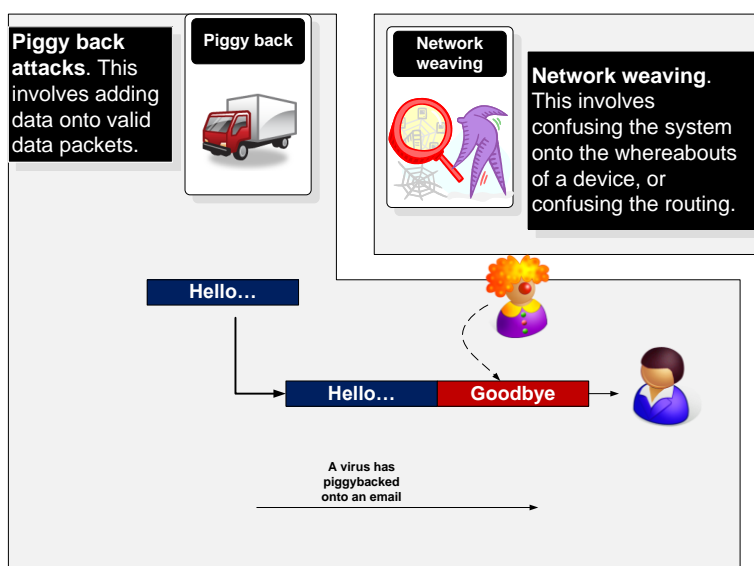


Figure 1.12 Piggy back and network weaving

Bypasses

These type of bypasses include:

- **Trap door impersonation.** This involves the creation of pages or login screens which look valid, but are used to gain information from a user, such as for their bank details, or login password (Figure 1.13).
- **Authorization attacks.** This involves trying to gain access to a higher level of authorization than is valid for the user, such as with password attacks.

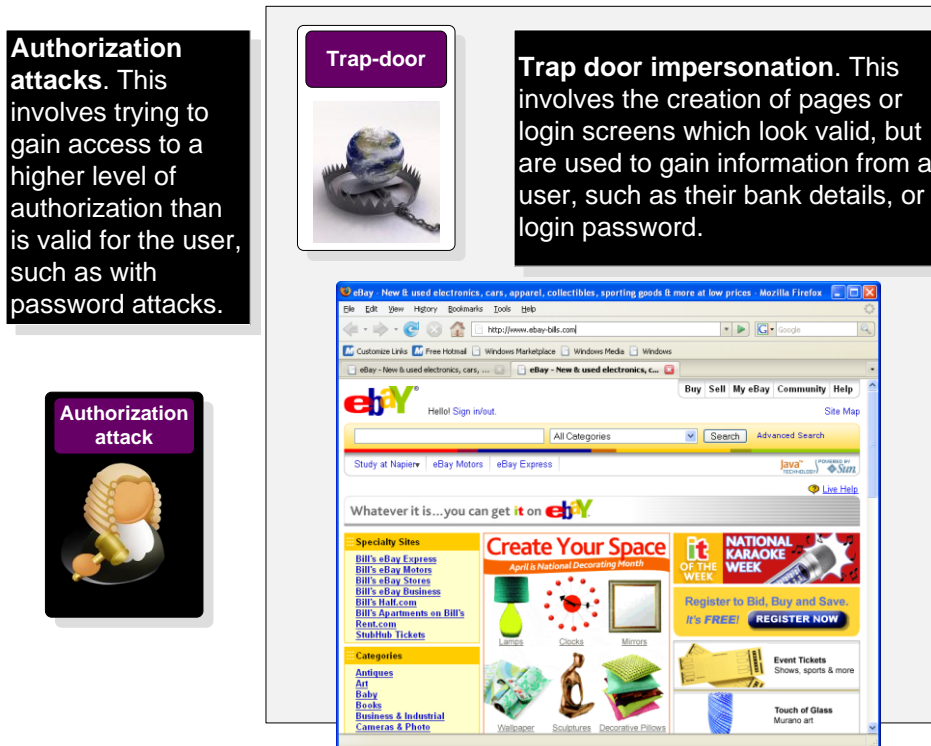


Figure 1.13 Authorization attacks and trap door impersonation

1.5 Service-oriented infrastructures

The first few generations of computers were based around centralized systems, with terminals which accessed central mainframe, which ran processes for them. With the introduction of the IBM PC, more software was installed locally on machines, and centralized services were used for key applications such as for network storage and email. As networks and, often, servers, were initially unreliable, most PCs were setup to run most of their applications locally. As the reliability and spread of networks has increased, there has been a tendency to move applications and services away from the local machine to distributing them around a network. This makes it easy to update software without requiring the local installation of updates. The main benefit, though, is that distributed services provide a fairly robust infrastructure, and it allows for the processing to move from the localized machine to networked services, which increases the scope of the devices that can access applications. For example, a mobile phone typically has limited processing and storage facilities, but with a service-oriented architecture, it is possible for them to access large scale processing facilities, along with access to mass storage.

Figure 1.14 shows an example of creating a service-oriented architecture, where services can either be local, domain-, remote- or cloud-based. Firewalls can block the access to services, including host-based, local and domain firewalls, each of which can block services at their different levels. Typical services include proxy services, directory services, Web services, file transfer services, email services, and so on. Generally, to aid robustness, there is a move towards having multiple service points, so that the infrastructure can cope with network outages.

Servers are normally used to provide networked services, as these are typically

built in order to be robust for their power supplies, their storage, their network connection, and so on. In the past, servers have often run a range of services, such as email and file services, but generally most organizational servers are now setup to focus on providing one type of service, thus distributing services across a range of servers.

A service connection has a client connecting to a service. The definition of the service is normally found using an IP address to find the server, and a TCP/UDP port to define the connection point on the server in which to connect to. The connection thus becomes:

IP(host)Port(host) -> IP(service)Port(service)

Normally the IP(service) and Port(service) are well know so that hosts can access them. On the Internet, each of these mapping must be unique, thus Port(host) requires to change in order to make the mapping unique (Figure 1.15). Typical service ports are 80 (HTTP), 21 (FTP), 23 (Telnet), 25 (SMTP), 110 (POP3), but increasing the port used is 80, and the services are tunnelled through this port as firewalls tend not to block HTTP access. The services can also run locally on a host, and can access their own services in the same way as they are done remotely.

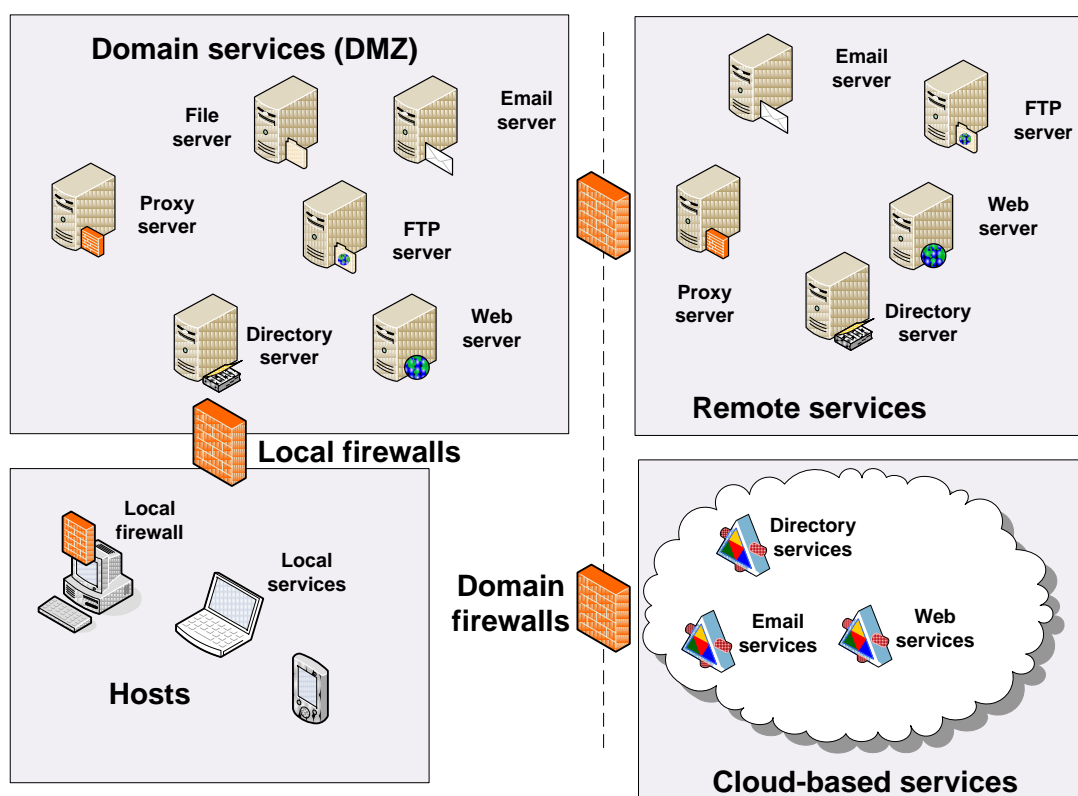


Figure 1.14 Security policy definition, implementation and verification

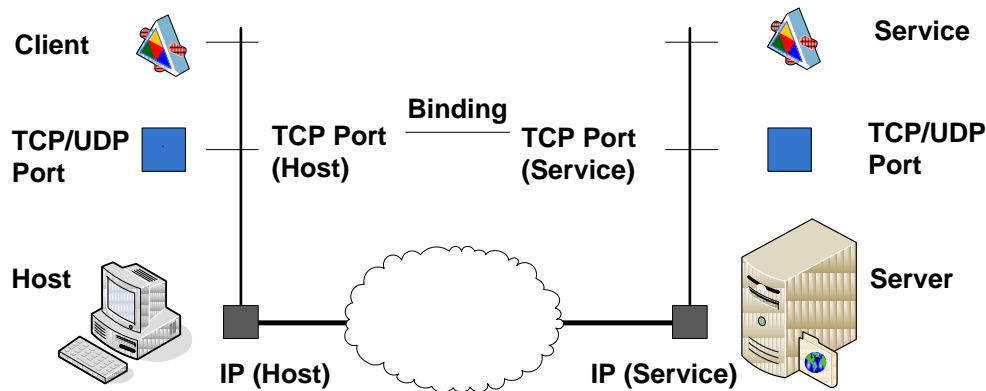


Figure 1.15 Port binding for services

http://buchananweb.co.uk/adv_security_and_network_forensics/dotnetclient/dotnetclient.htm

1.6 Security policies

Often military analogies are used in security, and the equivalent of having internal and external attacks is equivalent to fighting an external army at a defensive line, while also being attacked from behind a defensive line. In networked systems, with security, there is no attack, as the goal is purely to defend. Overall the key factor in any security system is that the aims and objectives of the organization must map directly onto the implementation of the security policy.

A networked system is a complex entity, composed of many elements, such as hardware devices, operating systems, application programs, file systems, and users. In a highly secure system the overall system should be also be broken down into entities, each of which have security policies for individual users, and also for groups of users. For example, a networked printer should have a policy which restricts access to individual users, and also groups of users. Often in a hierarchal network the entities should inherit security policies from the hierarchy above them. For example with file directories, the subdirectories will often inherit their security policies from the level above, unless otherwise stated. This type of approach typically simplifies the security policy for the overall system.

Often the key elements of any security policy are to:

- **Deter.** This is where the system is designed and implemented in order to initially deter intruders from attacking the system in the first place.
- **Log.** This is a key element in modern systems which require some form of logging system. It is important that the data that is logged does not breach any civil liberties, and is in a form which can be used to enhance the future security of the system.
- **Detect.** This is where detection agents are placed within the network to detect intrusions, and has some method of tracing the events that occurred in an intrusion, so that it can be used either in a forensic computing investigation, and/or to overcome a future intrusion. Organisations often have many reasons for detect-

ing network traffic.

- **Protect.** This is where policies are created which protect systems, users and data against attack, and this potential damage. A key element of this is to protect them against accidental damage, as accidental damage is often more prevalent than non-accidental damage.
- **React.** This is where a policy is defined which reacts to intrusions, and defines ways to overcome them in the future. Often organisations do not have formal policies for this type of activity, and often rely on *ad-hoc* arrangement, where the method of reacting to a security breach is created after the event.
- **Recover.** This is where policies are defined to overcome any system damage, whether it is actual physical damage, the abuse of users; or the damage to data.
- **Audit/verify.** It is important that the security policy allows for auditing and for the verification that it achieves its requirements.

Security, typically, focuses on the detection, protection and recovery from an attack, whereas forensic computing focuses on not just the malicious activity, but also in capturing the after-effects of an attack, as well as for non-malicious behaviour. A key component is that security tends to focus on the assumption of guilt within attacks, whereas forensic computing must focus on both malicious and non-malicious data so that a fair case can be presented for an investigation. This a forensics policy will typically focus on the detection of events, and the associated procedures. The key focus for the forensic computing parts of this module will be on:

- **Log.** This will define the data that is recorded, and, possibly, the rights of the data to be viewed by certain individuals within an organisation.
- **Detect.** This would be the activities which were to be detected for forensic investigations.
- **React.** This is where a policy is defined which reacts to malicious activities, and, especially in a forensic computing investigation, the procedures involved.
- **Audit/verify.** It is important that the forensics policy allows for auditing and verification that it achieves its requirements.

1.7 Defining the policy

A key element of security is to have a policy which is defined at the highest-level in the organisation, and which is well-known to the employees in the organisation. Also, if there is public access to the network, they should also be informed as to the security restrictions placed on the network. Figure 1.16 shows a transparent and auditable system where the policy is defined at the highest level, and includes: the aims and objectives of the organisation; the legal moral and social responsibilities of the organisation; and the technical feasibility of the policy. These are then decided upon, and a policy is implemented by technical staff. A key feature is that this policy should be audited in some way, and also verified that it achieves the policy requirements.

There are possibly many different types of network/user activity that should be detected and which could breach the aims and objectives of the organisation, or which breach the social, moral and legal responsibilities of the organisation. Exam-

ples of classifications for attacks might be:

- Attempted administrator privilege gain.
- Attempted user privilege gain.
- Denial-of-service.
- ICMP event.
- Information leak.
- Network scan.
- Non-standard protocol.
- Policy violation.
- Suspicious string detection.
- Suspicious login.
- Trojan activity.
- Unusual client-port connection.
- Web application attack.

There are many examples of network traffic/user activity that might be monitored with an intrusion detection system (IDS). It can be seen that it is not just treats to the network, but also activities that might be wasteful in resources, or which breach social and moral rules. It can often be just as embarrassing for a user in an organisation to be involved in an immoral activity, than it is to have a network intrusion. Thus applications such as peer-to-peer file sharing, such as Kazaa, should be avoided in organisations, as they have many copyright issues. Along with this audio and video streaming, such as from news sites, may be wasteful on bandwidth, and, if this type of traffic was great enough, it might swamp traffic which is important for the organisation.

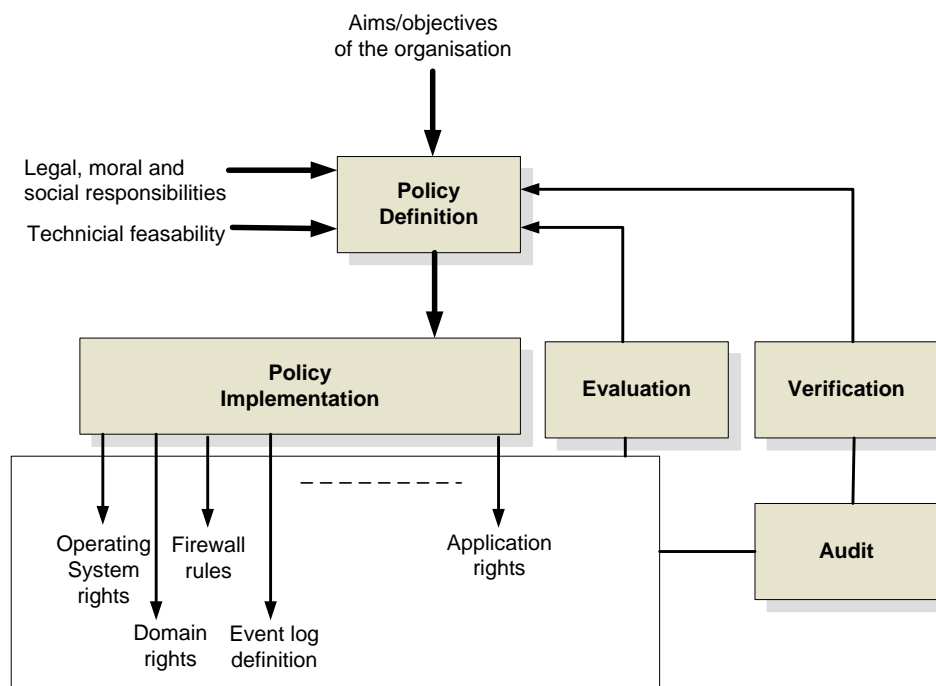


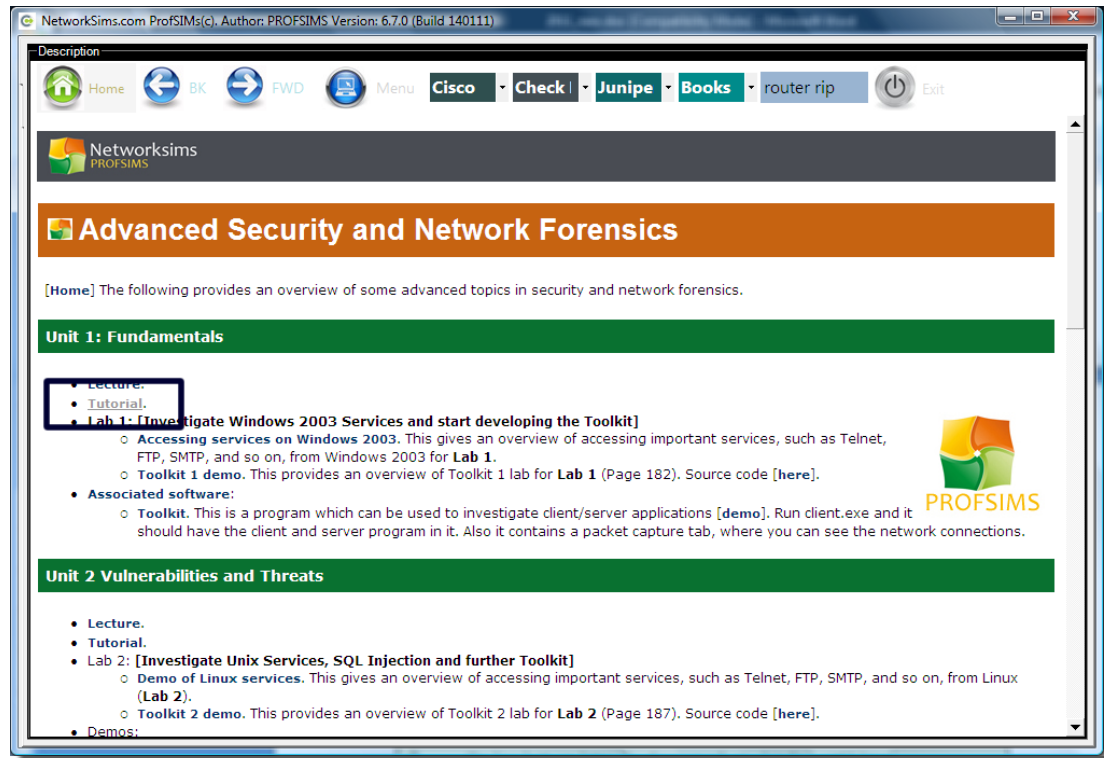
Figure 1.16 Security policy definition, implementation and verification

1.8 Tutorial

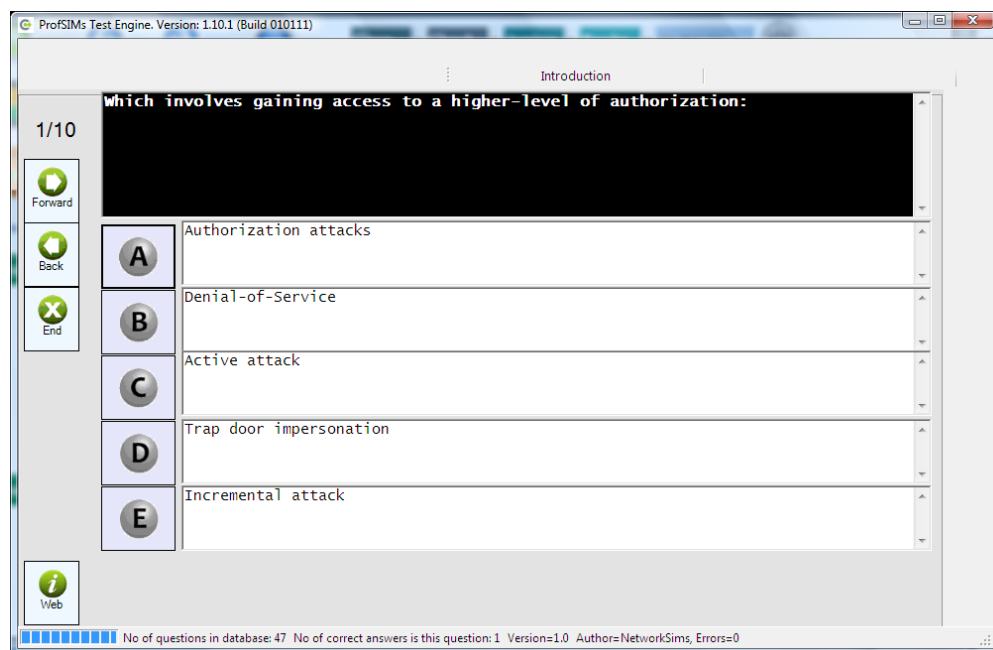
An outline tutorial is at:

On-line tutorial: <http://buchananweb.co.uk/adv01.html>

The main tutorial is in NetworkSims ProfSIMs at:



And then select the Tutorial to give:



2 Threat Analysis



On-line lecture: <http://buchananweb.co.uk/adv/unit02.html>

2.1 Objectives

The key objectives of this unit are to:

- Understand the basis steps that an intruder might undertake in an intrusion.
- Provide a background in the usage of vulnerability scanning.
- Outline key current threats, and their operation.
- Provide practical skills in vulnerability analysis.

2.2 Introduction

The previous unit outlined some of the key classifications of threats, while this one focuses on how to assess vulnerabilities can be assessed. A key factor in this is the use of evaluation tools such as Nmap and Nessus, which contain a number of tests which evaluate potential vulnerabilities. Organisations such as US-CERT (US Computer Emergency Response Team) and CVE (Common Vulnerabilities and Exposures) also maintain databases of vulnerabilities, and their current status, which are useful in keeping track of current threats, and methodologies which can be used to overcome them.

2.3 Intruder detection

It is important to know the main stages of an intrusion, so that they can be detected at an early phase, and to overcome them before they can do any damage. Typically an intrusion goes through alert phases from yellow, which shows some signs of a potential threat, to red, which involves the potential stealing of data or some form of abuse. The main phases are defined in Figure 2.1.

Often it takes some time for an intruder to profit from their activities, and it is important to put in as many obstacles as possible to slow down their activity. The slower the intrusion, the more chance there is in detecting the activities, and thus in thwarting them. Figure 2.1 shows a typical sequence of intrusion, which goes from a yellow alert (on the outside reconnaissance) to a red alert (for the profit phase).

Initially an intruder might gain information from outside the network, such as determining network addresses, or domain names. There are, unfortunately, many databases which contain this type of information, as the Internet is a global network, and organisations must register their systems for network addresses and domain names. Once gained, the intruder could move into an internal reconnaissance phase, where more specific information could be gained, such as determining the location of firewalls, subnetworks, network layouts, host/server locations, and so on. It is thus important that this type of activity is detected, as it is typically a sign of some form of future intrusion. Key features could be things such as:

- A scan of network addresses for a range of hosts on a given subnetwork (ping sweep).
- A scan of open TCP ports for a range of hosts on a given subnetwork (port scan).
- A scan of a specific TCP port for a range of hosts on a given subnetwork (port sweep).
- An interrogation of the configuration of network devices.
- Accessing systems configuration files, such as ones which contain user names and passwords.

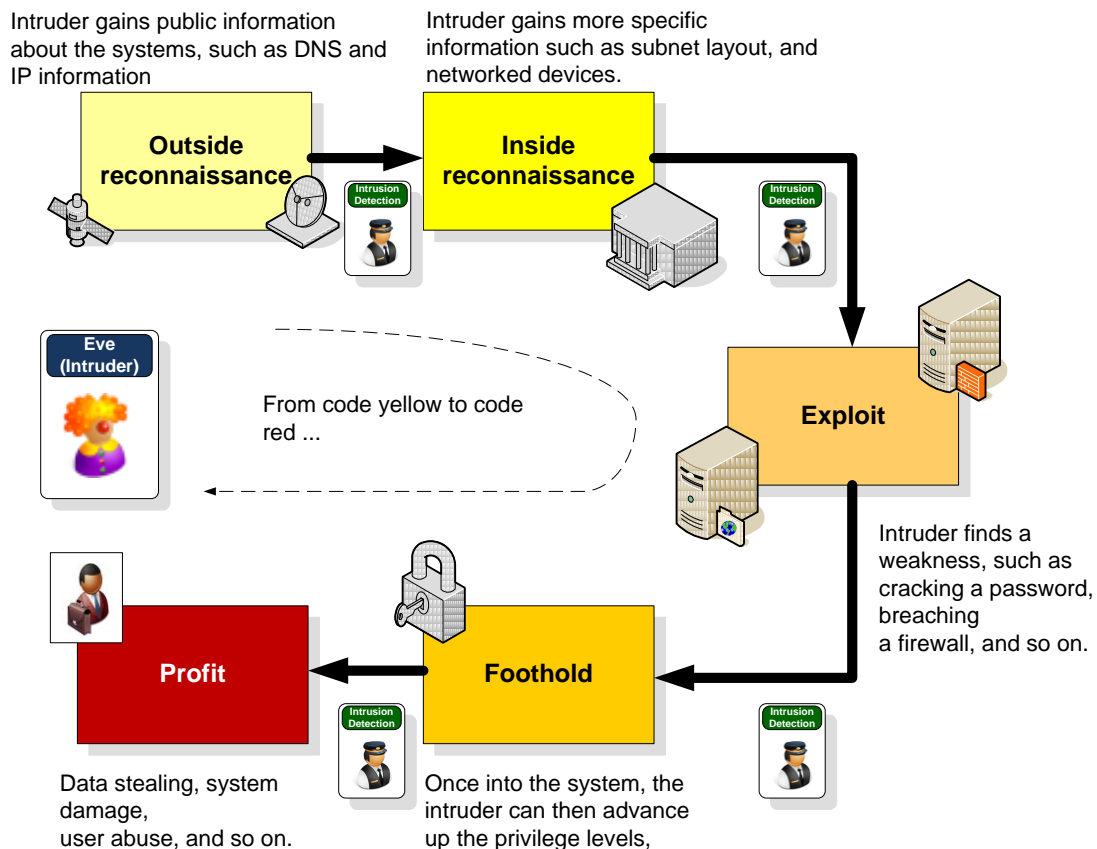


Figure 2.1 Intrusion pattern

Once the intruder has managed to gain information from the internal network, they may then use this information to gain a foothold, from which they can exploit. Example of this may be:

- Hijacking a user ID which has a default password (such as for the password of **default** or **password**), and then using this to move up the levels of privilege on a system. Often the administrator has the highest privileges on the system, but is normally secured with a strong password. An intruder, though, who gains a foothold on the system, normally through a lower-level account, could then glean more information, and move up through the privilege hierarchy.
- Using software flaws to exploit weaknesses, and gain a higher-level privilege to the system. Software flaws can be intentional, where the writer has created an exploit which can be used to cause damage. This might include a back-door exploit, where an intruder could connect into a host through some form of network

connection, or through a virus or worm. A non-intentional one is where the software has some form of flaw which was unintentional, but which can be used by an intruder. Typical types of non-intentional flaws are: **validation flaws** (where the program does not check for correct input data); **domain flaws** (where data can leak from one program to another); **identification flaws** (where the program does not properly identify the requester); and **logical problems** (where the program does not operate correctly with certain logical steps).

One problem with IDS systems is that they cannot investigate encrypted content, which is setup through an encryption tunnel. These tunnels are often used to keep data private when using public networks. It is thus important that the usage of encryption tunnels on corporate network should be carefully used, as threats within them may not be picked-up, and virus/worm scanners and IDS systems will not be able to decrypt the traffic.

Sweeps

One activity which typically indicates a potential future security breach is sweeping activities. This typically involves: TCP/UDP sweeps (as illustrated in Figure 2.2); ping sweeps (as illustrated in Figure 2.3), OS identification, and account scans (Figure 2.4).

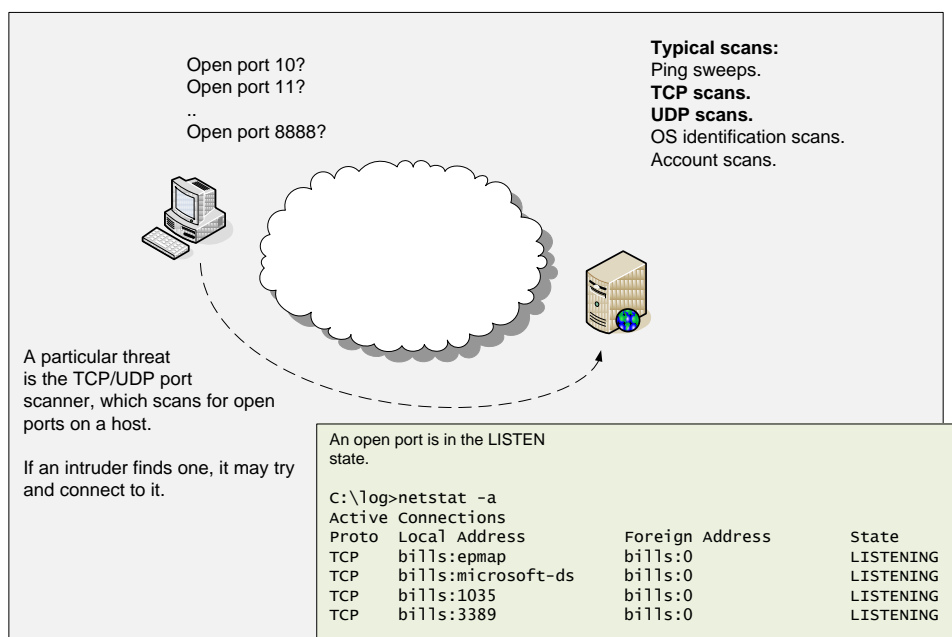


Figure 2.2 TCP/UDP port sweeps

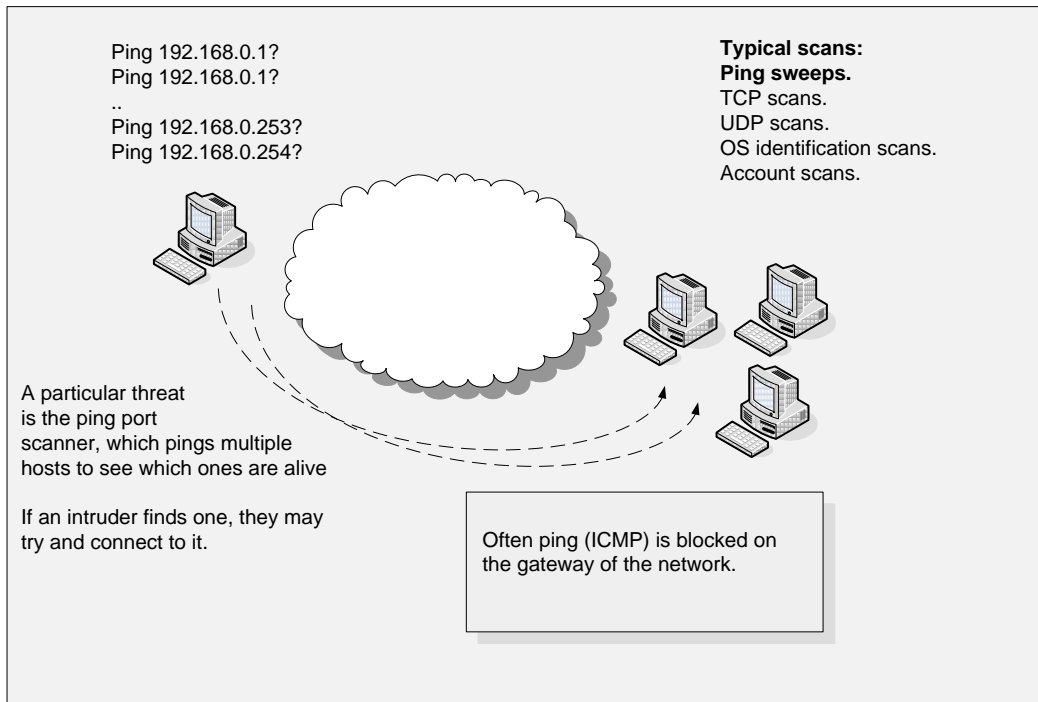


Figure 2.3 Ping sweeps

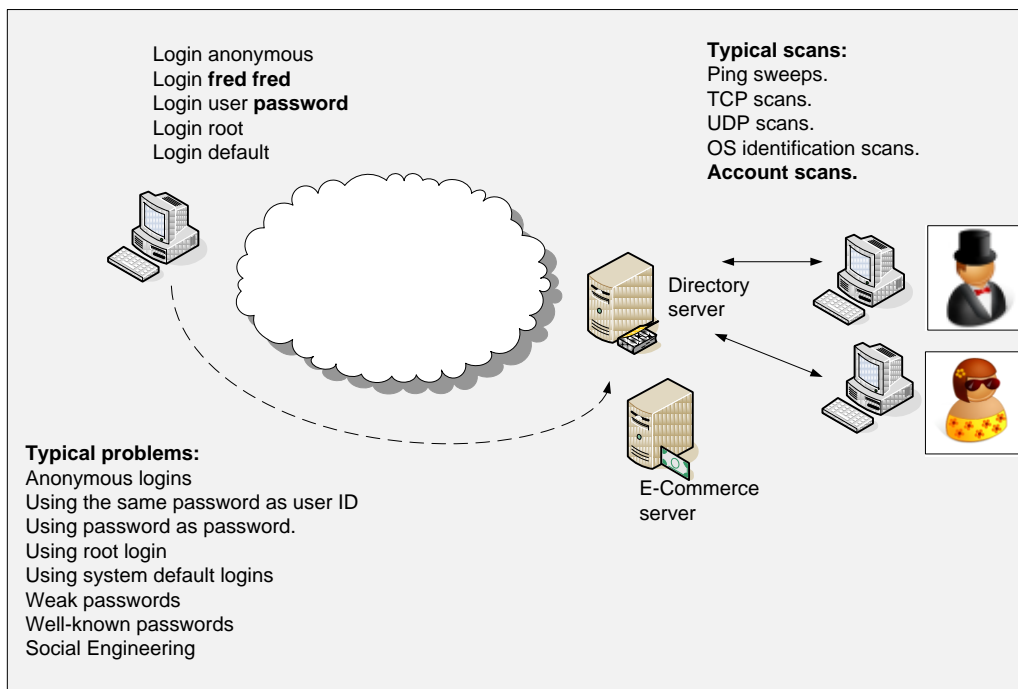


Figure 2.4 Account sweeps

2.4 Vulnerability analysis

US-CERT provides support against cyber attacks and interacts with a wide range of partners in order to disseminate information on cyber security information to the public. As part of this US-CERT maintains a database of vulnerabilities (CERT, 2009a), which define unique IDs and names to each vulnerability. Recent examples include:

- VU#515749. Microsoft Internet Explorer CSS style element vulnerability
- VU#723308. TCP may keep its offered receive window closed.
- VU#545228. Microsoft Office Web Components Spreadsheet ActiveX control vulnerability indefinitely (RFC 1122).
- VU#180513. Microsoft Video ActiveX control stack buffer overflow

For each vulnerability, CERT then defines an overview, a description, the impact, a solution, and also defines the vendors which are affect. For example (CERT, 2009b):

VU#120541: SSL and TLS protocols renegotiation vulnerability

Overview
A vulnerability exists in SSL and TLS protocols that may allow attackers to execute an arbitrary HTTP transaction.

I. Description
The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols are commonly used to provide authentication, encryption, integrity, and non-repudiation services to network applications such as HTTP, IMAP, POP3, LDAP. A vulnerability in the way SSL and TLS protocols allow renegotiation requests may allow an attacker to inject plaintext into an application protocol stream. This could result in a situation where the attacker may be able to issue commands to the server that appear to be coming from a legitimate source. According to the Network Working Group:

The server treats the client's initial TLS handshake as a renegotiation and thus believes that the initial data transmitted by the attacker is from the same entity as the subsequent client data.

This issue affects SSL version 3.0 and newer and TLS version 1.0 and newer.

II. Impact
A remote, unauthenticated attacker may be able to inject an arbitrary amount of chosen plaintext into the beginning of the application protocol stream. This could allow and attacker to issue HTTP requests, or take action impersonating the user, among other consequences.

III. Solution
Users should contact vendors for specific patch information.

Systems Affected

Vendor	Status	Date Notified	Date Updated
3com Inc	Unknown	2009-11-05	2009-11-05
ACCESS	Unknown	2009-11-05	2009-11-05

NESSUS also maintain a database of vulnerabilities, and their vulnerability scanner

can be used to assess weaknesses within systems. Along with NESSUS, CVE maintains a dictionary of publicly known information security vulnerabilities and exposures, and aims to provide common identifiers enabling data exchange between differing vendors/tools. An example of a CVE-ID is (CVE, 2009):

CVE-2009-0076

Summary: Microsoft Internet Explorer 7, when XHTML strict mode is used, allows remote attackers to execute arbitrary code via the zoom style directive in conjunction with unspecified other directives in a malformed Cascading Style Sheets (CSS) stylesheet in a crafted HTML document, aka "CSS Memory Corruption Vulnerability."

Published: 02/10/2009

CVSS Severity: 9.3 (HIGH)

Vulnerability scanners

There are a number of vulnerability scanner which can be used for penetration testing. These include Nessus and Nmap, whereas tools such as hping can be used to craft network traffic for evaluations. Nessus uses a Web-based client with a server to scan for vulnerabilities (Figure 2.5). When defining the scan, a policy is created which defines the test to be undertaken. Figure 2.6 shows a sample TCP port scan. In can be seen in this case that the host has a number of open ports, including port 80 (www), port 123 (ntp) and 445 (cifs).

Nessus Demo Link:

http://buchananweb.co.uk/adv_security_and_network_forensics/nessus/nessus.htm

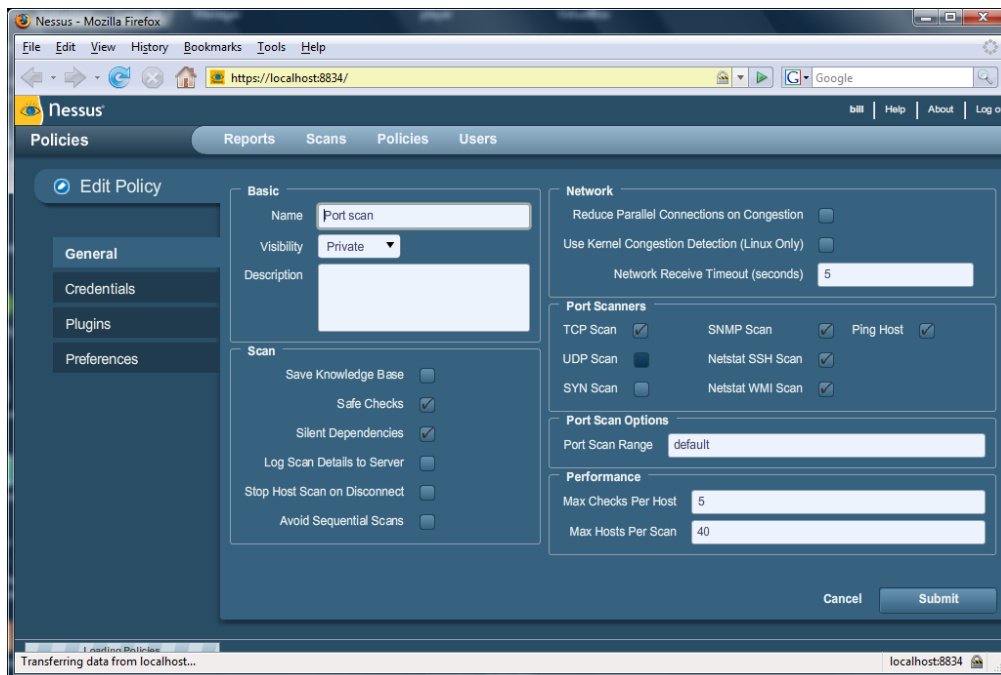


Figure 2.5 Nessus policy definition

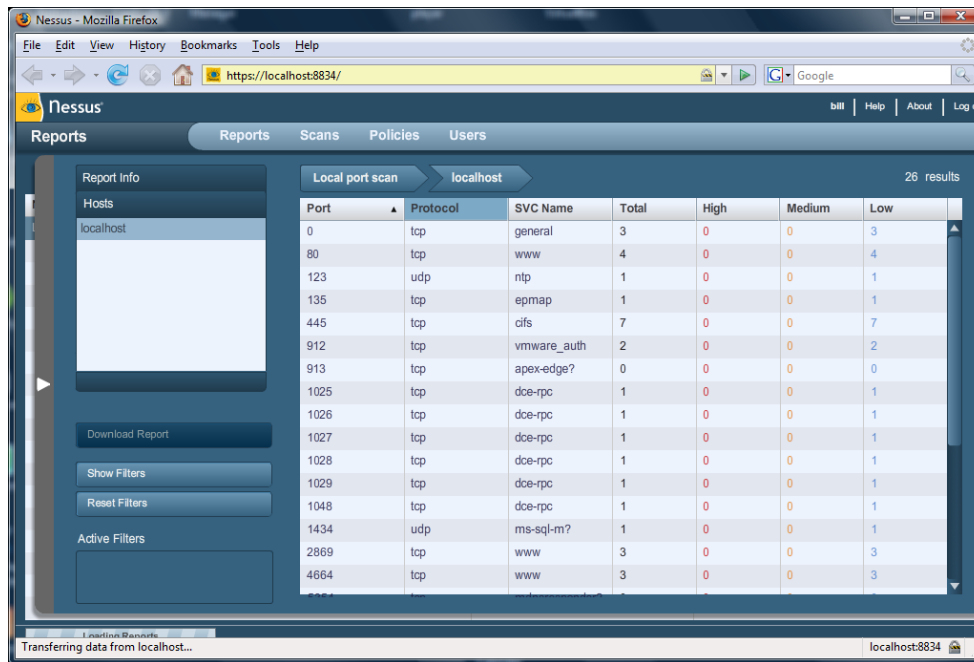


Figure 2.6 Nessus scan report

PORT SCANS. For port scans an intruder may scan certain hosts or every host on a subnet, to determine the ports which they have open, as certain ports could be used to gain a foothold on the host. Programs such as **nmap**, for example, can scan whole networks looking for open ports. A key objective of Snort is thus to detect this type of activity. Luckily Snort has a pre-processor rule for this, which acts before other rules. An example is:

```
sfportscan: proto { all } memcap { 10000000 } sense_level { low }
```

where the arguments might include:

- **proto.** This can be tcp, udp, icmp, ip or all, and are the types of protocol scans to be detected.
- **scan_type.** This can be portscan, portsweep, decoy_portscan, distributed_portscan or all, and defines the scan type to be detected.
- **sense_level.** This can be low, medium or high, and defines the sensitivity of the portscans. A low sense level detects response errors, such as ICMP unreachables. Medium sensitivity level detects portscans and filtered portscans (which are portscans that do not have any responses). High sensitivity level has a lower threshold than medium and has a longer time window to detect sweeps.
- **Memcap.** This defines the maximum memory size (in bytes) – this limits the possibility of buffer overflows.
- **Watch_Ip.** This defines the hosts that are to be detected.

To save to a file named portscan.log (scan.rule):

```
preprocessor sfportscan: proto { all } scan_type { all } \
    sense_level { low } logfile { portscan.log }
```

It is always important to understand the ports that are open on a computer, such as with running NMAP:

```
C:\> snort -c scan.rule -dev -i 3 -p -l c:\\bill -K ascii
Initializing Preprocessors!
Initialzing Plug-ins!
Parsing Rules file scan.rule
-----[Flow Config]-----
| Stats Interval: 0
| Hash Method: 2
| Memcap: 10485760
| Rows : 4096
| Overhead Bytes: 16388(%0.16)
-----
Portscan Detection Config:
  Detect Protocols: TCP UDP ICMP IP
  Detect Scan Type: portscan portsweep decoy_portscan distributed_portscan
  Sensitivity Level: Low
  Memcap (in bytes): 1048576
  Number of Nodes: 3869
  Logfile: c:\\bill/portscan.log

Tagged Packet Limit: 256
```

Then for a scan:

```
C:\> nmap -o -A 192.162.0.1
Starting Nmap 4.20 ( http://insecure.org ) at 2007-01-09 21:58 GMT Standard Time
Interesting ports on 192.162.0.1:
Not shown: 1695 closed ports
PORT      STATE SERVICE
80/tcp    open  http
8888/tcp  open  sun-answerbook
MAC Address: 00:0B:44:F5:33:D5 (The Linksys Group)
Nmap finished: 1 IP address (1 host up) scanned in 1.500 seconds
```

The resulting log then gives the trace of the port sweep and scan:

```
Time: 08/17-14:41:54.495296
event_ref: 0
192.162.0.3 -> 63.13.134.49 (portscan) TCP Portsweep
Priority Count: 5
Connection Count: 135
IP Count: 43
Scanned IP Range: 63.13.134.49:216.239.59.99
Port/Proto Count: 1
Port/Proto Range: 80:80

Time: 08/17-14:42:52.431092
event_ref: 0
192.162.0.3 -> 192.162.0.1 (portscan) TCP Portsweep
Priority Count: 5
Connection Count: 10
IP Count: 5
Scanned IP Range: 66.249.93.165:192.162.0.7
Port/Proto Count: 3
Port/Proto Range: 80:2869

Time: 08/17-14:42:52.434852
event_ref: 0
192.162.0.3 -> 192.162.0.1 (portscan) TCP Portscan
Priority Count: 5
Connection Count: 9
IP Count: 1
Scanner IP Range: 192.162.0.3:192.162.0.3
Port/Proto Count: 10
Port/Proto Range: 21:636
```

PING SCANS. With ping scans, the intruder tries to determine the hosts which are active on a network. An example of detecting a Windows ping sweep is:

```

alert icmp $EXTERNAL_NET any -> $HOME_NET any (
  msg:"ICMP PING Windows"; itype:8; content:"abcdefghijklmnop";
  depth:16; sid:999)

```

where an ICMP ping packet is detected with the standard contents of "abc....op". An example of the contents of a ping request is:

0000	00 0c 41 f5 23 d5 00 15	00 34 02 f0 08 00 45 00	..A.#... .4....E.
0010	00 3c 10 7c 00 00 80 01	a6 8f c0 a8 01 64 c0 a8	.<.d..
0020	01 01 08 00 60 55 04 00	e9 06 61 62 63 64 65 66`U.. ..abcdef
0030	67 68 69 6a 6b 6c 6d 6e	6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67	68 69	wabcdefg hi

And a ping reply:

0000	00 15 00 34 02 f0 00 0c	41 f5 23 d5 08 00 45 00	...4.... A.#...E.
0010	00 3c 10 7c 00 00 96 01	90 8f c0 a8 01 01 c0 a8	.<.
0020	01 64 00 00 68 55 04 00	e9 06 61 62 63 64 65 66	.d..hu.. ..abcdef
0030	67 68 69 6a 6b 6c 6d 6e	6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67	68 69	wabcdefg hi

Snort Demo Link:
http://buchananweb.co.uk/adv_security_and_network_forensics/ids01/ids01.htm

OS SCANS. For OS identification the intruder searches hosts for certain machines, which possibly have an OS weakness, such as searching for Windows 95 machines, as these tend to have FAT32 file systems which have very little security associated with them. For account scans, an intruder may scan the user ID's for weak passwords, where the tests are:

- **TSeq.** This is where SYN packets are sent, and the TCP sequence numbers are analysed.
- **T1.** This is a SYN packet with certain options (WNMTE) set is sent to an open TCP port.
- **T2.** This is a NULL packet with options (WNMTE) and is sent to an open TCP port.
- **T3.** This is a SYN,FIN,PSH,URG packet with options (WNMTE), and sent to an open TCP port.
- **T4.** This is an ACK packet with options (WNMTE) and is sent to an open TCP port.
- **T5.** This is a SYN packet with options (WNMTE) and is sent to a closed TCP port.
- **T6.** This is an ACK packet with options (WNMTE) and is sent to a closed TCP port.
- **T7.** This is a FIN,PSH,URG packet with options (WNMTE) and is sent to a closed TCP port.
- **PU.** This is a packet sent to a closed UDP port.

For example the following is a fingerprint from XP Professional:

```
TSeq(Class=RI%gcd=<8%SI=<2959A&>356%IPID=I)
T1 (DF=Y%W=FAF0|402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2 (Resp=N)
T3 (Resp=N)
T4 (DF=N%W=0%ACK=0%Flags=R%Ops=)
T5 (DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6 (DF=N%W=0%ACK=0%Flags=R%Ops=)
T7 (Resp=N)
PU (DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

where:

- **Resp:** defines whether the host responds. Y - for a response, and N - no response.
- **DF:** defines whether the host responds with a “Don’t Fragment” bit set in response. Y - DF was set, N - DF was not set.
- **W:** defines the acknowledgement sequence number response and is the Window advertisement size sent by the host. ACK 0 - ack zero, S - ack sequence number, S++ - ack sequence number + 1.
- **Flags:** this defines the flags set in response. S = SYN, A = ACK, R = RST, F = FIN, U = URG, P = PSH.
- **Ops:** this is the options set for the response. M - MSS, E - Echoed MSS, W - Window Scale, T - Timestamp, and N - No Option.

For example DF=Y%W=FAF0|402E%ACK=S++%Flags=AS%Ops=MNWNNT

defines that the “Don’t Fragment” bit is set, the Window size is set to FAF0 or 402E, the acknowledgement sequence number is set to one more than the requesting packet, the flags set to ACK/SYN, with Options of MNWNNT.

A result from a scan of a Windows 2003 server image gives:

```
Starting Nmap 5.10BETA1 ( http://nmap.org ) at 2009-12-29 16:26 GMT Standard
Time
Nmap scan report for 192.162.75.132
Host is up (0.00071s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
135/tcp   open  msrpc  Microsoft Windows RPC
MAC Address: 00:0C:29:0F:71:A3 (VMware)
Device type: general purpose
Running: Microsoft Windows 2003
OS details: Microsoft Windows Server 2003 SP1 or SP2
Network Distance: 1 hop
Service Info: OS: Windows

HOP RTT      ADDRESS
1   0.71 ms 192.162.75.132
```

2.5 Hping

Hping is a vulnerability tool which can be used to generate data packets. It can be used, for example, to generate SYN packets at regular intervals using the -S option:

```

napier@ubuntu:~$ sudo hping -S 192.162.75.132 -e eth0
[sudo] password for napier:
HPING 192.162.75.132 (eth0 192.162.75.132): S set, 40 headers + 4 data bytes
[main] memlockall(): Success
Warning: can't disable memory paging!
len=46 ip=192.162.75.132 ttl=128 id=2052 sport=0 flags=RA seq=0 win=0 rtt=69.3 ms
len=46 ip=192.162.75.132 ttl=128 id=2053 sport=0 flags=RA seq=1 win=0 rtt=0.5 ms
len=46 ip=192.162.75.132 ttl=128 id=2054 sport=0 flags=RA seq=2 win=0 rtt=2.9 ms
--- 192.162.75.132 hping statistic ---
7 packets transmitted, 7 packets received, 0% packet loss

```

which will use random TCP port to connect to. Listening on the eth0 interface gives:

```

14:03:05.859738 IP ubuntu.local.2714 > 192.162.75.132.0: Flags [S], seq
1222983093:1222983097, win 512, length 4
14:03:05.859975 IP 192.162.75.132.0 > ubuntu.local.2714: Flags [R.], seq 0, ack
1222983098, win 0, length 0
14:03:06.860566 IP ubuntu.local.2715 > 192.162.75.132.0: Flags [S], seq
1026211710:1026211714, win 512, length 4

```

Which shows that port 0 is used to connect to on the remote host. If a specific port is require, the `-P` option is used. For example on port 80:

```

napier@ubuntu:~$ sudo hping -S 192.162.75.132 -e eth0 -p 80
HPING 192.162.75.132 (eth0 192.162.75.132): S set, 40 headers + 4 data bytes
[main] memlockall(): Success
Warning: can't disable memory paging!
len=46 ip=192.162.75.132 ttl=128 id=2072 sport=80 flags=SA seq=0 win=64240 rtt=11.3 ms
len=46 ip=192.162.75.132 ttl=128 id=2073 sport=80 flags=SA seq=1 win=64240 rtt=0.5 ms
len=46 ip=192.162.75.132 ttl=128 id=2074 sport=80 flags=SA seq=2 win=64240 rtt=0.4 ms
--- 192.162.75.132 hping statistic ---
15 packets transmitted, 15 packets received, 0% packet loss
round-trip min/avg/max = 0.4/1.5/11.3 ms

```

which gives:

```

14:04:31.090418 IP ubuntu.local.2222 > 192.162.75.132.www: Flags [S], seq
56776272:56776276, win 512, length 4
14:04:31.092037 IP ubuntu.local.57490 > 192.162.75.2.domain: 34223+ PTR?
132.75.162.192.in-addr.arpa. (45)
14:04:31.093064 IP 192.162.75.132.www > ubuntu.local.2222: Flags [S.], seq 447090437,
ack 56776273, win 64240, options [mss 1460], length 0
14:04:31.093132 IP ubuntu.local.2222 > 192.162.75.132.www: Flags [R], seq 56776273,
win 0, length 0

```

Hping Demo:

http://buchananweb.co.uk/adv_security_and_network_forensics/hping/hping.htm

2.6 Botnets

One of the most worrying threats is Botnets, which are created with a master controller, and with number installed Bot agents (slaves), which create a Botnet. With this a Bot agent can be installed on a host, and then wait for control signals from a Bot master (Figure 2.7). A study of Torpig over 10 days in 2009 by the Department of Computer Science, University of California, Santa Barbara (Gross, 2009), found that there were more than 180,000 infections and gathered over 70 GB of data. This included more than 1.2 million IP addresses which contacted the command and control server. The details sent included the credentials of over 8,310 accounts at 410 different institutions, included 1,770 PayPal account, with 1,660 unique credit and debit

card numbers.

A taxonomy of Botnets (Figure 2.8) classifies them in terms of (Trend Micro, 2006):

- **Attacking Behaviour.** This can be a multitude of attacking behaviours from SYN floods for a Distributed Denial of Service to identity theft.
- **Command and Control (C&C).** This is the method that the Botnet master uses to control the Bot slaves. This can be a centralized model, a P2P-Based C&C model or a random one.
- **Rallying Mechanisms.** This defines the way that Bot slaves rally around the master, and can include hard-coded IP addresses, Dynamic DNS Domain Name and a Distributed DNS service.
- **Communication Protocols.** This defines the communication protocol that the Bot master uses to communicate with the Bots, and includes IRC, HTTP, Instant Messenger (IM), P2P, and various other protocols.
- **Evasion Techniques.** This defines the methods that the Bot can use to disguise their propagation, activation, and storage. This includes HTTP/VoIP tunnelling, IPv6 tunnelling, and P2P encrypted traffic.
- **Other Observable Activities.** This includes their activities which identify themselves such as their network-based behaviour, their host-based behaviour, and global correlated behaviours. Typical activities include abnormal system calls, and trackable DNS queries.

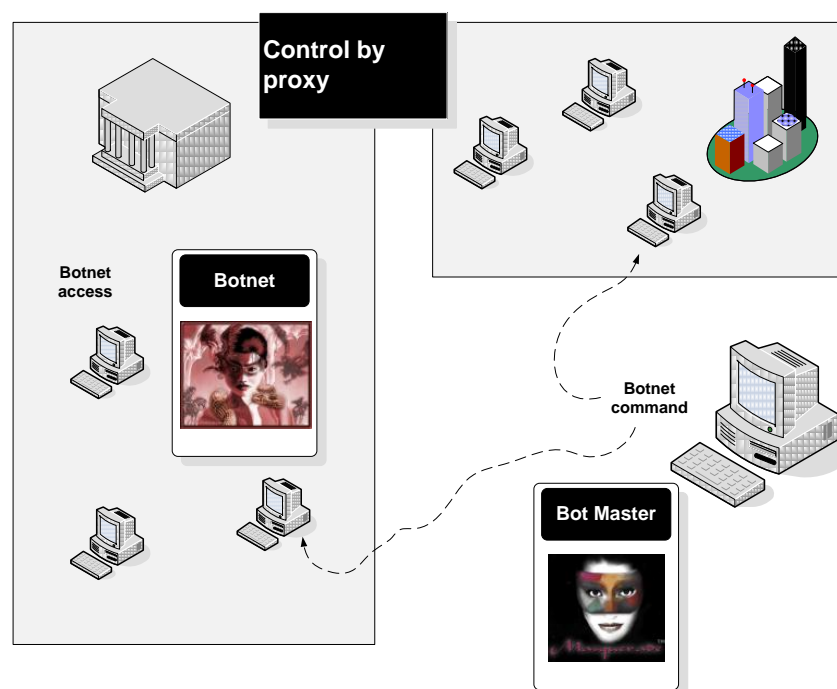


Figure 2.7 Botnets

Torpig, for example, is distributed using Mebroot, which is a rootkit which replaces the Master Boot Record (MBR), so that it is restarted at boot time. It is typically downloaded through non-malicious Web sites which have been compromised to include a piece of JavaScript code which tries to compromise the Web browser. If an

exploit is found, it downloads an executable to the host machine, and runs it. This installs Mebroot, and hands on the rest of the installation to the file manager (explorer.exe). This then loads a kernel driver that integrates with the disk driver (disk.sys), which gives Mebroot direct access to the hard disk. Once rebooted, Mebroot contacts its C&C server in order to get malicious modules, which are then stored in the c:\windows\system32 directory.

Mebroot contacts its C&C server on a regular basis using encrypted messages. This C&C server, in the case of Torpig, downloads three malicious DLLs and injects them into several key applications such as services.exe (Service Control Manager), Web browsers (Internet Explorer, Mozilla, and so on), FTP clients, instant messengers (such as Skype and MSN Messenger) and the command line prompt (CMD.EXE). Torpig is then able to listen to these applications, and pick-off information such as logins and passwords. It then reports this information back every 20 minutes to the Torpig C&C server. The method used for the communications is fairly simple, using HTTP communications with the text XOR-ed with an 8-byte key, and then converted to Base-64.

A botnet was used in a UK recent crime (2009), where a user was redirected to a fake site which stole his bank login details. The data was then passed to a bot in the UK which did the actual transfer. The trace of the money transfer is then difficult to determine, as obfuscation is used to hide the destination (Figure 2.9). A key element of this system is that the bank transfer is done through agents which are based in the UK, and this looks valid.

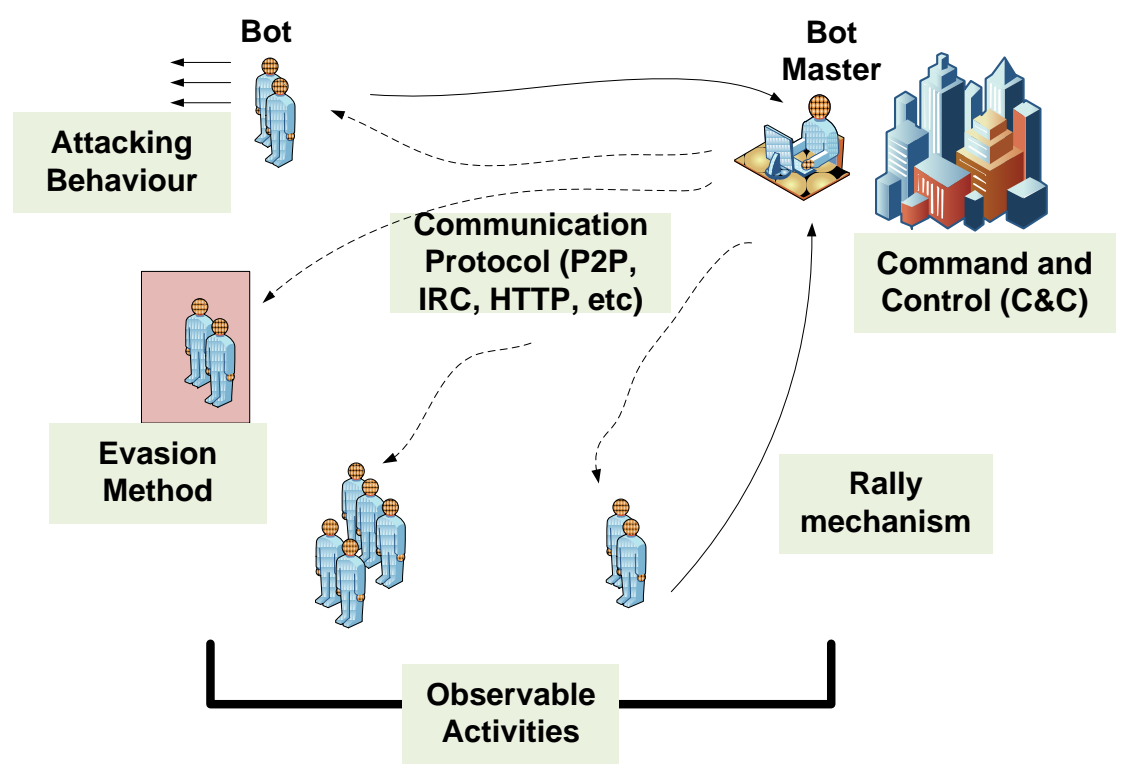


Figure 2.8 Fraud by proxy

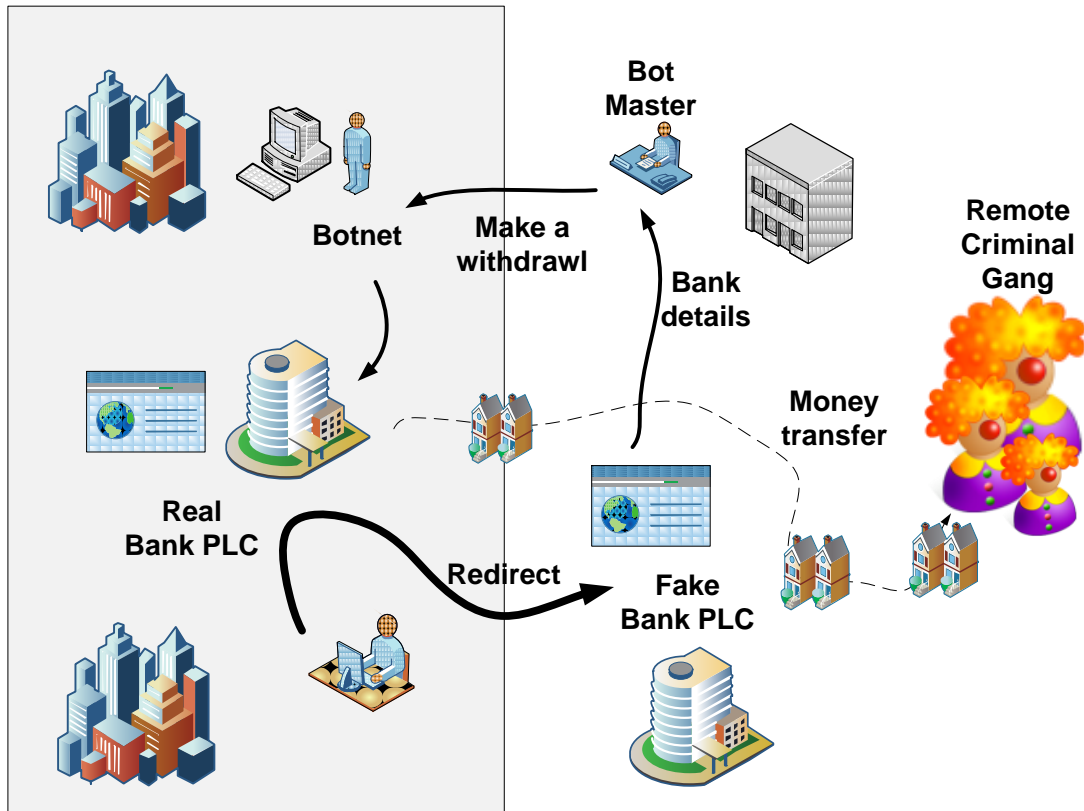


Figure 2.9 Fraud by proxy

2.7 Phishing

A major problem with most types of digital communication, processing and storage is that it is often difficult to differentiate between a true event or one which has been falsified. This is mainly because the Internet has been created with protocols which are neither secure or have any form of authentication. For example the following email looks as if it is from e-Bay (Figure 2.10). The email address of the sender of the email has been spoofed in this case, as some email relay systems allow for any email address to be used in the sender's field. It is only when the user clicks on the link do they find that it goes to a Korean Web site, which obviously asks the user to login with their e-Bay details (which could then be used to breach their e-Bay account).

It is only by looking at the raw format is there some information on the details of the email. For example, in the header, the sender of the email has not been verified:

```
Microsoft Mail Internet Headers Version 2.0
Received: from mer-w2003-6.napier-mail.napier.ac.uk ([146.176.223.1]) by
EVS1.napier-mail.napier.ac.uk with Microsoft SMTPSVC(6.0.3790.1830);
wed, 18 Jan 2006 00:17:45 +0000
Received: from pcp0011634462pcs.ivy1nd01.pa.comcast.net (Not Veri-
fied[62.32.82.127]) by mer-w2003-6.napier-mail.napier.ac.uk with NetIQ MailMarshal
(v6,1,3,15)
id <B43cd89280000>; wed, 18 Jan 2006 00:17:44 +0000
FCC: mailbox://support_id_1779124147875@ebay.com/Sent
Date: Tue, 17 Jan 2006 17:10:39 -0700
From: eBay <support_id_1779124147875@ebay.com>
```

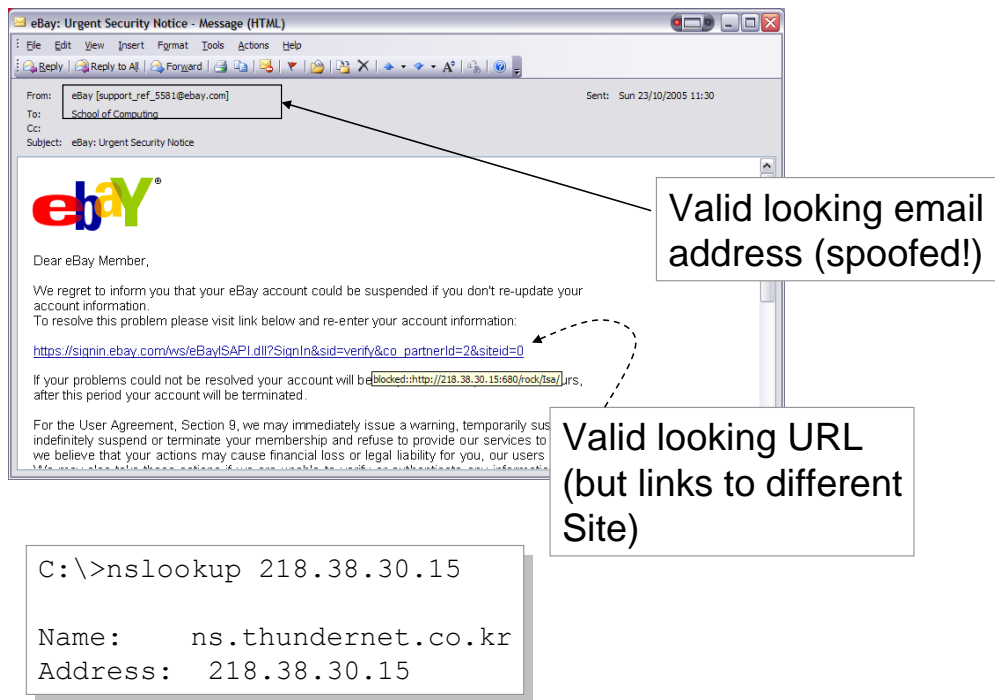



Figure 2.10 Spoofed email

This type of email is normally spotted as being fake, but an altogether more difficult one is where the sender tries to trick the reader into thinking that it was a human who wrote the email, and is asking them to prompt for some interaction, such as in Figure 2.11. In this case the text is:

“I have been waiting for quite a long time for you to reply, with the payments details . For this reason I will be forced to report you to ebay as , an unpaid item ...”

which puts pressure on the reader, as bad feedback is something that most e-Bay users want to avoid. Along with this the text looks almost like some with sloppy writing skills (which can be the case in with some e-Bay users).

Some investigation of the HTML in the email gives:

```
<TD><FONT face="Arial, Verdana" size=2>Thank you for using eBay</FONT></TD></TR>
<TR><TD><FONT face="Arial, Verdana" size=2><A
href="http://www.ebay.com">http://www.ebay.com</A>
</FONT></TD></TR></TBODY></TABLE></TD>
<TD width=358><<form method="POST" action="http://www.mailform.cz/en/form.asp">
<input type="hidden" name="mailform_userid" value="38485"><TABLE cellSpacing=0
cellPadding=0 width="99%" border=0><TBODY>
```

which shows that, rather than going to e-Bay, it goes to a Web server with a CZ domain, which will obviously mimic the e-Bay site, and steal a user's details. After which, any accesses to e-Bay must be called into doubt.

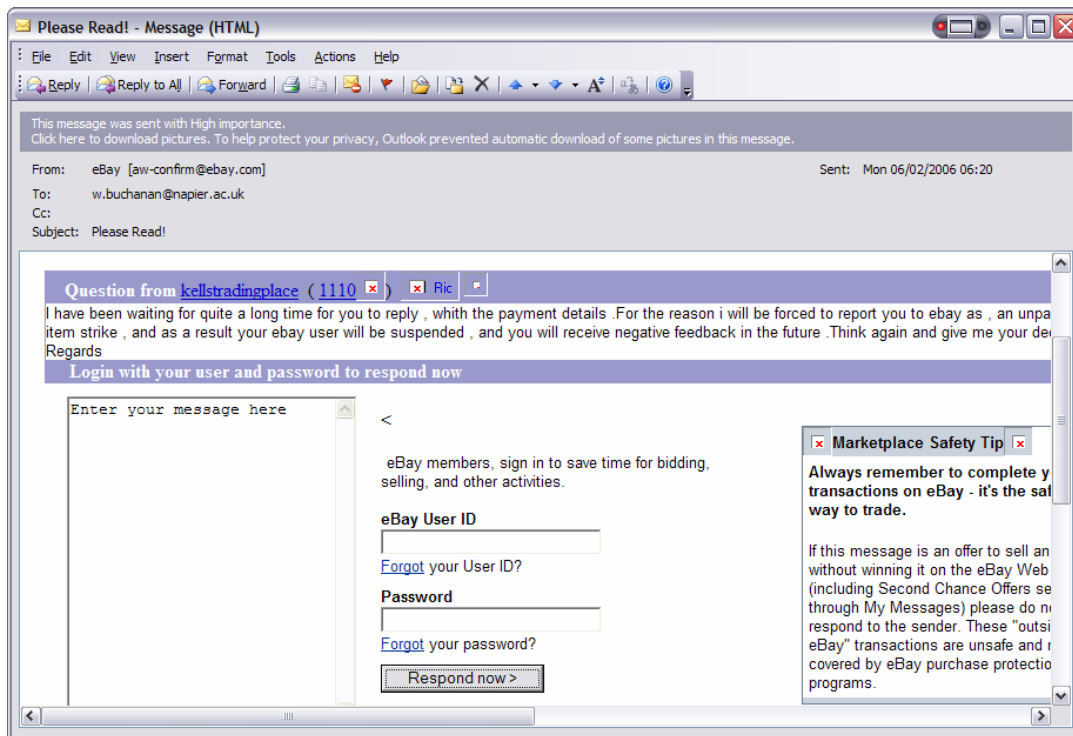


Figure 2.11 Spooled email

The methods to detect phishing includes improved training for users, and scanning content for Web links. Particular problems include:

- Any email which requests a username and a password.
- Graphics used to display text.
- Poorly laid-out content.
- IP address in a Web link. Normally a domain name would be used to identity a Web server, whereas an IP address can identity maliciousness.
- Domain on Web link differs from the sending domain. Normally the receiving domain for a Web link would relate to the sender (which would be from a trusted site).
- Graphic content taken from an external site within an email. This can be used by a malicious site to determine when an email has been read.
- Iframes within HTML content. An `<iframe>` tag allows external content to be integrated within a valid page from a trusted site.

For example, an email could have a single pixel graphic as part of the HTML content, such as:

```

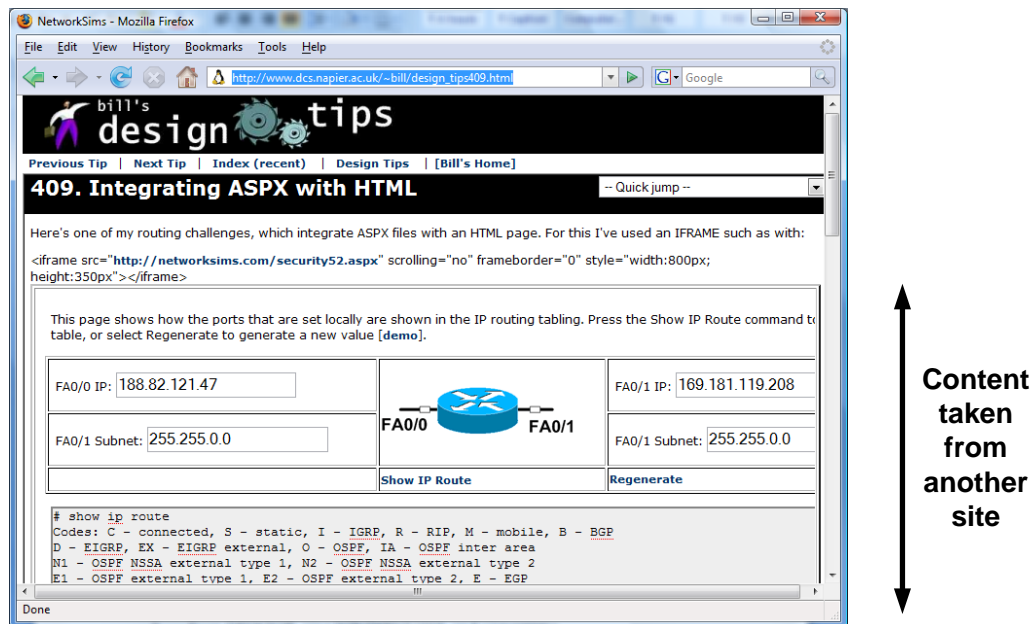
```

Where access to the pixel.gif graphic can be traced for the IP address which accessed it. In this way a spammer could determine the hosts which have successfully read an email.

With an IFRAME, content from another site can be inserted into a valid looking page, from a trusted site. For example:

http://www.dcs.napier.ac.uk/~bill/design_tips409.html

contains an external page has been integrated with an HTML file (Figure 2.12).



```
<iframe src="http://networksims.com/security52.aspx"
  scrolling="no" frameborder="0" style="width:800px;
  height:350px"></iframe>
```

Figure 2.12 IFRAME integration

2.8 Active attacks

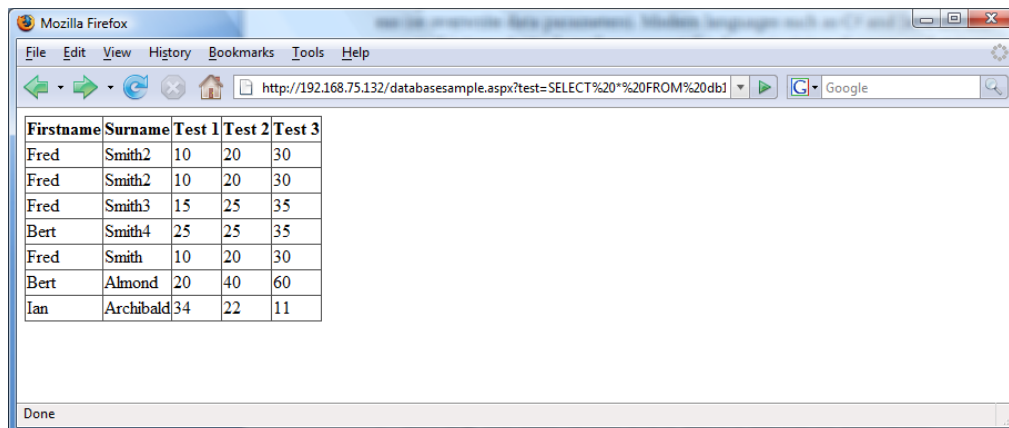
Two typical active attacks are buffer overflows, and cross scripting. **Buffer overflows** normally involve systems created with software which uses legacy software, especially C/C++, Perl and CGI script. These types of systems are often open to incorrect data input, as it is often possible to overrun the buffers used for variables, and thus write into code areas (or overwrite data parameters). Modern languages such as C# and Java are less open to this type of attack, as they support the dynamic sizing of arrays and strings.

With **cross-scripting (XSS)**, the threat normally relates to injecting scripts from one level of the system into another. An example of this is SQL injection, where the SQL commands for the database are fed through the URL of the HTTP call. For example a URL may be: <http://192.162.75.132/databasesample.aspx>, of which the variable "test" sends a variable straight to a database. Thus the call of:

```
http://192.162.75.132/databasesample.aspx?test=SELECT%20\*%20FROM%20db1
```

pass the following SQL command directly to the server:

```
SELECT *  
FROM db1
```



The screenshot shows a Mozilla Firefox browser window displaying a table of test results. The table has five columns: Firstname, Surname, Test 1, Test 2, and Test 3. The data is as follows:

Firstname	Surname	Test 1	Test 2	Test 3
Fred	Smith2	10	20	30
Fred	Smith2	10	20	30
Fred	Smith3	15	25	35
Bert	Smith4	25	25	35
Fred	Smith	10	20	30
Bert	Almond	20	40	60
Ian	Archibald	34	22	11

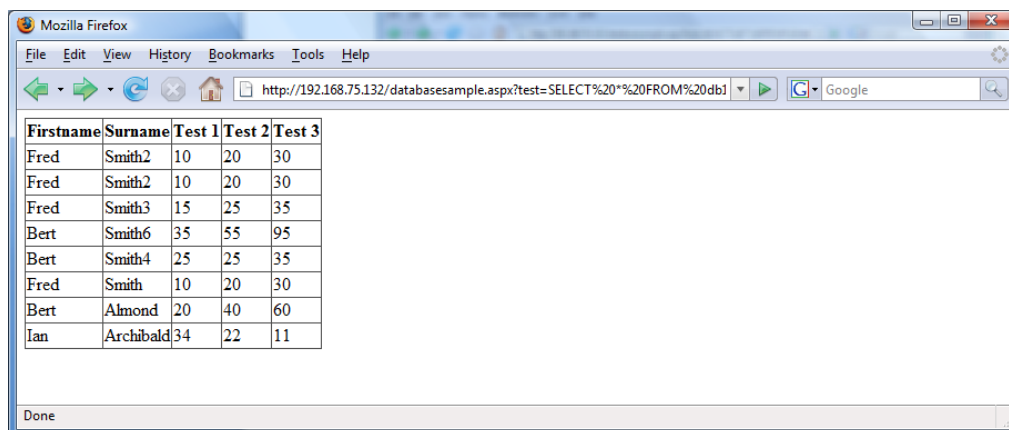
Next the following can be used to add a row onto the database:

```
SELECT * FROM db1  
INSERT INTO db1 VALUES ('Bert', 'Smith4', '25', '25', '35')
```

with:

```
http://192.162.75.132/databasesample.aspx?test=INSERT%20INTO%20db1%20VALUES%  
20('Bert','Smith6','35','55','95')
```

to give:



The screenshot shows the same Mozilla Firefox browser window, but the table now includes an additional row for Bert Smith6. The updated data is as follows:

Firstname	Surname	Test 1	Test 2	Test 3
Fred	Smith2	10	20	30
Fred	Smith2	10	20	30
Fred	Smith3	15	25	35
Bert	Smith6	35	55	95
Bert	Smith4	25	25	35
Fred	Smith	10	20	30
Bert	Almond	20	40	60
Ian	Archibald	34	22	11

The way to avoid SQL Injection is to filter any input strings, and parse them before they reach the database.

SQL Injection Demo:

http://buchananweb.co.uk/adv_security_and_network_forensics/cross_script/cross_s_cript.htm

2.9 Inference

Inference involves exploiting database weaknesses using inferences (Figure 2.13). An **indirect attack** involves deriving sensitive data from non-sensitive statistics. In the example in Figure 2.13, the user is not allowed to see the individual marks of students, but is allowed to see the average of a number of students. It can be seen that for three students, and three queries for an average mark of each of each group of two students, results in the inference of their individual mark. Inference is difficult to defend against, as there are an almost infinite number of ways that someone may view data, and the only way to overcome it is to make sure that the queries allowed on a system is limited to valid ones.

For example, in the database in Figure 2.13 there are ages of the users in the Address table. A search for the average age of two or more users is allowed, but a single user is not allowed. To breach this the intruder could search for the following average ages:

Average(Alice,Bob) = 20
Average(Bob,Eve) = 30
Average(Eve,Alice) = 40

Thus we get:

$$(A+B)/2=20 \quad [1]$$

$$(B+E)/2=30 \quad [2]$$

$$(E+A)/2=40 \quad [3]$$

$$(A+B)=40 \quad [4]$$

$$(B+E)=60 \quad [5]$$

$$(E+A)=80 \quad [6]$$

$$[4]-[5] \text{ gives } (A+B)-(B+E) = -20$$

$$\text{Thus: } A-E = -20 \quad [7]$$

$$[6]+[7] \text{ gives } (A+E) + (A-E) = 80+(-20)$$

$$\text{Thus } 2A = 80$$

$$A = 30$$

Thus B=10, and E=50 (from [5] and [6]). Thus we can infer that Alice is 30, Bob is 10 and Eve is 50.

For example an SQL query of the following will reveal the average age of Alice and Bob:

```
SELECT avg(age)
FROM address
WHERE Name='Alice' OR Name='Bob'
```

And gives a result of:

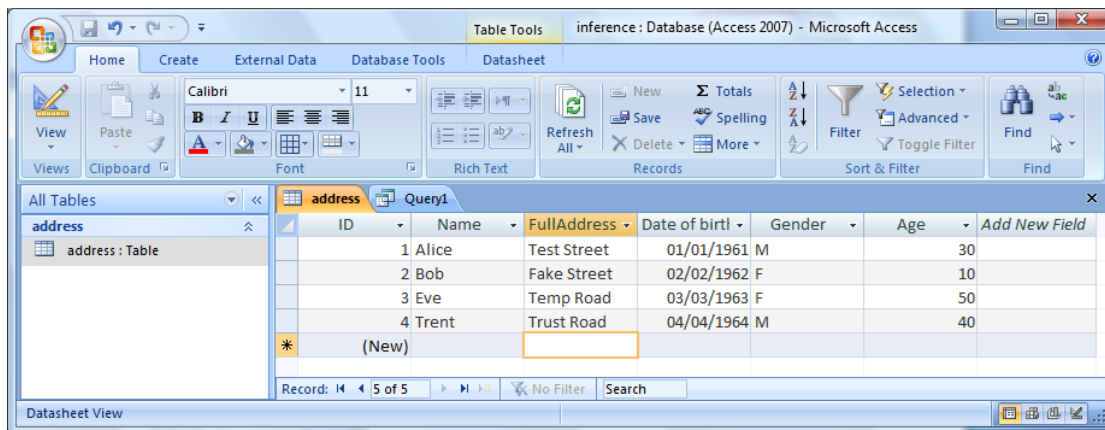


Figure 2.13 Sample database

Another example could be that users are not allowed to find the total age of all the users on the database, but the intruder could search for the total of all the male users, and then the female ones, such as:

```
SELECT sum(age)
FROM address
WHERE (Gender='M')
```

Which gives a result of: 70,

Followed by:

```
SELECT sum(age)
FROM address
WHERE (Gender='F')
```

Which gives a result of: 60.

A **direct attack** on a database involves hiding a query with a bogus condition. For example in the database in Figure 2.13 the searching for an address with a name might be disallowed, but the following obfuscates the query with a condition which will always be false (that the person is less than 30 and also greater than 30):

```
SELECT FullAddress
FROM address
WHERE (Name = 'Bob') OR (Age<30 AND Age>30)
```

The query will give:

```
Fake Street
```

which will give access to privileged information.

Often a way to overcome the release of data is to limit the number of rows returned. This can be overcome though using multiple accesses. For example the user could be limited to not showing all the names on a database, and could thus run:

```
SELECT Name
FROM address
WHERE (Gender='F')
```

followed by:

```
SELECT Name
FROM address
WHERE (Gender='M')
```

which will release all the names on the database.

Applying different levels of database security

Polyinstantiation is used in many applications for security, such as within operating systems that create new instances of directories, such as for a /tmp folder, and where the user cannot see the real /tmp folder, or any other instances of them. Within a database system Polyinstantiation is used as a way to protect high security entries where two different row instances have the same name (identifier, primary key). A relation can thus contain multiple rows using the same primary key, each with different security levels. For example a low security access would be able to access one row, with did not have sensitive information, while another one could allow access to a different row with the sensitive information. In Figure 2.14 the Security column defines security level, with a primary key of Name. In this example the table has two entries for Bob: one is a low security one without his date of birth, and with an incorrect age, where the other one has the correct details and has a high security level.

Name	FullAddress	Date of birth	Gender	Age	Security	Ad
Alice	Test Street	01/01/1961	M	30	L	
Bob	Fake Street		F	10	L	
Bob	Fake Street	02/02/1962	M	30	H	
Eve	Temp Road	03/03/1963	F	50	L	
Trent	Trust Road	04/04/1964	M	40	H	

Figure 2.14 Polyinstantiation

2.10 Affiliate scams

The Internet is being used increasingly to sell goods, and adverts are increasingly being placed on Web pages. In order to do this, the Web page owner may often charge either for the advert to be placed there, and also to gain commission for any clicks on the adverts. This can result in click-through fraud, where false clicks can be made on the advert to generate commission for the provider (Figure 2.15). As there is increasing need for the provision for adverts, affiliate networks have been created which have a lead site which has links to a number to a commercial partners, which they then link with affiliates. This works well for most affiliates, but there is a possibility that a fake affiliate can setup a number of fake Web pages, and then click-through to generate finance. A typical rate for this is around 50p/click, which could generate a considerable income with a wide range of fake sites.

In a large scale scam, there is more money to be gained from commission if the customer actually follows-through on the purchase, and makes a purchase. For this the commission gained can be considerable, such as for large-scale commission rates (such as for a 50% commission rate for a £2000 sale). The scammers then need to have access to fake IDs and/or stolen credit card details, in order to purchase the goods, or to apply for a credit card. Along with this the scammer know that IP addresses related to non-UK based hosts are unlikely to be allowed to purchase UK-based goods or a credit card from a UK-based company. Thus the scammer can create proxy agents (such as from a Botnet) where a UK-based program can create the click on the advert. An example of the scam is shown in Figure 2.16, where the Affiliate Network host has a number of customers, which is then promotes to its affiliates. It can be seen that AffiliateA actually becomes CustA, and thus gains commission from clicks or fake purchases.



Figure 2.15 Affiliate scam

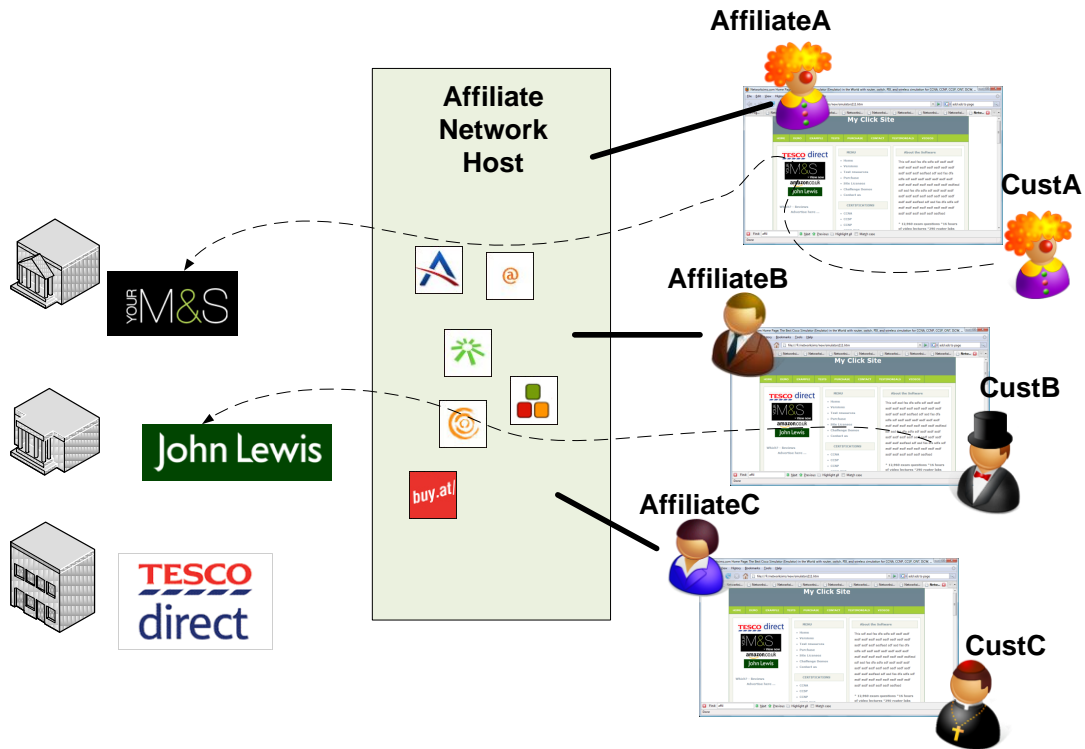


Figure 2.16 Affiliate scam

2.11 Password cracking programs

Passwords are a typical method used to protect assets and user accounts, but they are unfortunately often weak as they can often be guessed from a limited range of words from a standard dictionary. The measure of how strong a password is, is measured by its entropy.

Key entropy

Encryption key length is only one of the factors that can give a pointer to the security of the encryption process. Unfortunately most encryption processes do not use the full range of keys, as the encryption key itself is typically generated using an ASCII password. For example in wireless systems typically use a pass phase to generate the encryption key. Thus for 64-bit encryption, only five alphanumeric characters (40-bits) are used and 13 alphanumeric characters (104 bits) are used for 128-bits encryption¹. These characters are typically defined from well-know words and phrases such as:

Nap1

Whereas 128-bit encryption could use:

¹ In wireless, a 64-bit encryption key is actually only a 40 bit key, as 24 bits is used as an initialisation vector. The same goes for a 128-bit key, where the actual key is only 104 bits.

NapierStaff1

Thus, this approach typically reduces the number of useable keys, as the keys themselves will be generated from dictionaries, such as:

About
Apple
Aardvark

and keys generated from strange pass phrases such as:

xyRg54d
io2Fddse

will not be common (and could maybe be checked if the standard dictionary pass phrases did not yield a result.

Entropy measures the amount of unpredictability, and in encryption it relates to the degree of uncertainty of the encryption process. If all the keys in a 128-bit key were equally likely, then the entropy of the keys would be 128 bits. Unfortunately, due to the problems of generating keys through pass phrases the entropy of standard English can be less than 1.3 bits per character, and it is typically passwords at less than 4 bits per character. Thus for a 128-bit encryption key in wireless, and using standard English gives a maximum entropy of only 16.9 bits (1.3 times 13), which is equivalent, almost to a 17-bit encryption key length. So rather than having 202,82,409,603,651,670,423,947,251,286,016 (2^{104}) possible keys, there is only 131,072 (2^{17}) keys.

As an example, let's say an organisation uses a 40-bit encryption key, and that the organisation has the following possible phases:

Napier, napier, napier1, Napier1, napierstaff, Napierstaff, napierSoc, napier-SoC, SoC, Computing, DCS, dcs, NapierAir, napierAir, napierair, Aironet, MyAironet, SOCAironet, NapierUniversity, napieruniversity, NapierUni

which gives 20 different phases, thus the entropy is equal to:

$$\begin{aligned} \text{Entropy(bits)} &= \log_2(N) \\ &= \log_2(20) \\ &= \frac{\log_{10}(20)}{\log_{10}(2)} \\ &= 4.3 \end{aligned}$$

Thus the entropy of the 40-bit code is only 4.3 bits.

Unfortunately many password systems and operating systems such as Microsoft Windows base their encryption keys on **pass-phases**, where the private key is protected by a password. This is a major problem, as a strong encryption key can be used, but the password which protects it is open to a dictionary attack, and that the

overall entropy is low.

Hydra - just for research

Hydra is a network password cracking which should only be used to find loopholes in system, and should never be used to intrude on a system. In the following example the Windows VMware image (at 192.162.75.132) contacts the Linux image (at 192.162.75.135) for the FTP service:

```
C:\hydra-5.4-win> hydra -L login.txt -P passwd.txt 192.162.75.135 ftp
Hydra v5.4 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2009-12-29 23:10:46
[DATA] 16 tasks, 1 servers, 24 login tries (l:4/p:6), ~1 tries per task
[DATA] attacking service ftp on port 21
[STATUS] attack finished for 192.162.75.135 (waiting for childs to finish)
[21][ftp] host: 192.162.75.135 login: napier password: napier123
Hydra (http://www.thc.org) finished at 2009-12-29 23:10:58
```

Where login.txt contains a list of user IDs, and passwd.txt contains a list of passwords. It can be seen that the password and user ID have been found, as they were in these files. The verbose mode shows the details of the user IDs and passwords tried:

```
C:\hydra-5.4-win> hydra -V -L login.txt -P passwd.txt 192.162.75.135 ftp
Hydra v5.4 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2009-12-29 23:18:46
[DATA] 16 tasks, 1 servers, 24 login tries (l:4/p:6), ~1 tries per task
[DATA] attacking service ftp on port 21
[ATTEMPT] target 192.162.75.135 - login "admin" - pass "anon" - child 0 - 1 of 24
[ATTEMPT] target 192.162.75.135 - login "admin" - pass "napier" - child 1 - 2 of 24
[ATTEMPT] target 192.162.75.135 - login "admin" - pass "fred" - child 2 - 3 of 24
[ATTEMPT] target 192.162.75.135 - login "admin" - pass "none" - child 3 - 4 of 24
[ATTEMPT] target 192.162.75.135 - login "admin" - pass "password" - child 4 - 5 of 24
[ATTEMPT] target 192.162.75.135 - login "admin" - pass "napier123" - child 5 - 6 of 24
[ATTEMPT] target 192.162.75.135 - login "test" - pass "anon" - child 6 - 7 of 24
[ATTEMPT] target 192.162.75.135 - login "test" - pass "napier" - child 7 - 8 of 24
[ATTEMPT] target 192.162.75.135 - login "test" - pass "fred" - child 8 - 9 of 24
[ATTEMPT] target 192.162.75.135 - login "test" - pass "none" - child 9 - 10 of 24
[ATTEMPT] target 192.162.75.135 - login "test" - pass "password" - child 10 - 11 of 24
[ATTEMPT] target 192.162.75.135 - login "test" - pass "napier123" - child 11 - 12 of 24
[ATTEMPT] target 192.162.75.135 - login "test1" - pass "anon" - child 12 - 13 of 24
[ATTEMPT] target 192.162.75.135 - login "test1" - pass "napier" - child 13 - 14 of 24
[ATTEMPT] target 192.162.75.135 - login "test1" - pass "fred" - child 14 - 15 of 24
[ATTEMPT] target 192.162.75.135 - login "test1" - pass "none" - child 15 - 16 of 24
[ATTEMPT] target 192.162.75.135 - login "test1" - pass "password" - child 0 - 17 of 24
[ATTEMPT] target 192.162.75.135 - login "test1" - pass "napier123" - child 1 - 18 of 24
[ATTEMPT] target 192.162.75.135 - login "napier" - pass "anon" - child 2 - 19 of 24
[ATTEMPT] target 192.162.75.135 - login "napier" - pass "napier" - child 5 - 20 of 24
[ATTEMPT] target 192.162.75.135 - login "napier" - pass "fred" - child 4 - 21 of 24
[ATTEMPT] target 192.162.75.135 - login "napier" - pass "none" - child 6 - 22 of 24
[ATTEMPT] target 192.162.75.135 - login "napier" - pass "password" - child 3 - 23 of 24
[STATUS] attack finished for 192.162.75.135 (waiting for childs to finish)
[ATTEMPT] target 192.162.75.135 - login "napier" - pass "napier123" - child 7 - 24 of 24
[21][ftp] host: 192.162.75.135 login: napier password: napier123
Hydra (http://www.thc.org) finished at 2009-12-29 23:18:57
```

Remember ... only use this program on the local NAT.

Hydra Demo:

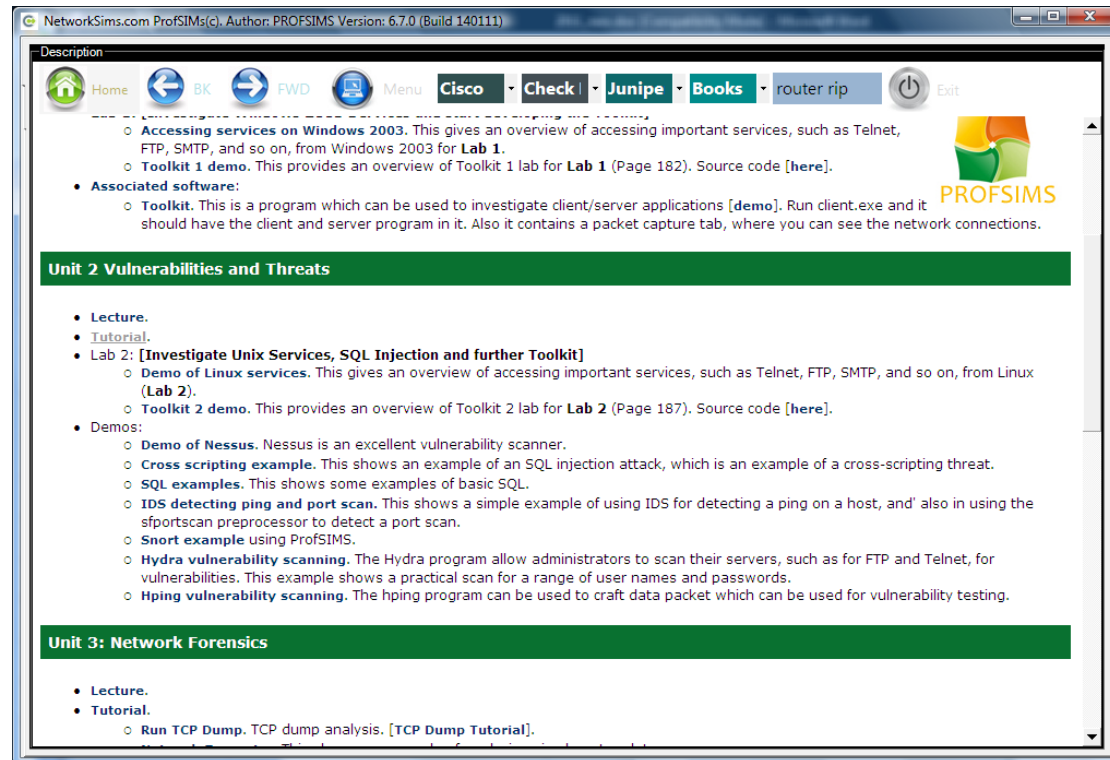
http://buchananweb.co.uk/adv_security_and_network_forensics/hydra/hydra.htm

2.12 Tutorial

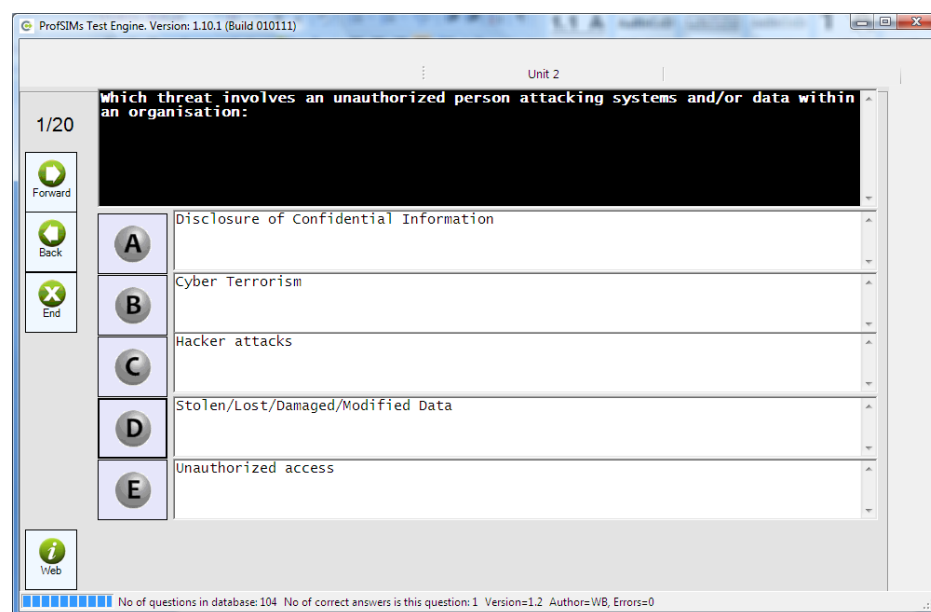
The main tutorial is at:

On-line tutorial: <http://buchananweb.co.uk/adv02.html>

The main tutorial is in NetworkSims ProfSIMs at:



And then select the Tutorial to give:



3 Network Forensics

🔗 On-line lecture: <http://buchananweb.co.uk/adv/unit03.html>

3.1 Objectives

The key objectives of this unit are to:

- Understand some of the methodologies used in network forensics.
- Provide an in-depth understanding of the key network protocols, including IP, TCP, ARP, ICMP, DNS, Application Layer protocols, and so on.
- Define a range of audit sources for network activity.

3.2 Introduction

The requirement for network forensics can be many fold, including deconstructing an internal/external network attack, a criminal investigation, and debugging a problem with a system. This unit is focus on the methodology for analysing network traffic, and in determining the key parameters that can be used to determine an evidence base for an investigation.

3.3 The key protocols

The key networking element that are typically used in an analysis of network traffic are:

- **TCP flags.** Most of the communications which occurs on the Internet involves client-server communications using TCP. The start of a connection normally involves an exchange of SYN, SYN/ACK and ACK TCP segments. Thus the start of a connection normally involves this exchange. At the end of the negotiation the TCP ports will be identified.
- **ARP activity.** This is often a sign of a host machine connect to another computer on the local network, or to the default gateway.
- **ICMP activity.** This is often a sign of the discovery of hosts, or for the route to a host.
- **DNS activity.** This is typically seen before some sort of remote access to a host.
- **Application Protocol activity.** This normally identifies the details of the actual transaction.

3.4 Ethernet, IP and TCP headers

Data is normally encapsulated with headers in order to pass the required information to be processed correctly. Figure 3.1 shows an example of a layered model, with the Application Layer (Layer 5-7), the Transport Layer (Layer 4), the Network Layer (Layer 3), the Data Link Layer (Layer 2) and the Physical Layer (Layer 1). As

the data goes through these layers extra information is added in sequence, with most layers adding the extra information at the start of the encapsulated data. The Data Link Layer is typically different as it adds information at the start and the end, as this will define the start and end of the encapsulated data. Normally at Layer 2 the transmitted encapsulated data is known as a **data frame**, while at Layer 3 it is known as a **data packet**, and at Layer 4 it is known as a **segment**.

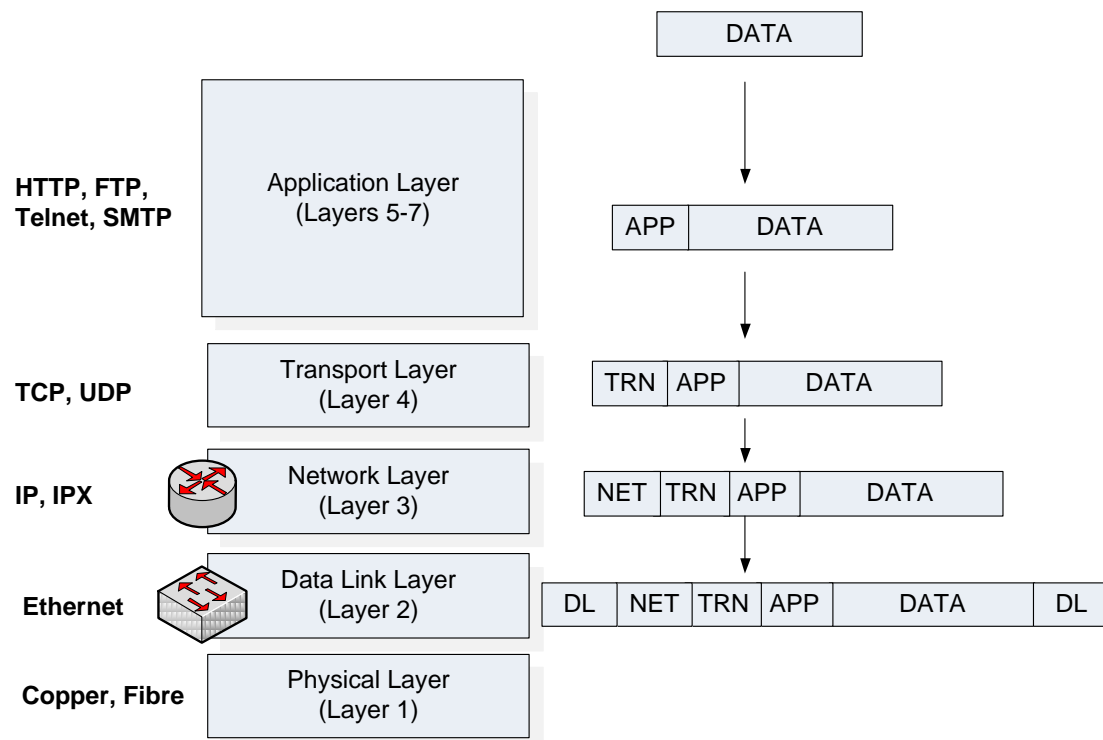


Figure 3.1 Data encapsulation

The most important Layer 2 data frame technology is Ethernet, at Layer 3 it is IP (Internet Protocol) and at Layer 4 it is TCP (Transport Control Protocol). An advantage of using Ethernet is that it will encapsulate a number of Layer 3 protocols. Figure 3.2 shows that the Type field is used to define the format of the data to be contained within its Data field. In this case, 0x800 identifies it will be an IP packet, while 0x806 defines an ARP packet. Then within an IP data packet, there is a Protocol field which will define the Layer 4 protocol. A value of 6 defines TCP, and a value of 17 defines UDP.

For a typical encapsulation of Ethernet, IP and TCP, the key parameters are:

- **Ethernet.** The Src MAC and Dest MAC addresses are 48-bit addresses which define the hardware address of the data frame.
- **IP.** The Src IP and Dest IP addresses defines the 32-bit IP (logical) addresses for the sender and the receiver of the data packet. The TTL field is used to stop the data packet from transversing the Internet infinitely. For this each intermediate routing device decrements this field by a given amount. Once it gets to zero, it will be deleted by the device which receives it. The Version field defines the IP Version, where most data packets use Version 4, while Version 6 is used to ex-

tend the address range.

- **TCP.** The reliability of the transmission is normally defined within the TCP operation. The key fields are the TCP Src and TCP Dest ports which define the source and destination TCP ports used in the communication. The Sequence Number and Acknowledge Number defines the sequence numbers for the data segments, and are used to provide acknowledgements for data transmitted and received. The Flags field are used to identify the state of the connection, and the Window field defines the number of data segments that can be received before an acknowledgement is required.

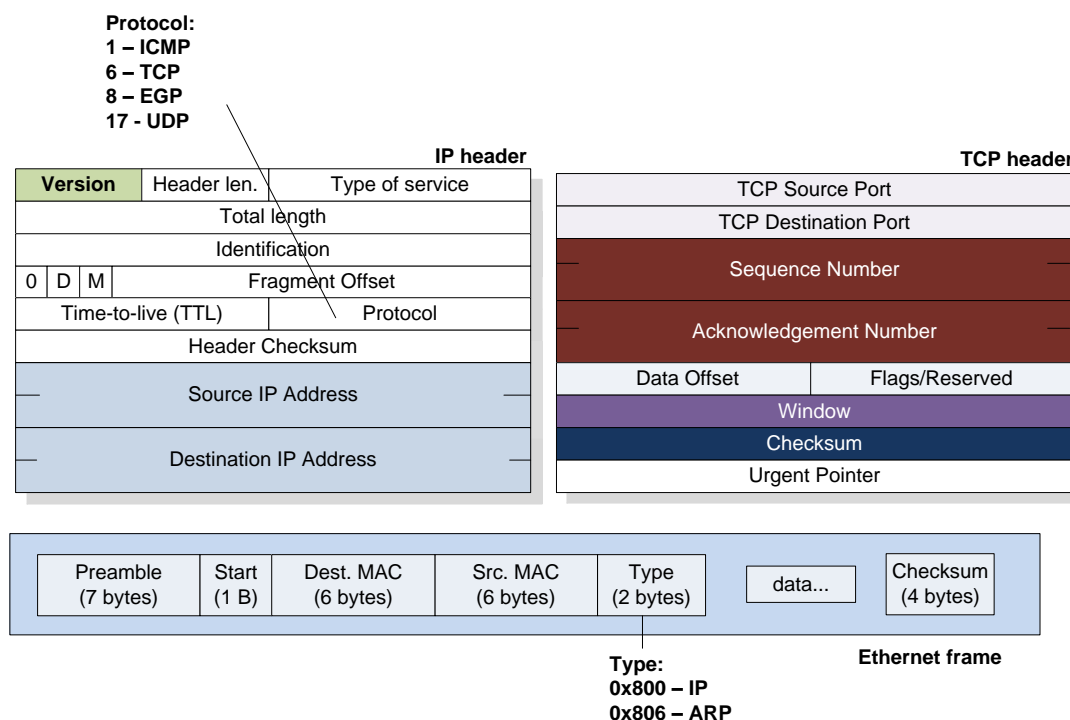


Figure 3.2 Ethernet, IP and TCP

3.5 TCP connection

The key to reliable communications over the Internet is the TCP protocol. At the core of this is the TCP flags, and the three way handshake. Figure 3.3 illustrates this procedure using a practical example. Initially three TCP data segments are exchanged, the first goes from the host (the client) to the server with the S (SYN) flag sent. The client also identifies the TCP port it wishes to use, and connects to the TCP port that the server is listening on. Next the server sends back a TCP segment with the S (SYN) and A (ACK) flags set, which identifies that it wishes to accept the connection. Finally the client sends back a TCP segment with the A (ACK) flag set. Once these TCP segments have been exchanged, there is a unique mapping of:

IP[Host]:TCPport[Host] -> IP[Server]:TCPport[Server]

As part of the three way handshake the host and server also negotiate the Window to

be used, as illustrated in Figure 3.3. In this case the final value which they settle on is 66,608. This defines the number of data segments that can be sent before the sender waits for an acknowledgement for the data previously transmitted.

A key element of TCP is that it is reliable, where each data segment has a sequence number, and data is then acknowledged for successful transmission. Figure 3.3 shows an example, where the Ack number defines the data segment that the host expects to receive next. For example in the 7th data segment, the host defines that it expects to see Seq No 70, and the 8th data segment has a Seq No of 70.

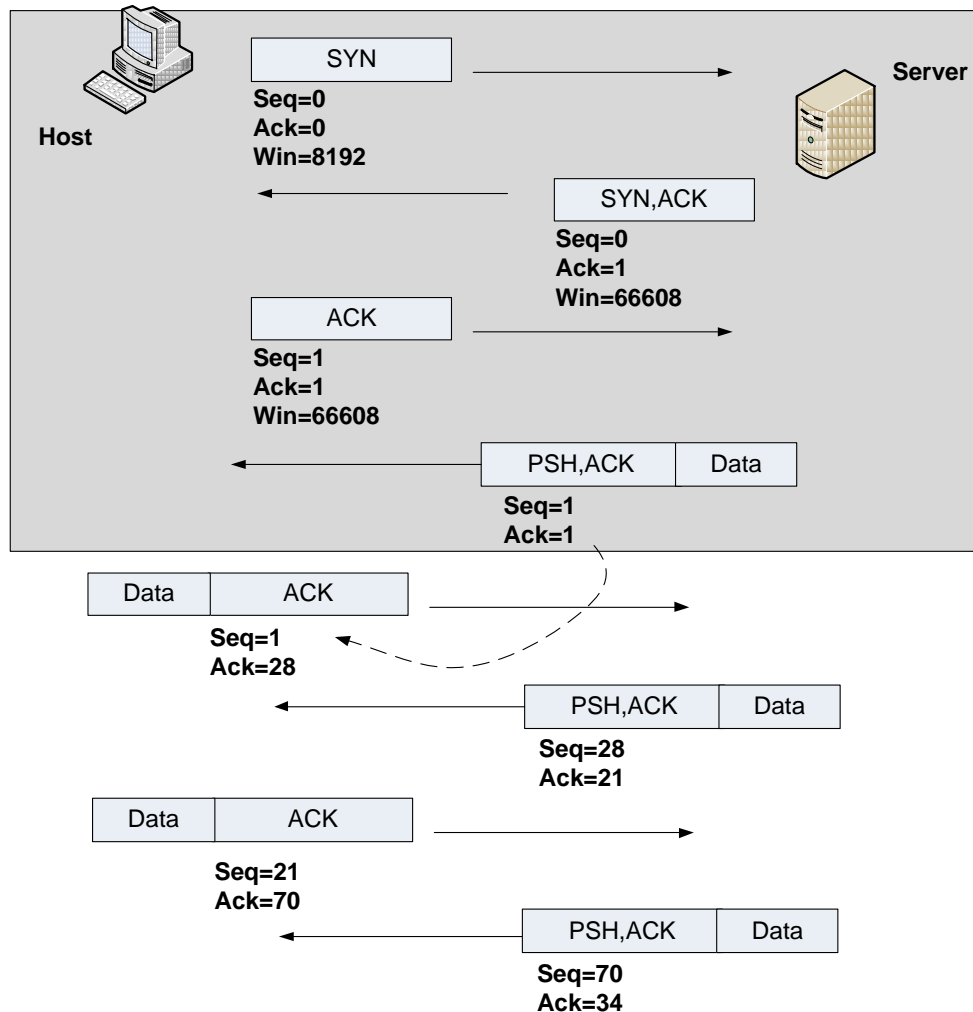


Figure 3.3 TCP flags

3.6 ARP

ARP is used to resolve an IP address to a MAC address, and is used for the first part of the communication path, and also the last part. Often ARP activity is one of the first traces of activity within any type of network connection. If a node communicates with a node within the same subnet, it can discover the MAC address for the node with an ARP broadcast. Figure 3.4 shows an example where Bob (at 192.168.75.132) needs to connect to the Internet, and thus requires the MAC address of the gateway (192.168.75.1). Thus Bob sends out an ARP request:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	Vmware_c0:00:08 192.168.75.132?	Broadcast 192.168.75.1	ARP	Who has
Frame 1 (42 bytes on wire, 42 bytes captured)					
Ethernet II, Src: Vmware c0:00:08 (00:50:56:c0:00:08), Dst: Broadcast (ff:ff:ff:ff:ff:ff)					
Address Resolution Protocol (request)					

<http://buchananweb.co.uk/log/ftp.txt> [Packet 1 and 2]

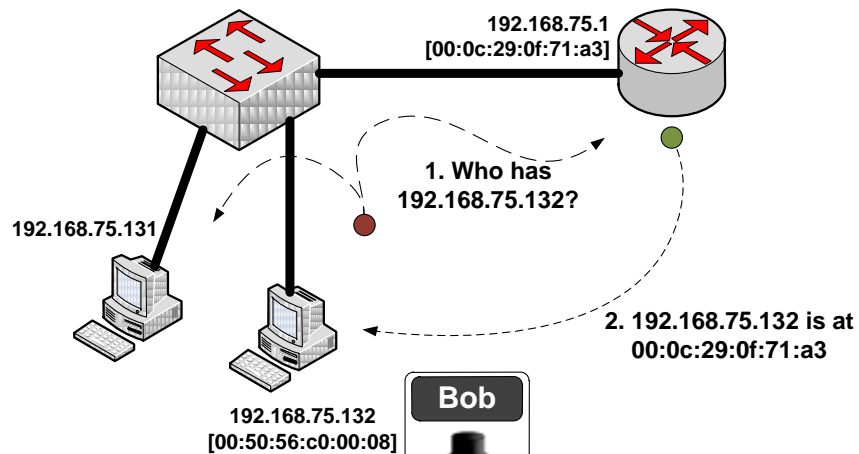
For which the gateway replies with its MAC address (00:0c:29:71:a3):

No.	Time	Source	Destination	Protocol	Info
2	0.021830	Vmware 0f:71:a3 192.168.75.132 is at	Vmware c0:00:08 00:0c:29:0f:71:a3	ARP	
Frame 2 (42 bytes on wire, 42 bytes captured)					
Ethernet II, Src: Vmware 0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware c0:00:08 (00:50:56:c0:00:08)					
Address Resolution Protocol (reply)					

On Bob's computer the ARP cache is then updated, such as:

```
C:\> arp -a
```

```
Interface: 192.168.75.1 --- 0x1d
Internet Address      Physical Address      Type
192.168.75.132       00-0c-29-0f-71-a3    dynamic
192.168.75.138       00-0c-29-6b-0e-96    dynamic
192.168.75.255       ff-ff-ff-ff-ff-ff    static
```



```
Interface: 192.168.75.1 --- 0x1d
Internet Address      Physical Address      Type
192.168.75.132       00-0c-29-0f-71-a3    dynamic
192.168.75.138       00-0c-29-6b-0e-96    dynamic
192.168.75.255       ff-ff-ff-ff-ff-ff    static
```

Figure 3.4 ARP activity

Most Windows computers have an ARP timeout of 10 minutes, where Cisco routers timeout after 4 hours.

3.7 SYN

In network forensics, the SYN flag is key to finding the starting point of a connection, as every TCP connection requires a three-way handshake. In the following example the connection details of the connection are:

192.168.75.1:3655 -> 192.168.75.132:21

Where the host is at 192.168.75.1, and the FTP server is at 192.168.75.132.

No.	Time	Source	Destination	Protocol	Info
3	0.021867	192.168.75.1	192.168.75.132	TCP	abatemgr >
ftp [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2 TSV=683746 TSER=0					
Frame 3 (74 bytes on wire, 74 bytes captured)					
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)					
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132)					
Transmission Control Protocol, Src Port: abatemgr (3655), Dst Port: ftp (21), Seq: 0, Len: 0					
No.	Time	Source	Destination	Protocol	Info
4	0.022961	192.168.75.132	192.168.75.1	TCP	ftp >
abatemgr [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 WS=0 TSV=0 TSER=0					
Frame 4 (78 bytes on wire, 78 bytes captured)					
Ethernet II, Src: Vmware_0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)					
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1 (192.168.75.1)					
Transmission Control Protocol, Src Port: ftp (21), Dst Port: abatemgr (3655), Seq: 0, Ack: 1, Len: 0					
No.	Time	Source	Destination	Protocol	Info
5	0.023078	192.168.75.1	192.168.75.132	TCP	abatemgr >
ftp [ACK] Seq=1 Ack=1 Win=66608 Len=0 TSV=683748 TSER=0					
Frame 5 (66 bytes on wire, 66 bytes captured)					
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)					
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132)					
Transmission Control Protocol, Src Port: abatemgr (3655), Dst Port: ftp (21), Seq: 1, Ack: 1, Len: 0					

View this file: <http://buchananweb.co.uk/log/ftp.txt> [Packets 3-5]

3.8 Application Layer Analysis - FTP

The FTP application protocol uses commands (USER, PASS, MKD, CWD, QUIT, RMD, and so on), where there is a numeric response value (such as 226 – Transfer complete and 250 – CWD command successful). The following shows the requests and replies passed from a client to a server:

Server

```
220 Microsoft FTP Service
331 Password required for Administrator.
230 User Administrator logged in.
215 Windows_NT
```

Client

```
USER Administrator
PASS napier
SYST
```

```

257 "/" is current directory.
227 Entering Passive Mode (192,168,75,132,4,22).
125 Data connection already open; Transfer starting.
226 Transfer complete.
250 CWD command successful.
227 Entering Passive Mode (192,168,75,132,4,23).
125 Data connection already open; Transfer starting.
226 Transfer complete.
257 "/" is current directory.
200 Type set to A.
227 Entering Passive Mode (192,168,75,132,4,24).
125 Data connection already open; Transfer starting.
226 Transfer complete.

```

PWD
PASV
LIST
CWD /
PASV
LIST
PWD
TYPE A
PASV
STOR db1.csv

This example uses Passive FTP, which creates a server port which the client must connect to. This is determined from:

```
227 Entering Passive Mode (192,168,75,132,4,24).
```

Where the last two digital determine the port that will be created. This is calculated from 256 times the second last digit, plus the last digit. Thus the port created is 1048 (4x256+24). The client will then create a connection on this port, and transfer the information.

The following shows the initial data packets exchanged for the connection defined in the previous two transfers (Sections 3.6 and 3.7):

No.	Time	Source	Destination	Protocol	Info
6	0.026461	192.168.75.132	192.168.75.1	FTP	Response: 220 Microsoft FTP Service
Frame 6 (93 bytes on wire, 93 bytes captured) Ethernet II, Src: Vmware 0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware c0:00:08 (00:50:56:c0:00:08) Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1 (192.168.75.1) Transmission Control Protocol, Src Port: ftp (21), Dst Port: abatemgr (3655), Seq: 1, Ack: 1, Len: 27 File Transfer Protocol (FTP)					
7	0.107380	192.168.75.1	192.168.75.132	FTP	Request: USER Administrator
Frame 7 (86 bytes on wire, 86 bytes captured) Ethernet II, Src: Vmware c0:00:08 (00:50:56:c0:00:08), Dst: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132) Transmission Control Protocol, Src Port: abatemgr (3655), Dst Port: ftp (21), Seq: 1, Ack: 28, Len: 20 File Transfer Protocol (FTP)					
8	0.108092	192.168.75.132	192.168.75.1	FTP	Response: 331 Password required for Administrator.
Frame 8 (108 bytes on wire, 108 bytes captured) Ethernet II, Src: Vmware_0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware_c0:00:08					

```

(00:50:56:c0:00:08)
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1 (192.168.75.1)
Transmission Control Protocol, Src Port: ftp (21), Dst Port: abatemgr (3655), Seq: 28, Ack:
21, Len: 42
File Transfer Protocol (FTP)

No.      Time           Source           Destination      Protocol Info
   9  0.108387     192.168.75.1    192.168.75.132  FTP          Request: PASS napier

Frame 9 (79 bytes on wire, 79 bytes captured)
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3
(00:0c:29:0f:71:a3)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132)
Transmission Control Protocol, Src Port: abatemgr (3655), Dst Port: ftp (21), Seq: 21, Ack:
70, Len: 13
File Transfer Protocol (FTP)

No.      Time           Source           Destination      Protocol Info
  10  0.110448     192.168.75.132  192.168.75.1    FTP          Response: 230 User
Administrator logged in.

Frame 10 (101 bytes on wire, 101 bytes captured)
Ethernet II, Src: Vmware_0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware_c0:00:08
(00:50:56:c0:00:08)
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1 (192.168.75.1)
Transmission Control Protocol, Src Port: ftp (21), Dst Port: abatemgr (3655), Seq: 70, Ack:
34, Len: 35
File Transfer Protocol (FTP)

```

View this file: <http://buchananweb.co.uk/log/ftp.txt> [Packets 6 and on]

3.9 ICMP

ICMP is a protocol used to provide debug information, such as to determine if a host is operating (using ping) or to trace the route to a destination (using tracer). Unfortunately it can also be used by malicious sources to determine if a device is on-line (and which ones), and the route that data packets take. The following shows a ping request from 192.168.75.1 to 192.168.75.132:

```

No.      Time           Source           Destination      Protocol Info
   10  13.706916     192.168.75.1    192.168.75.132  ICMP        Echo (ping)
request
Frame 10 (74 bytes on wire, 74 bytes captured)
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3
(00:0c:29:0f:71:a3)
  Destination: Vmware 0f:71:a3 (00:0c:29:0f:71:a3)
  Source: Vmware c0:00:08 (00:50:56:c0:00:08)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132
(192.168.75.132)
Internet Control Message Protocol

No.      Time           Source           Destination      Protocol Info
   11  13.707279     192.168.75.132  192.168.75.1    ICMP        Echo (ping)
reply
Frame 11 (74 bytes on wire, 74 bytes captured)
Ethernet II, Src: Vmware_0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware_c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1
(192.168.75.1)
Internet Control Message Protocol

```

View this file: <http://buchananweb.co.uk/log/ping.txt>

3.10 DNS

DNS lookup is often a key pointer to the start of some form of initial activity. The protocol operates, normally, using UDP on Port 53. In the following example, the host (192.168.0.20) contacts the DNS server at 192.168.0.1. Packet 7 shows that the lookup is for `www.intel.com`, which, in Packet 8, returns the lookup for `www.intel.com`, such as:

```
F:\docs\src\clientToolkit\log>nslookup www.intel.com
Server: UnKnown
Address: 192.168.0.1

Non-authoritative answer:
Name: a961.g.akamai.net
Addresses: 81.52.140.11
           81.52.140.83
Aliases: www.intel.com
         www.intel.com.edgesuite.net
         www.intel-sino.com.edgesuite.net
         www.intel-sino.com.edgesuite.net.chinaredirector.akadns.net
```

In this case the UDP details are:

192.168.0.20:63227 -> 192.168.0.1:53

No.	Time	Source	Destination	Protocol	Info
7	5.386386	192.168.0.20	192.168.0.1	DNS	Standard
query A www.intel.com					
Frame 7 (73 bytes on wire, 73 bytes captured)					
Ethernet II, Src: IntelCor_4f:30:1d (00:1f:3c:4f:30:1d), Dst: Netgear_b0:d6:8c (00:18:4d:b0:d6:8c)					
Destination: Netgear b0:d6:8c (00:18:4d:b0:d6:8c)					
Source: IntelCor 4f:30:1d (00:1f:3c:4f:30:1d)					
Type: IP (0x0800)					
Internet Protocol, Src: 192.168.0.20 (192.168.0.20), Dst: 192.168.0.1 (192.168.0.1)					
User Datagram Protocol, Src Port: 63227 (63227), Dst Port: domain (53)					
Domain Name System (query)					
8	5.461009	192.168.0.1	192.168.0.20	DNS	Standard
query response CNAME www.intel.com.edgesuite.net CNAME www.intel-sino.com.edgesuite.net CNAME www.intel-sino.com.edgesuite.net.chinaredirector.akadns.net CNAME a961.g.akamai.net A 92.122.126.176 A 92.122.126.146					
Frame 8 (547 bytes on wire, 547 bytes captured)					
Ethernet II, Src: Netgear_b0:d6:8c (00:18:4d:b0:d6:8c), Dst: IntelCor_4f:30:1d (00:1f:3c:4f:30:1d)					
Destination: IntelCor 4f:30:1d (00:1f:3c:4f:30:1d)					
Source: Netgear b0:d6:8c (00:18:4d:b0:d6:8c)					
Type: IP (0x0800)					
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.20 (192.168.0.20)					
User Datagram Protocol, Src Port: domain (53), Dst Port: 63227 (63227)					
Domain Name System (response)					

View this file: <http://buchananweb.co.uk/log/dnslookup.txt>

3.11 Port scan

A port scan is often seen as a sign of malicious activity, where an intruder tries to find the ports which are open on a computer. The following shows an NMAP scan

from 192.168.75.1 to 192.168.75.132, where it sends SYNs for key ports, such as Telnet (23), RAP (256), IMAPS (993), POP3 (110), and so on. If a connection is made on the port, there will be a response, otherwise NMAP continues to scan the ports. A continual accessing of a range of a ports over a time interval, often shows intruder activity.

No.	Time	Source	Destination	Protocol	Info
85	25.420710	192.168.75.1	192.168.75.132	TCP	54370 > telnet [SYN] Seq=0 Win=1024 Len=0 MSS=1460
Frame 85 (58 bytes on wire, 58 bytes captured) Ethernet II, Src: Vmware c0:00:08 (00:50:56:c0:00:08), Dst: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Destination: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Source: Vmware_c0:00:08 (00:50:56:c0:00:08) Type: IP (0x0800) Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132) Transmission Control Protocol, Src Port: 54370 (54370), Dst Port: telnet (23), Seq: 0, Len: 0					
86	25.420836	192.168.75.1	192.168.75.132	TCP	54370 > rap [SYN] Seq=0 Win=2048 Len=0 MSS=1460
Frame 86 (58 bytes on wire, 58 bytes captured) Ethernet II, Src: Vmware c0:00:08 (00:50:56:c0:00:08), Dst: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Destination: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Source: Vmware_c0:00:08 (00:50:56:c0:00:08) Type: IP (0x0800) Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132) Transmission Control Protocol, Src Port: 54370 (54370), Dst Port: rap (256), Seq: 0, Len: 0					
87	25.420897	192.168.75.1	192.168.75.132	TCP	54370 > imaps [SYN] Seq=0 Win=3072 Len=0 MSS=1460
Frame 87 (58 bytes on wire, 58 bytes captured) Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3 (00:0c:29:0f:71:a3) Destination: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Source: Vmware_c0:00:08 (00:50:56:c0:00:08) Type: IP (0x0800) Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132) Transmission Control Protocol, Src Port: 54370 (54370), Dst Port: imaps (993), Seq: 0, Len: 0					
88	25.420941	192.168.75.1	192.168.75.132	TCP	54370 > pop3s [SYN] Seq=0 Win=2048 Len=0 MSS=1460
Frame 88 (58 bytes on wire, 58 bytes captured) Ethernet II, Src: Vmware c0:00:08 (00:50:56:c0:00:08), Dst: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Destination: Vmware 0f:71:a3 (00:0c:29:0f:71:a3) Source: Vmware_c0:00:08 (00:50:56:c0:00:08) Type: IP (0x0800) Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132) Transmission Control Protocol, Src Port: 54370 (54370), Dst Port: pop3s (995), Seq: 0, Len: 0					
89	25.420984	192.168.75.1	192.168.75.132	TCP	54370 > microsoft-ds [SYN] Seq=0 Win=1024 Len=0 MSS=1460
Frame 89 (58 bytes on wire, 58 bytes captured) Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3					

```

(00:0c:29:0f:71:a3)
  Destination: Vmware 0f:71:a3 (00:0c:29:0f:71:a3)
  Source: Vmware c0:00:08 (00:50:56:c0:00:08)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132
(192.168.75.132)
Transmission Control Protocol, Src Port: 54370 (54370), Dst Port: microsoft-ds (445),
Seq: 0, Len: 0

No.      Time           Source           Destination      Protocol Info
    90  25.421026    192.168.75.1    192.168.75.132  TCP        54370 > smux
[SYN] Seq=0 Win=1024 Len=0 MSS=1460

Frame 90 (58 bytes on wire, 58 bytes captured)
Ethernet II, Src: Vmware c0:00:08 (00:50:56:c0:00:08), Dst: Vmware 0f:71:a3
(00:0c:29:0f:71:a3)
  Destination: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Source: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132
(192.168.75.132)
Transmission Control Protocol, Src Port: 54370 (54370), Dst Port: smux (199), Seq: 0,
Len: 0

No.      Time           Source           Destination      Protocol Info
    91  25.421069    192.168.75.1    192.168.75.132  TCP        54370 > pptp
[SYN] Seq=0 Win=2048 Len=0 MSS=1460

```

View this file: <http://buchananweb.co.uk/log/webpage.txt> [Packet 85 on]

3.12 SYN flood

Distributed Denial-of-Service (DDoS) is one of the most difficult attacks to defend against, as it is often difficult to differentiate malicious connections from non-malicious ones. The following shows an example of a host (192.168.75.137) connecting to port 80 on 192.168.75.1, and results in the connections of:

192.168.75.137:1608 -> 192.168.71.1:80

192.168.75.137:1609 -> 192.168.71.1:80

```

No.      Time           Source           Destination      Protocol Info
    2  4.510329    192.168.75.137  192.168.75.1    HTTP       Continuation
or non-HTTP traffic

Frame 2 (58 bytes on wire, 58 bytes captured)
Ethernet II, Src: Vmware 6b:0e:96 (00:0c:29:6b:0e:96), Dst: Vmware c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_6b:0e:96 (00:0c:29:6b:0e:96)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.137 (192.168.75.137), Dst: 192.168.75.1
(192.168.75.1)
Transmission Control Protocol, Src Port: smart-lm (1608), Dst Port: http (80), Seq: 0,
Len: 4
Hypertext Transfer Protocol

No.      Time           Source           Destination      Protocol Info
    3  5.514164    192.168.75.137  192.168.75.1    HTTP       Continuation
or non-HTTP traffic

Frame 3 (58 bytes on wire, 58 bytes captured)
Ethernet II, Src: Vmware 6b:0e:96 (00:0c:29:6b:0e:96), Dst: Vmware c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_6b:0e:96 (00:0c:29:6b:0e:96)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.137 (192.168.75.137), Dst: 192.168.75.1
(192.168.75.1)
Transmission Control Protocol, Src Port: isysg-lm (1609), Dst Port: http (80), Seq: 0,

```

```

Len: 4
Hypertext Transfer Protocol

No.      Time           Source           Destination      Protocol Info
    4 6.517235     192.168.75.137   192.168.75.1    HTTP      Continuation
or non-HTTP traffic

Frame 4 (58 bytes on wire, 58 bytes captured)
Ethernet II, Src: Vmware 6b:0e:96 (00:0c:29:6b:0e:96), Dst: Vmware c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_6b:0e:96 (00:0c:29:6b:0e:96)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.137 (192.168.75.137), Dst: 192.168.75.1
(192.168.75.1)
Transmission Control Protocol, Src Port: taurus-wh (1610), Dst Port: http (80), Seq:
0, Len: 4
Hypertext Transfer Protocol

No.      Time           Source           Destination      Protocol Info
    5 7.520267     192.168.75.137   192.168.75.1    HTTP      Continuation
or non-HTTP traffic

Frame 5 (58 bytes on wire, 58 bytes captured)
Ethernet II, Src: Vmware 6b:0e:96 (00:0c:29:6b:0e:96), Dst: Vmware c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_6b:0e:96 (00:0c:29:6b:0e:96)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.137 (192.168.75.137), Dst: 192.168.75.1
(192.168.75.1)
Transmission Control Protocol, Src Port: ill (1611), Dst Port: http (80), Seq: 0, Len:
4
Hypertext Transfer Protocol

```

View this file: http://buchananweb.co.uk/log/hping_port80.txt [Packet 2 on]

A FIN flood is shown in http://buchananweb.co.uk/log/hping_fin.txt

3.13 Spoofed addresses

One method that an intruder can use to hide their tracks is to substitute their IP address with another address. In the following example the intruder has used NMAP with a spoofed address of 10.0.0.1:

```
nmap -e eth0 192.168.75.132 -S 10.0.0.1 -sS
```

to give a result of:

```

No.      Time           Source           Destination      Protocol Info
    5 0.044549     10.0.0.1         192.168.75.132   TCP        40484 > https
[SYN] Seq=0 Win=1024 Len=0 MSS=1460

Frame 5 (58 bytes on wire, 58 bytes captured)
Ethernet II, Src: Vmware 6b:0e:96 (00:0c:29:6b:0e:96), Dst: Vmware 0f:71:a3
(00:0c:29:0f:71:a3)
Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 192.168.75.132 (192.168.75.132)
Transmission Control Protocol, Src Port: 40484 (40484), Dst Port: https (443), Seq: 0,
Len: 0

No.      Time           Source           Destination      Protocol Info
    6 0.044857     10.0.0.1         192.168.75.132   TCP        40484 > ptp
[SYN] Seq=0 Win=2048 Len=0 MSS=1460

Frame 6 (58 bytes on wire, 58 bytes captured)
Ethernet II, Src: Vmware 6b:0e:96 (00:0c:29:6b:0e:96), Dst: Vmware 0f:71:a3
(00:0c:29:0f:71:a3)
Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 192.168.75.132 (192.168.75.132)

```



```

Transmission Control Protocol, Src Port: 40484 (40484), Dst Port: ptp (1723), Seq: 0,
Len: 0

No.      Time           Source           Destination      Protocol Info
    7 0.044871    192.168.75.132  10.0.0.1         TCP      https > 40484
[RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Frame 7 (54 bytes on wire (42 bytes captured) on interface 0:00:00:00:00:00)
Ethernet II, Src: Vmware 0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware f5:2e:f3
(00:50:56:f5:2e:f3)
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 10.0.0.1 (10.0.0.1)
Transmission Control Protocol, Src Port: https (443), Dst Port: 40484 (40484), Seq: 1,
Ack: 1, Len: 0

No.      Time           Source           Destination      Protocol Info
    8 0.045043    192.168.75.132  10.0.0.1         TCP      ptp > 40484
[RST, ACK] Seq=1 Ack=1 Win=0 Len=0

```

View this file: http://buchananweb.co.uk/log/spoof_address.txt [Packet 5 on]

Private addresses within a public address space normally shows maliciousness. These addresses are in the following ranges:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255

3.14 Application Layer Analysis - HTTP

The foundation protocol of the WWW is the Hypertext Transfer Protocol (HTTP) which can be used in any client/server application involving hypertext. It is used on the WWW for transmitting information using hypertext jumps and can support the transfer of plaintext, hypertext, audio, images, or any Internet-compatible information. The most recently defined standard is HTTP 1.1, which has been defined by the IETF standard.

HTTP is a stateless protocol where each transaction is independent of any previous transactions. Thus when the transaction is finished the TCP/IP connection is disconnected, as illustrated in Figure 3.5. The advantage of being stateless is that it allows the rapid access of WWW pages over several widely distributed servers. It uses the TCP protocol to establish a connection between a client and a server for each transaction then terminates the connection once the transaction completes.

HTTP also supports many different formats of data. Initially a client issues a request to a server which may include a prioritized list of formats that it can handle. This allows new formats to be easily added and also prevents the transmission of unnecessary information.

A client's WWW browser (the user agent) initially establishes a direct connection with the destination server which contains the required WWW page. To make this connection the client initiates a TCP connection between the client and the server. After this is established the client then issues an HTTP request, such as the specific command (the method), the URL, and possibly extra information such as request parameters or client information. When the server receives the request, it attempts to perform the requested action. It then returns an HTTP response, which includes status information, a success/error code, and extra information. After the client receives this, the TCP connection is closed.

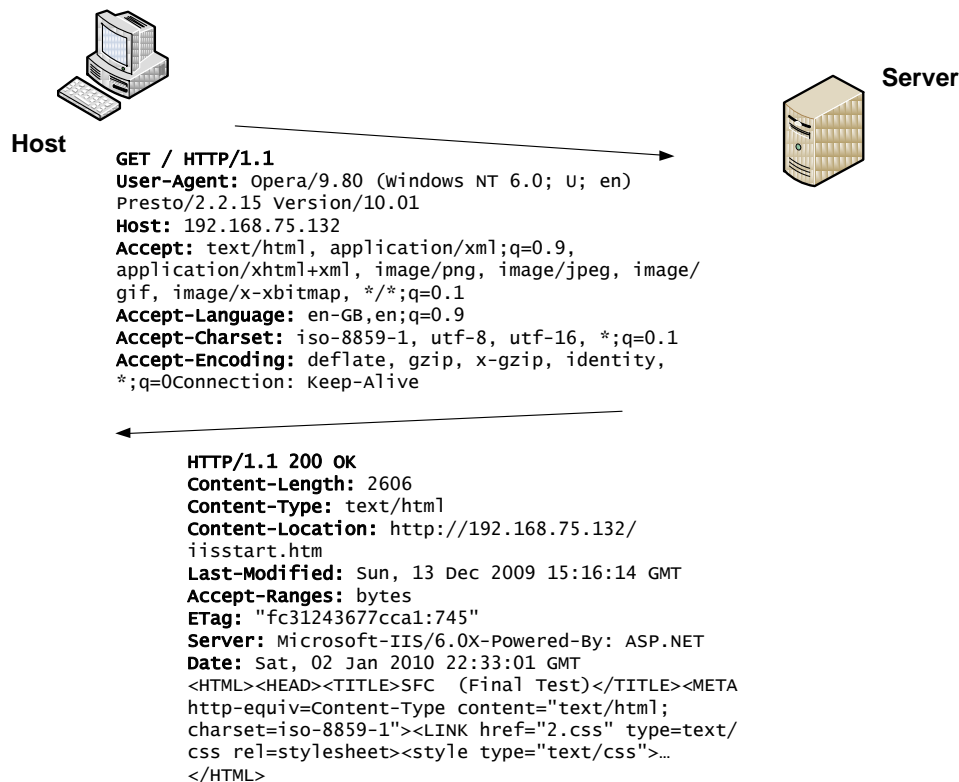


Figure 3.5 Example HTTP transaction

HTTP messages

The simple request is a `GET` command with the requested URI such as:

```
GET /info/dept/courses.html
```

The simple response is a block containing the information identified in the URI (called the entity-body).

Full requests/responses

Very few security measures or enhanced services are built into the simple requests/responses. HTTP Version 1.0/1.1 improves on the simple requests/responses by adding many extra requests and responses, as well as adding extra information about the data supported. Each message header consists of a number of fields which begin on a new line and consist of the field name followed by a colon and the field value. A full request starts with a request line command (such as `GET`, `MOVE` or `DELETE`) and is then followed by one or more of the following:

- General-headers which contain general fields that do not apply to the entity being transferred (such as MIME version, date, and so on).
- Request-headers which contain information on the request and the client (e.g. the client's name, its authorization, and so on).
- Entity-headers which contain information about the resource identified by the request and entity-body information (such as the type of encoding, the language, the title, the time when it was last modified, the type of resource it is, when it ex-

pires, and so on).

- Entity-body which contains the body of the message (such as HTML text, an image, a sound file, and so on).

A full response starts with a response status code (such as OK, Moved Temporarily, Accepted, Created, Bad Request, and so on) and is then followed by one or more of the following:

- General-headers, as with requests, contain general fields which do not apply to the entity being transferred (MIME version, date, and so on).
- Response-headers which contain information on the response and the server (e.g. the server's name, its location and the time the client should retry the server).
- Entity-headers, as with request, which contain information about the resource identified by the request and entity-body information (such as the type of encoding, the language, the title, the time when it was last modified, the type of resource it is, when it expires, and so on).
- Entity-body, as with requests, which contains the body of the message (such as HTML text, an image, a sound file, and so on).

The following example shows an example request. The first line is always the request method; in this case it is GET. Next there are various headers. The general-header field is Content-Type, the request-header fields are If-Modified-Since and From. There are no entity parts to the message as the request is to get an image (if the command had been to PUT then there would have been an attachment with the request). Notice that a single blank line delimits the end of the message as this indicates the end of a request/response. Note that the headers are case sensitive, thus Content-Type with the correct types of letters (and GET is always in uppercase letters).

An example is:

No.	Time	Source	Destination	Protocol	Info
3	0.000362	192.168.75.1	192.168.75.132	TCP	mgcp-gateway
> http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2 TSV=344415 TSER=0					
Frame 3 (74 bytes on wire, 74 bytes captured)					
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)					
Destination: Vmware 0f:71:a3 (00:0c:29:0f:71:a3)					
Source: Vmware c0:00:08 (00:50:56:c0:00:08)					
Type: IP (0x0800)					
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132 (192.168.75.132)					
Transmission Control Protocol, Src Port: mgcp-gateway (2427), Dst Port: http (80), Seq: 0, Len: 0					
4	0.000602	192.168.75.132	192.168.75.1	TCP	http > mgcp-gateway
[SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 WS=0 TSV=0 TSER=0					
Frame 4 (78 bytes on wire, 78 bytes captured)					
Ethernet II, Src: Vmware_0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)					
Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)					
Source: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)					
Type: IP (0x0800)					
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1 (192.168.75.1)					

```

Transmission Control Protocol, Src Port: http (80), Dst Port: mgcp-gateway (2427),
Seq: 0, Ack: 1, Len: 0

No.      Time          Source          Destination      Protocol Info
    5 0.000681    192.168.75.1    192.168.75.132  TCP        mgcp-gateway
> http [ACK] Seq=1 Ack=1 Win=66608 Len=0 TSV=344415 TSER=0

Frame 5 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3
(00:0c:29:0f:71:a3)
  Destination: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Source: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132
(192.168.75.132)
Transmission Control Protocol, Src Port: mgcp-gateway (2427), Dst Port: http (80),
Seq: 1, Ack: 1, Len: 0

No.      Time          Source          Destination      Protocol Info
    6 0.000835    192.168.75.1    192.168.75.132  HTTP       GET /
HTTP/1.1

Frame 6 (475 bytes on wire, 475 bytes captured)
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3
(00:0c:29:0f:71:a3)
  Destination: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Source: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132
(192.168.75.132)
Transmission Control Protocol, Src Port: mgcp-gateway (2427), Dst Port: http (80),
Seq: 1, Ack: 1, Len: 409
Hypertext Transfer Protocol

No.      Time          Source          Destination      Protocol Info
    7 0.055477    192.168.75.132  192.168.75.1    TCP        [TCP segment
of a reassembled PDU]

Frame 7 (1514 bytes on wire, 1514 bytes captured)
Ethernet II, Src: Vmware_0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware_c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1
(192.168.75.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: mgcp-gateway (2427),
Seq: 1, Ack: 410, Len: 1448

No.      Time          Source          Destination      Protocol Info
    8 0.055715    192.168.75.132  192.168.75.1    TCP        [TCP segment
of a reassembled PDU]

Frame 8 (1514 bytes on wire, 1514 bytes captured)
Ethernet II, Src: Vmware_0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware_c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1
(192.168.75.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: mgcp-gateway (2427),
Seq: 1449, Ack: 410, Len: 1448

No.      Time          Source          Destination      Protocol Info
    9 0.055759    192.168.75.1    192.168.75.132  TCP        mgcp-gateway
> http [ACK] Seq=410 Ack=2897 Win=66608 Len=0 TSV=344421 TSER=15586

Frame 9 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3
(00:0c:29:0f:71:a3)
  Destination: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Source: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132
(192.168.75.132)

```

```

Transmission Control Protocol, Src Port: mgcp-gateway (2427), Dst Port: http (80),
Seq: 410, Ack: 2897, Len: 0

No.      Time           Source            Destination      Protocol Info
  10  0.056010     192.168.75.132   192.168.75.1    HTTP      HTTP/1.1 200
OK (text/html)

Frame 10 (79 bytes on wire, 79 bytes captured)
Ethernet II, Src: Vmware 0f:71:a3 (00:0c:29:0f:71:a3), Dst: Vmware c0:00:08
(00:50:56:c0:00:08)
  Destination: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Source: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.132 (192.168.75.132), Dst: 192.168.75.1
(192.168.75.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: mgcp-gateway (2427),
Seq: 2897, Ack: 410, Len: 13
[Reassembled TCP Segments (2909 bytes): #7(1448), #8(1448), #10(13)]
Hypertext Transfer Protocol
Line-based text data: text/html

No.      Time           Source            Destination      Protocol Info
  11  0.090363     192.168.75.1     192.168.75.132   HTTP      GET /2.css
HTTP/1.1

Frame 11 (565 bytes on wire, 565 bytes captured)
Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_0f:71:a3
(00:0c:29:0f:71:a3)
  Destination: Vmware_0f:71:a3 (00:0c:29:0f:71:a3)
  Source: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Type: IP (0x0800)
Internet Protocol, Src: 192.168.75.1 (192.168.75.1), Dst: 192.168.75.132
(192.168.75.132)
Transmission Control Protocol, Src Port: mgcp-gateway (2427), Dst Port: http (80),
Seq: 410, Ack: 2910, Len: 499
Hypertext Transfer Protocol

```

View this file: <http://buchananweb.co.uk/log/webpage.txt>

The request/response sequence is then:

Client

```

GET / HTTP/1.1
User-Agent: Opera/3.80 (Windows NT 6.0; U; en) Presto/2.2.15 Version/10.01
Host: 192.168.75.132
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, im-
age/jpeg, image/gif, image/x-xbitmap, */*;q=0.1
Accept-Language: en-GB,en;q=0.9
Accept-Charset: iso-8859-1, utf-8, utf-16, *,q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *,q=0
Connection: Keep-Alive

HTTP/1.1 200 OK
Content-Length: 2606
Content-Type: text/html
Content-Location: http://192.168.75.132/iisstart.htm
Last-Modified: Sun, 13 Dec 2009 15:16:14 GMT
Accept-Ranges: bytes
ETag: "fc31243677cca1:745"
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Date: Sat, 02 Jan 2010 22:33:01 GMT
<HTML>
<HEAD>
<TITLE>SFC (Final Test)</TITLE>
<META http-equiv=Content-Type content="text/html; charset=iso-8859-1">
<LINK href="2.css" type=text/css rel=stylesheet>
<style type="text/css">
...
</HTML>

GET /2.css HTTP/1.1
User-Agent: Opera/3.80 (Windows NT 6.0; U; en) Presto/2.2.15 Version/10.01
Host: 192.168.75.132
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, im-
age/jpeg, image/gif, image/x-xbitmap, */*;q=0.1

```

```
Accept-Language: en-GB,en;q=0.9
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Referer: http://192.168.75.132/
Connection: Keep-Alive, TE
TE: deflate, gzip, chunked, identity, trailers
```

HTTP/1.1 200 OK

```
Content-Length: 14135
Content-Type: text/css
Last-Modified: Sun, 13 Dec 2009 15:15:09 GMT
Accept-Ranges: bytes
ETag: "744c36f77ccal:745"
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Date: Sat, 02 Jan 2010 22:33:01 GMT
...H1
{font: bold 16pt Verdana, Arial, Helvetica, sans-serif;
background: transparent;
```

It can be seen that the data format that the client and the server can accept are identified in the header sent.

3.15 Network logs on hosts

Captured network packets are useful for analyzing systems in real-time, but often malicious activity can take place over long intervals. It is thus difficult to analyze a trail of evidence of network packets over a relatively long period of time. Most systems, though, have audit logs which can provide evidence of activities. In Windows the Web server stores its log at:

```
C:\WINDOWS\system32\LogFiles
```

For the instance of Web instance of W3SVC1, a sample log is:

```
#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2010-01-02 22:29:25
#Fields: date time s-sitename s-ip cs-method cs-uri-stem cs-uri-query s-port cs-
username c-ip cs(User-Agent) sc-status sc-substatus sc-win32-status
2010-01-02 22:29:25 W3SVC1 192.168.75.132 GET /iisstart.htm - 80 - 192.168.75.1 Mozil-
la/5.0+(Windows;+U;+Windows+NT+6.0;+en-
US;+rv:1.8.1.20)+Gecko/20081217+Firefox/2.0.0.20 200 0 0
2010-01-02 22:29:25 W3SVC1 192.168.75.132 GET /2.css - 80 - 192.168.75.1 Mozil-
la/5.0+(Windows;+U;+Windows+NT+6.0;+en-
US;+rv:1.8.1.20)+Gecko/20081217+Firefox/2.0.0.20 200 0 0
2010-01-02 22:29:25 W3SVC1 192.168.75.132 GET /favicon.ico - 80 - 192.168.75.1 Mozil-
la/5.0+(Windows;+U;+Windows+NT+6.0;+en-
US;+rv:1.8.1.20)+Gecko/20081217+Firefox/2.0.0.20 404 0 2
2010-01-02 22:29:35 W3SVC1 192.168.75.132 GET /iisstart.htm - 80 - 192.168.75.1 Mozil-
la/5.0+(Windows;+U;+Windows+NT+6.0;+en-
US;+rv:1.8.1.20)+Gecko/20081217+Firefox/2.0.0.20 304 0 0
2010-01-02 22:29:35 W3SVC1 192.168.75.132 GET /2.css - 80 - 192.168.75.1 Mozil-
la/5.0+(Windows;+U;+Windows+NT+6.0;+en-
US;+rv:1.8.1.20)+Gecko/20081217+Firefox/2.0.0.20 304 0 0
2010-01-02 22:33:01 W3SVC1 192.168.75.132 GET /iisstart.htm - 80 - 192.168.75.1
Opera/3.80+(Windows+NT+6.0;+U;+en)+Presto/2.2.15+Version/10.01 200 0 0
2010-01-02 22:33:01 W3SVC1 192.168.75.132 GET /2.css - 80 - 192.168.75.1
Opera/3.80+(Windows+NT+6.0;+U;+en)+Presto/2.2.15+Version/10.01 200 0 0
2010-01-02 22:33:01 W3SVC1 192.168.75.132 GET /favicon.ico - 80 - 192.168.75.1
Opera/3.80+(Windows+NT+6.0;+U;+en)+Presto/2.2.15+Version/10.01 404 0 2
```

Where it can be seen that the host 192.168.75.132 has been accessing Web pages on

the server (192.168.75.1). It can also be seen that the accesses have been from Firefox and Presto (Opera). For FTP, we can access the instance of an FTP server (\MSFTPSVC1) gives:

```
#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2010-01-04 19:36:08
#Fields: time c-ip cs-method cs-uri-stem sc-status sc-win32-status
19:36:08 192.168.75.138 [2]closed - 426 10054
20:18:05 192.168.75.1 [3]USER Administrator 331 0
20:18:05 192.168.75.1 [3]PASS - 230 0
20:18:44 192.168.75.1 [3]created index_asfc.html 550 5
20:19:21 192.168.75.1 [3]QUIT - 226 0
```

Where the client IP address and the requests are defined in the log file. Error logs are also an important place to look for maliciousness.

In Linux, the /var/log folder contains a host of log files. For example, the Apache Web server stores its log files in:

/var/log/apache2

An example of the access file is:

```
napier@ubuntu:/var/log/apache2$ cat access.log.1
192.168.75.1 - - [29/Dec/2009:09:09:25 -0800] "GET / HTTP/1.1" 200 471 "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.8.1.20) Gecko/20081217 Firefox/2.0.0.20"
192.168.75.132 - - [30/Dec/2009:11:42:37 -0800] "GET / HTTP/1.0" 200 430 "-" "-"
192.168.75.132 - - [30/Dec/2009:11:42:39 -0800] "GET / HTTP/1.1" 200 430 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)"
192.168.75.132 - - [30/Dec/2009:11:42:39 -0800] "GET /robots.txt HTTP/1.1" 404 491 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)"
192.168.75.132 - - [30/Dec/2009:11:42:39 -0800] "GET /favicon.ico HTTP/1.1" 404 492 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)"
```

In Linux many of the system messages are stored to messages, or syslog. For example from messages:

```
Jan  4 13:07:51 ubuntu ftpd[2044]: connection from bills.local
Jan  4 13:08:02 ubuntu ftpd[2044]: FTP LOGIN FROM bills.local as napier
```

3.16 Tripwire

Tripwire is a useful method of watching key file and auditing their changes. In Ubuntu a profile is created by editing:

```
/usr/tripwire/twpol.txt
```

This is then is compiled into an encrypted policy file with (and produces tw.cfg):

```
twadmin --create-polfile --cfgfile ./tw.cfg --site-keyfile ./site.key
```

```
./twpol.txt
```

of which the database is created with (using the tw.pol file):

```
tripwire --init --cfgfile /etc/tripwire/tw.cfg --polfile
/etc/tripwire/tw.pol --site-keyfile /etc/tripwire/site.key --local-keyfile
/etc/tripwire/ubuntu-local.key
```

Then using:

```
tripwire --check
```

produces a report of the system changes:

```
Database file used: /var/lib/tripwire/ubuntu.twd
Command line used: tripwire --check

=====
Rule Summary:
=====

-----
Section: Unix File System
-----

Rule Name                Severity Level   Added   Removed  Modified
-----
Invariant Directories    66              0       0         0
* Tripwire Data Files    100             1       0         0
Other binaries           66              0       0         0
Tripwire Binaries        100             0       0         0
Other libraries          66              0       0         0
Root file-system executables 100             0       0         0
System boot changes      100             0       0         0
Root file-system libraries 100             0       0         0
(/lib)
Critical system boot files 100             0       0         0
* Other configuration files 66              0       0         2
(/etc)
Boot Scripts             100             0       0         0
Security Control         66              0       0         0
Root config files        100             0       0         0
* Devices & Kernel information 100            159     155       0

Total objects scanned: 70781
Total violations found: 317

=====
Object Summary:
=====

-----
# Section: Unix File System
-----

Rule Name: Tripwire Data Files (/var/lib/tripwire/ubuntu.twd)
Severity Level: 100
-----

Added:
"/var/lib/tripwire/ubuntu.twd"

-----
Rule Name: Other configuration files (/etc)
Severity Level: 66
-----

Modified:
```



```
"/etc/tripwire"  
"/etc/tripwire/list"
```

Where we can see that the file "list" has been changed. A sample rule is given next, where Tripwire watches the passwd and shadow files:

```
(  
  rulename = "Security Control",  
  severity = $(SIG_MED)  
)  
{  
  /etc/passwd    -> $(SEC_CONFIG) ;  
  /etc/shadow    -> $(SEC_CONFIG) ;  
}
```

When the /etc/passwd file changes it results in:


```
-----  
Rule Name: Security Control (/etc/passwd)  
Severity Level: 66  
-----  
  
Modified:  
"/etc/passwd"
```

Tripwire Demo Link:

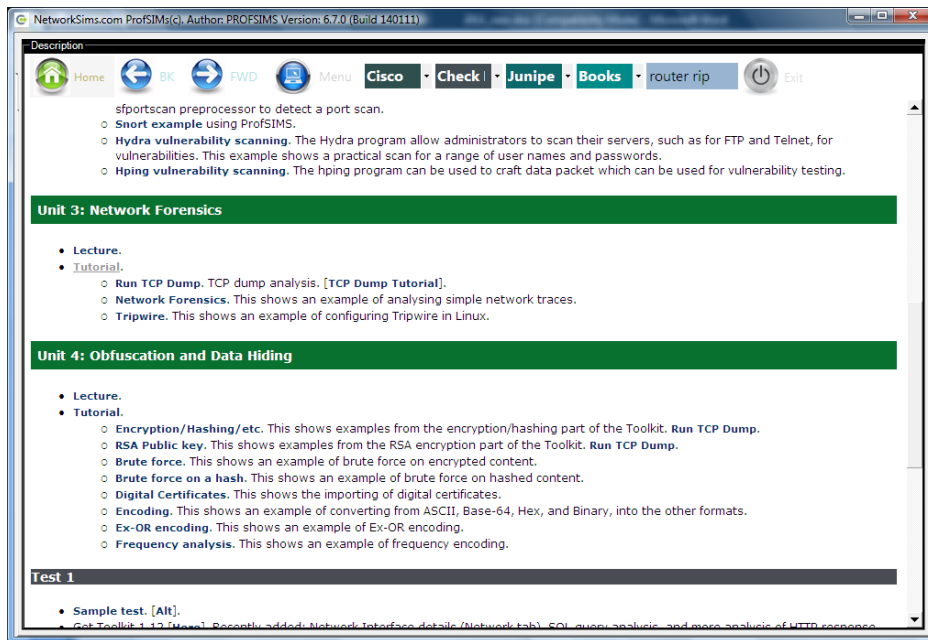
http://buchananweb.co.uk/adv_security_and_network_forensics/tripwire/

3.17 Tutorial

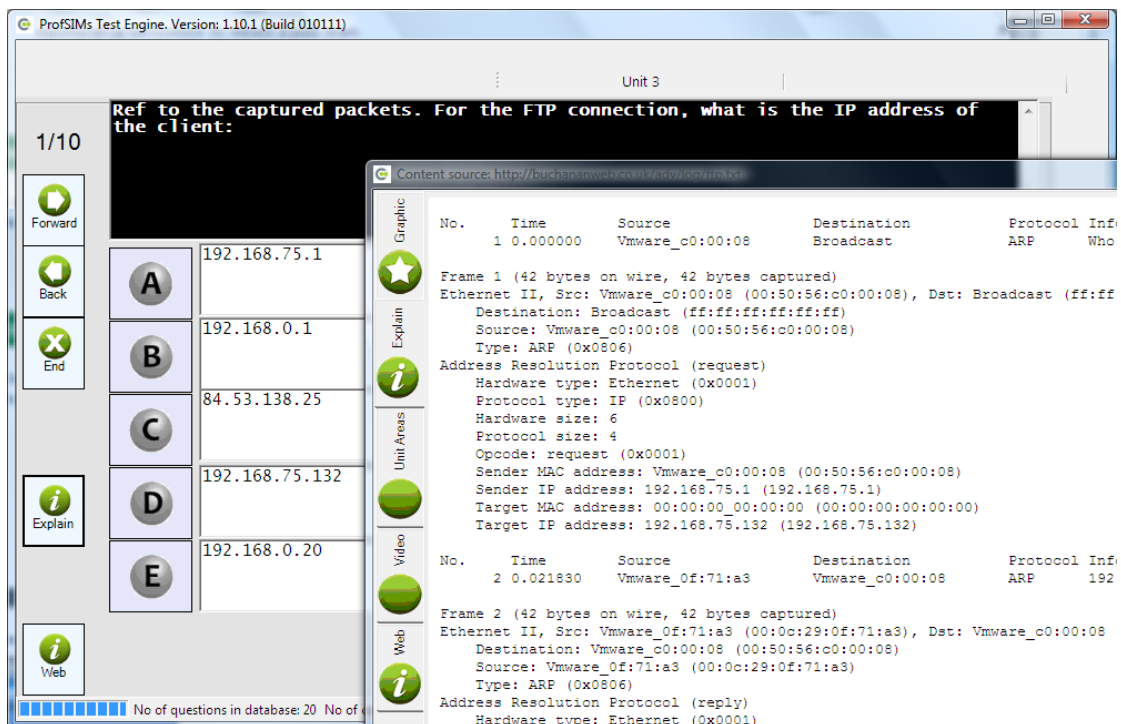
The main tutorial is at:

 On-line tutorial: <http://buchananweb.co.uk/adv03.html>

The main tutorial is in NetworkSims ProfSIMs at:



And then select the Tutorial to give:



4 Data Hiding and Obfuscation

On-line lecture: <http://buchananweb.co.uk/adv/unit04.html>

4.1 Objectives

The key objectives of this unit are to:

- Outline obfuscation methods.
- Define methods used to encode data in order to hide its original content.
- Understand encryption methods used to hide data, and possible methods to overcome this obfuscation.
- Define how file types can be discovered.

4.2 Introduction

This unit provides an outline in some of the methods that a suspect may use to hide their tracks. Hiding information has existed for many decades in many different forms. In fact steganography, which is the science of hiding information within content, has been arranged for thousands of years, and includes using invisible inks and to hide information. Another method of hiding information is to embed it into messages, such as in:

Let everyone tango. This has Edward's mind in
some simple inquiry of nothing, before everyone
gets into Nirvana.

which, when each of the starting characters is taken, gives the message of **Let the mission begin**. This type of hidden information is known as a covert channel where information is added through a communications channel which is was not intended for. Other covert channels have included, in the past: Passing a briefcase in a busy place; Hiding microfilms in objects; and using templates for typewritten text. Unfortunately as we move into the Information Age, the places that covert channels can exist increases by the day and it can often be difficult to detect this type of communication in electronic transmissions.

Figure 4.1 shows the main classifications for information hiding, including the use of:

- **Covert channels**. This is used a communication channel for a purpose that it was not intended for (Llamas, 2004).
- **Steganography**. This is the methods used to hide information in content that only the recipient knows where to look for the hidden information.
- **Anonymity**. This is the methods used to hide the original source of the information.

- **Copyright marking.** This typically involves embedded information, normally which is hidden with content.

The requirements for copyright marking is obviously a growing issue, as many content creators, such as musicians, artists, and so on, are keen to preserve their copyright on content. It is, though, a constant challenge, as many methods of copyrighting are normally flawed in some sort of way that means that copyright protection can often be overcome. The challenge is sometimes to preserve the copyright in some way which is invisible to the user, but can be revealed when required.

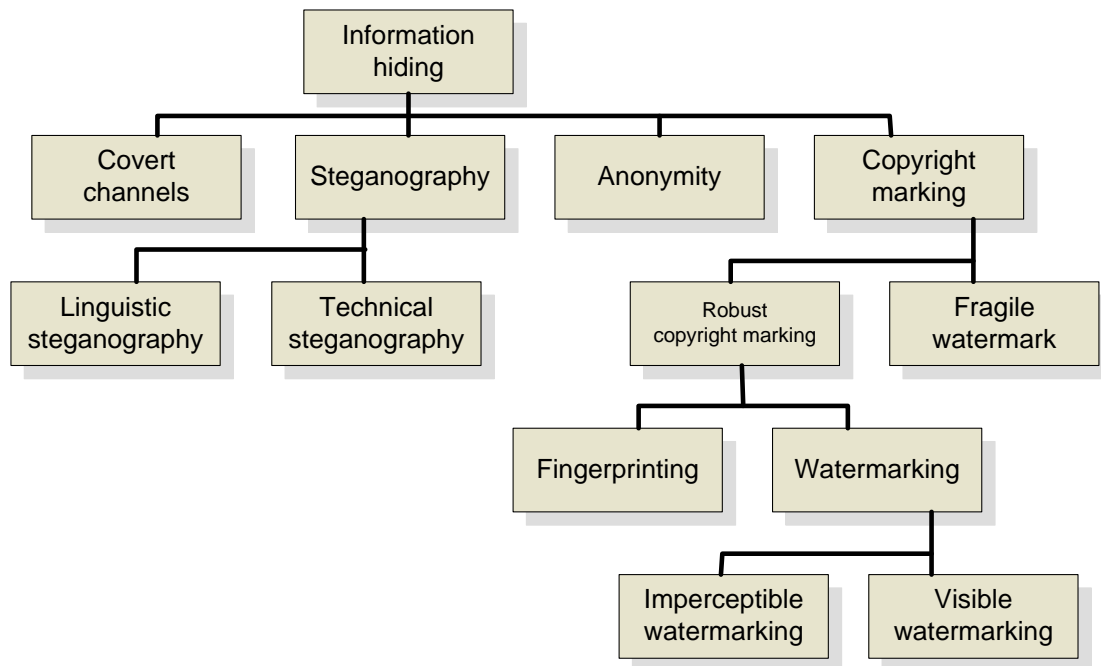


Figure 4.1 Information hiding classifications

4.3 Obfuscation using Encryption

One method of obfuscating data is to encrypt or to encode the data in a non-clear format. The main methods that can be used include:

- **Private key encryption.** With private key a secret key is used to encrypt the data. To decrypt the original key must be found. Normally, though, the encryption key is generated through a password generate program, thus the range of actual encryption keys used can thus be limited to a search of well-known phrases. Typical private-key encryption methods are DES, 3DES and AES.
- **Public key encryption.** With public-key encryption the data is encrypted with one key (normally the public key) and a private key is used to decrypt the ciphertext. A typical public-key method is RSA.
- **Hashing.** Hashing normally involves a one-way hashing function, where it is difficult to reverse the hashing. Some form of dictionary lookup is normally used to try and determine the original data.

- **Encoding.** This normally involves obfuscating messages by converting them into a non-readable format. Typical methods used include converting in Base-64 and also using an X-OR with a passphrase.

Private-key data hiding

Private key involves using the same key to encrypt as to decrypt. It is often used in encryption as it is fairly fast, and it does not need the same processing power of public key encryption. It can thus be supported on a wide range of devices. The typical ways to decrypt private-key encryption is:

- **Search for key strings.** With this method a scan is made of the host machine to find all the string that have been used in other types of access, such as for Internet Explorer passwords. These are the most likely ones that could lead to a successful decrypt.
- **Use a dictionary.** The next quickest method to find an encryption key is to use a standard dictionary to determine the key that it data has been encrypted with.
- **Perform a brute-force.** If the first two methods fail, a brute force can be conducted which will search through the key space.

In Figure 4.2 it can be seen that a word named “fred” has been encrypted with the key word of “apples”, to produce a ciphertext of “2A699...A04”. A search is then conducted from words in the dictionary, where an exception is caused if the encryption process fails. This results in a number of possible encryption keys. In this case, these are “anyway”, “apples”, “assembler”, and so on. It can be seen that “apples” is the only one which produces a sensible decryption.

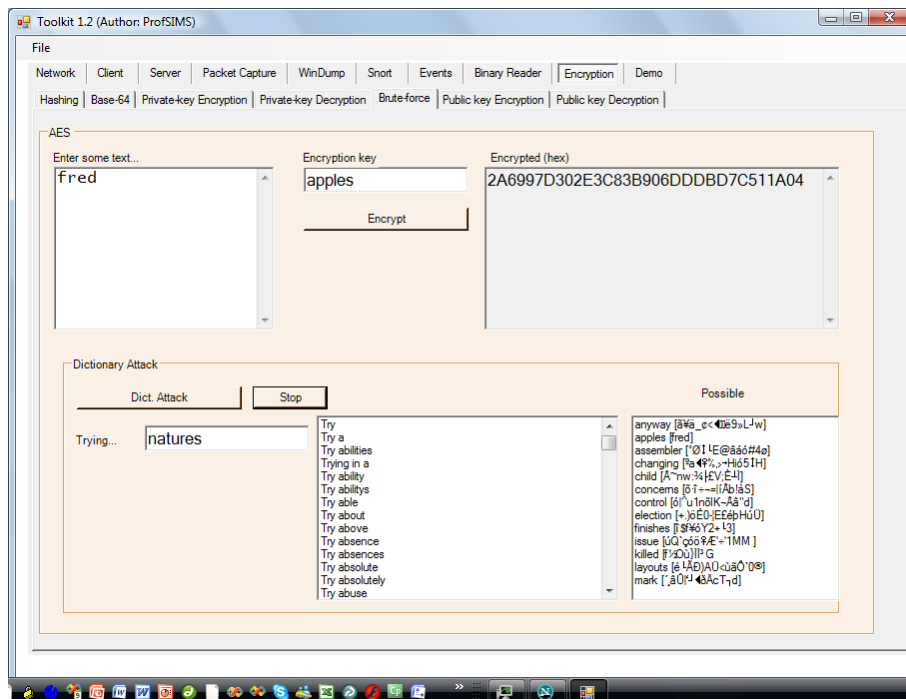


Figure 4.2 Dictionary search

http://buchananweb.co.uk/adv_security_and_network_forensics/dotnetclient_brute/dotnetclient_brute.htm

Public-key data hiding

Public key methods, such as with RSA, involve a different decryption key from the encryption one. These are known as a key pair. The key sizes tend to be fairly large as compared with private key methods (typically more than 1,024 bits, as opposed to 128/256 bit sizes for AES). It is thus extremely difficult to perform a brute force attack on the private key. The normal method is to try and determine the digital certificate which stores the public key and the private key. In Figure 4.3, it can be seen that the certificate on the left-hand side only contains the public key, whereas the one of the right-hand side contains both the public and the private key. Normally this certificate is protected by a password, thus the certificate can be opened using a dictionary or brute force search. A typical format for a certificate with a password is PFX.

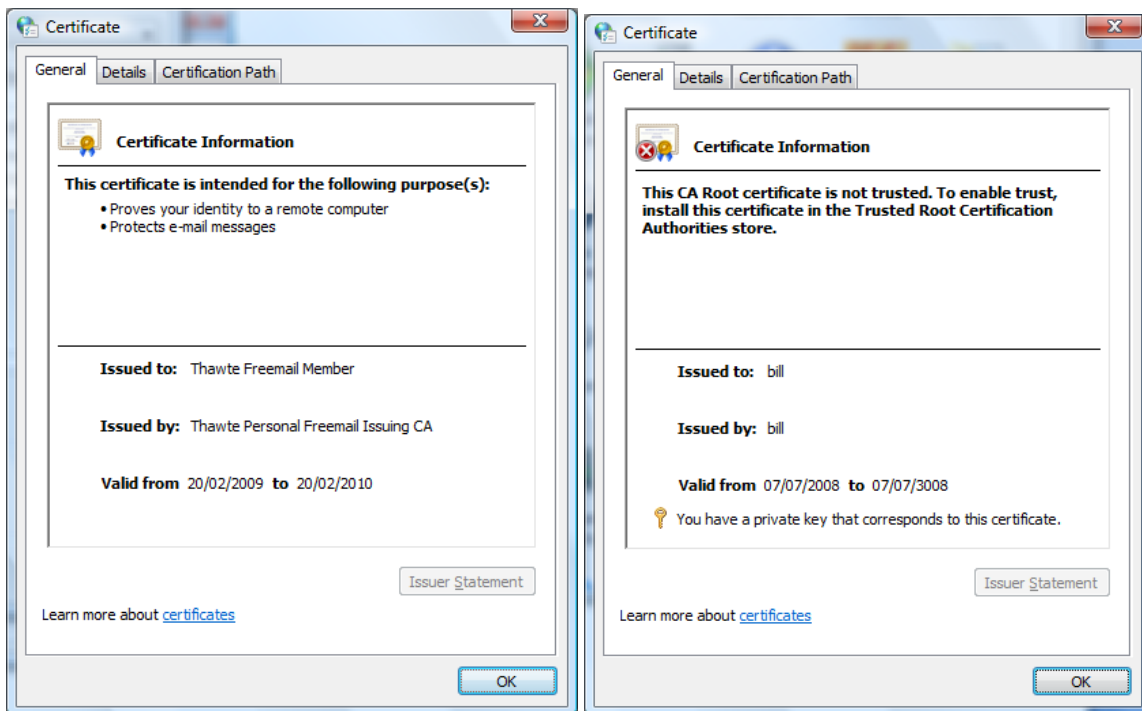


Figure 4.3 Digital certificates

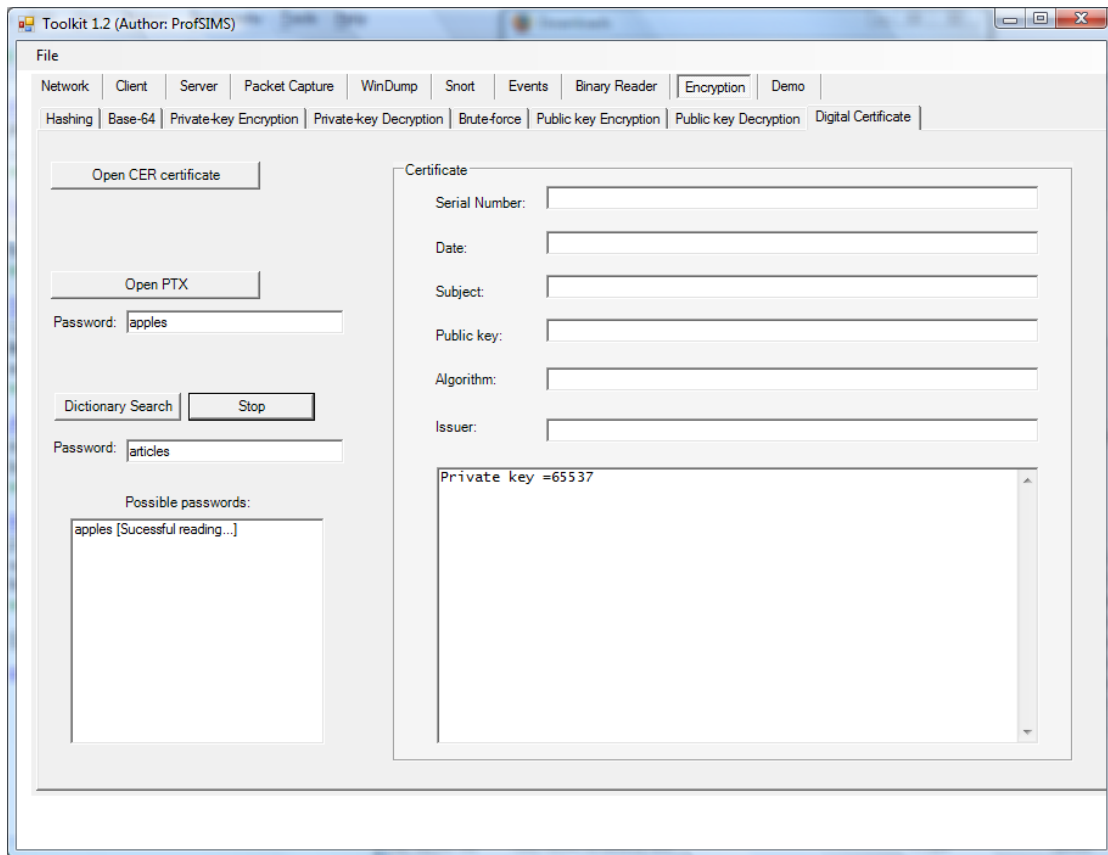


Figure 4.4 Searching for a password on a certificate

http://buchananweb.co.uk/adv_security_and_network_forensics/dotnetclient_digitalcert/dotnetclient_digitalcert.htm

Hashing

Hashing can be used to store messages, using a one-way encryption process. It is almost impossible to determine the original message from a hashed version, unless there is a dictionary for well-known hash functions. For example, "test" gives:

098F6BCD4621D373CADE4E832627B4F6

In this way a secret message can be kept in a hash format. A way to change the hash is to apply salt, where the hash varies based on a number of known keywords. For example:

```
Password="test";
Salt=One of {"fred","bert","ken"}
Hash = md5>Password.Salt;
```

The mdcrack program can be used to reverse the process, such as:

```
C:\test> mdcrack --algorithm=MD5 098F6BCD4621D373CADE4E832627B4F6
System / Starting MDCrack v1.8(3)
System / Running as mdcrack-sse --algorithm=MD5 098F6BCD4621D373CADE4E832627B4F6
```

```

System / Charset is: abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNQRSTUUVW
XYZ
System / Detected processor(s): 2 x INTEL Itanium | MMX | SSE | SSE2 | SSE3
System / Target hash: 098F6BCD4621D373CADE4E832627B4F6
System / >> Using MD5 cores: maximal candidate/user salt size: 16/54 bytes
Info / Press ESC for available runtime shortcuts (Ctrl-c to quit)
Info / Thread #0: >> Using Core 1
Info / Thread #0: Candidate size: 1 ( + user salt: 0 )
Info / Thread #0: Candidate size: 2 ( + user salt: 0 )
Info / Thread #0: Candidate size: 3 ( + user salt: 0 )
Info / Thread #1: >> Using Core 1
Info / Thread #1: Candidate size: 1 ( + user salt: 0 )
Info / Thread #1: Candidate size: 2 ( + user salt: 0 )
Info / Thread #1: Candidate size: 3 ( + user salt: 0 )
Info / Thread #0: Candidate size: 4 ( + user salt: 0 )
Info / Thread #1: Candidate size: 4 ( + user salt: 0 )
-----/ Thread #1 (Success)
\----
System / Thread #1: Collision found: test
Info / Thread #1: Candidate/Hash pairs tested: 2 341 902 ( 2.34e+006 ) in 812m
s
Info / Thread #1: Allocated key space: 2.42e+028 candidates, 0.00% done
Info / Thread #1: Average speed: ~ 2 884 116 ( 2.88e+006 ) h/s

```

This takes less than two seconds to run, while longer text sequences take much longer.

Encoding

There are many standard for encoding data from one format to another. One of the most common is Base-64, which is used to convert from an 8-bit format into 6-bit values, which are converted to Base-64 characters. The table for the conversion is given in Table 4.1.

Figure 4.1 Base-64 conversion

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	
15	P	31	f	47	v	63	/

For example:

“What’

Results in:

“ 00100010
W 01010111


```

h 01101000
a 01100001
t 01110100
' 00100111

```

Which gives: 00100010010101110110100001100001011101000010011101
001000 100101 011101 101000 011000 010111 010000 100111
I l d o Y X Q n

The conversion thus becomes:

ASCII: "What's in a name? That which we call a rose. By any other name would
smell as sweet."

Base-64: IldoYXQncyBpbiBhIG5hbWU/IFRoYXQgd2hpY2ggd2UgY2Fs
bCBhIHJvc2UuIEJ5IGFueSBvdGhlciBuYW1lIHdvdWxkIHhtZ
WxsIGFzIHN3ZWV0LiI=

Hex: 2257686174277320696E2061206E616D653F2054686174207768696368
2077652063616C6C206120726F73652E20427920616E79206F74686572206E6
16D6520776F756C6420736D656C6C2061732073776565742E22

Binary: 00100010 ... 000010111000100010

http://buchananweb.co.uk/adv_security_and_network_forensics/dotnet_base64/dotnet_base64.htm

Ex-OR encoding

The Ex-OR operator is used in many applications in data hiding and encryption, especially as it does not lose any information within the bit stream. Its basic operation is:

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

The main advantage of Ex-OR is that a bit stream when Ex-OR'ed with a given value will result in the same value when it is Ex-OR'ed again. For example, if the text message is "Hello", then the bit stream will be:

H	e	l	l	o
01001000	01100101	01101100	01101100	01101111

If we Ex-OR this with a bitvalue of 0101 0101 ('U') we get:

01001000	01100101	01101100	01101100	01101111
01010101	01010101	01010101	01010101	01010101
00011101	00110000	00111001	00111001	00111010

(1D, 30, 39, 3A)

And if we Ex-OR this with the same value:

00011101	00110000	00111001	00111001	00111010
01010101	01010101	01010101	01010101	01010101
01001000	01100101	01101100	01101100	01101111

Which results in the original value.

http://buchananweb.co.uk/adv_security_and_network_forensics/dotnet_xor/dotnet_xor.htm

Coding

There are obviously an infinite amount of ways that someone can hide or pass secret information using their own standard codes. Alphabet shifting is an example of this, where the alphabet is shifted by a given number of space, such as for a three letter shift:

Input: abcdefghijklmnopqrstuvwxyz
Output: DEFGHIJKLMNOPQRSTUVWXYZABC

Where "fred" would give "IUHG". Unfortunately this type of code is relatively easy to crack, as there are only 25 unique mappings. A more robust code is to randomly assign the letters, such as for:

In Chapter 1 the concept of defence-in-depth was discussed, where a defence system has many layers of defence. Unfortunately, as in military systems, it is not always possible to protect using front-line [... text missed out ...] where intrusion detection agents are used to listen to network traffic, and network/user activity to try and detect any breaches in security.

And using the mapping of:

Code A

abcdefghijklmnopqrstuvwxyz
BIHOQKWCDVLEJSRGXFAUTMYPZ

Becomes (Figure 4.5):

DS HCBGUQF 1 UCQ HRSHQGU RK OQKQSHQ-DS-OQGUC YBA ODAHTAAQO, YCQFQ B OQKQSHQ APAUQJ CBA JBSP EBPQFA RK OQKQSHQ. TSKRFUTSBUQEP, BA DS JDEDUBFP APAUQJA, DU DA SRU BEYBPA GRAADIEQ UR GFRUQHU TADSW KFRSU-EDSQ OQKQSHQA, QMQS DK UCQFQ BFQ JTEUDGEQ EBPQFA RK UCQJ, BWBDSAU IFQBHCQA DS AQHTFDUP (KDWTFQ 2.2). UCDA HBS IQ IQHBTAQ BS DSUFTOQF CBA KRTSO B YQBLAQAA YDUCDS UCQ [... text missed out ...] HRSHQGU, YCQFQ DSUFTADRS OQUQHURDS BWQSUA BFQ TAQO UR EDAUQS UR SQUYRFL UFBKKDH, BSO SQUYRFL/TAQF BHUDMDUP UR UFP BSO OQUQHU BSP IFQBHCQA DS AQHTFDUP.

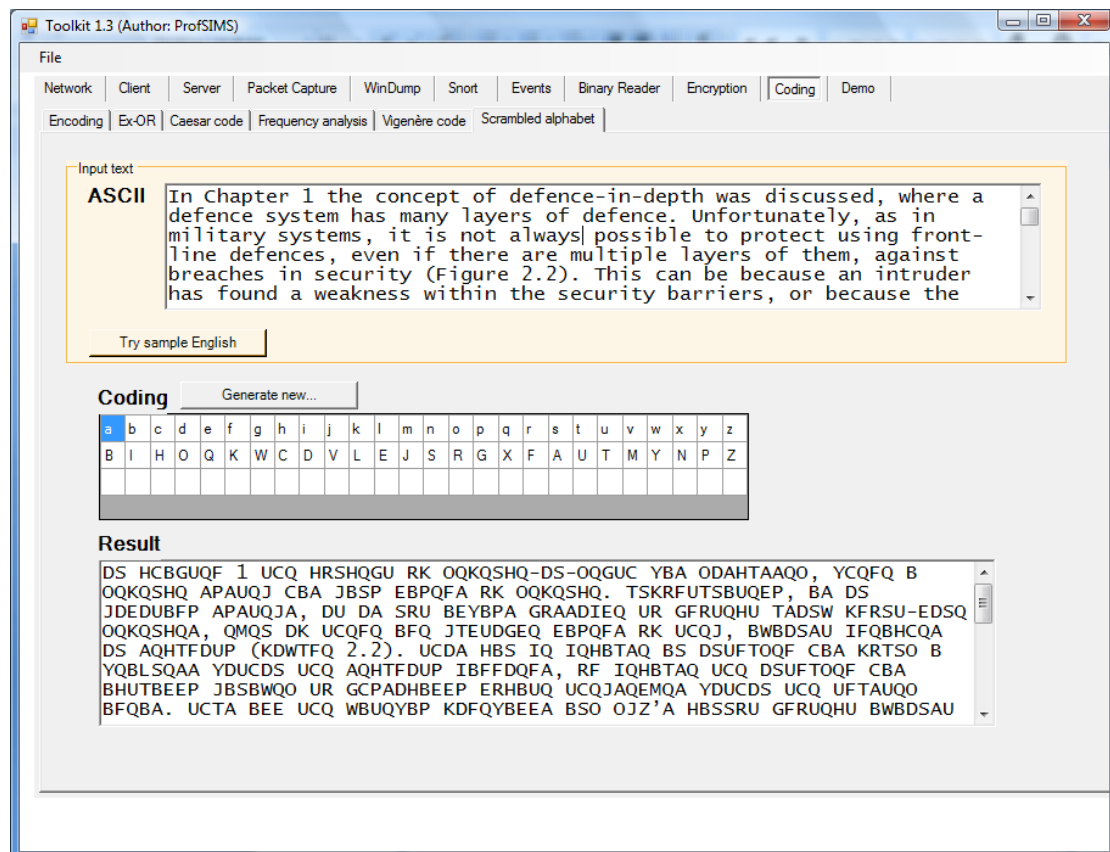


Figure 4.5 Searching for a password on a certificate

In standard English text, some letters are more probable than others, such the most popular is "E", and the least popular is "Z". In the following the coded text probability from the previous example has been mapped to the most probable letters to give (Figure 4.6):

Code B

E T O A N I R S H D L C F U M P Y W G B V K X J Q Z
q u b a s d f r c h e p t o k y w j g l m i z x v n

If we refer to the before, then the 'q' is the most popular letter, which has successfully determined the mapping (see Code A). The next most popular letter is a 'u', which maps to a 'T', which again is correct and just with these two letters gives:

DS HCBGtE 1 tCe HRSHeGt RK OeKeSHe-DS-OeGtC YBA ODAHTAAeO

After this it is normally a matter of moving the letters around, and identifying common words. For example, it can be seen that the four work is likely to be “The”, thus a “C” could map to an “h” to give:

DS HhBGtE 1 the HRSHeGt RK OeKeSHe-DS-OeGtC YBA ODAHTAAeO

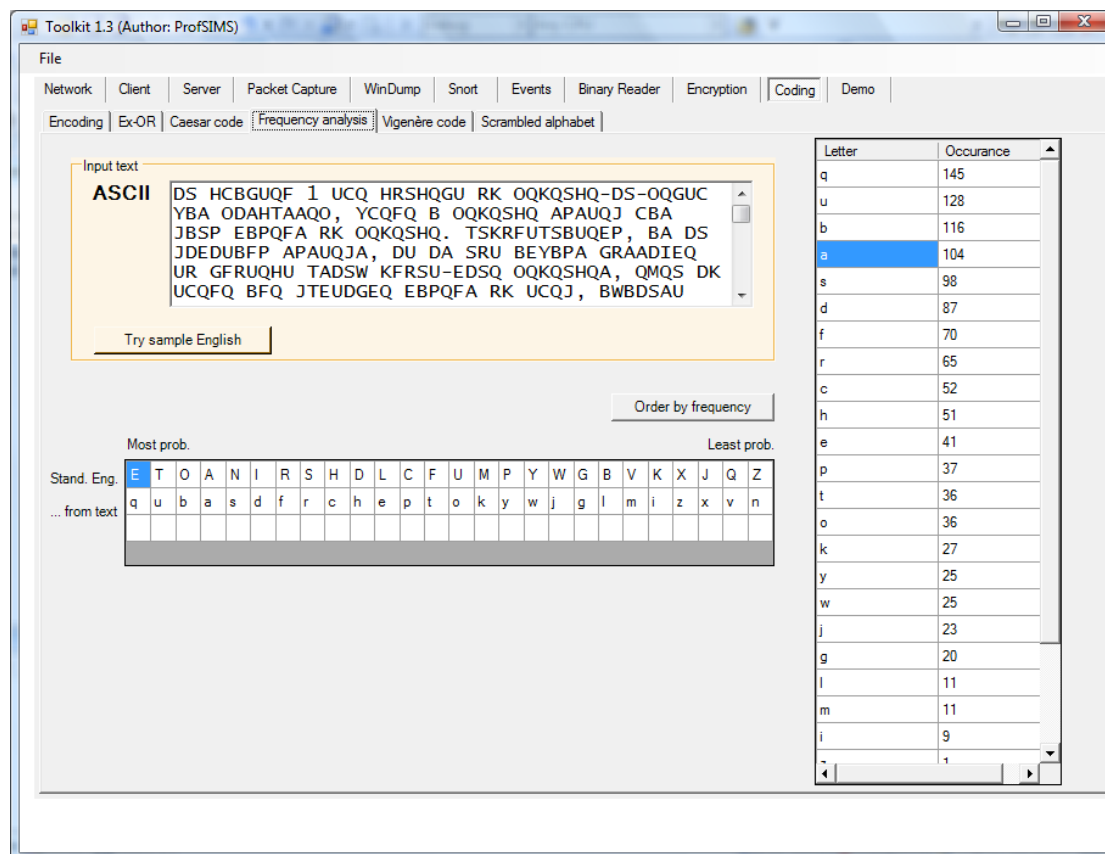


Figure 4.6 Statistical analysis

4.4 Obfuscation through tunneling

One method used to hide communications is to tunnel the information either through an encryption tunnel, or through another protocol. For an encryption tunnel the two ends of the tunnel negotiate their encryption keys, and the communications will then be encrypted for the session. Thus any listening devices will not be able to decrypt the content, as they do not have the encryption keys required to decrypt the message. The two main methods used to create a tunnel are IPSec and SSL. With IPSec the start of the connection is identified with a connection on UDP Port 500, such as:

No.	Time	Source	Destination	Protocol	Info
6	5.007300	192.168.0.20	146.176.24.2	ISAKMP	Aggressive

User Datagram Protocol, Src Port: 65341 (65341), Dst Port: isakmp (500)

```

Source port: 65341 (65341)
Destination port: isakmp (500)
Length: 884
Checksum: 0xa205 [correct]
  [Good Checksum: True]
  [Bad Checksum: False]
Internet Security Association and Key Management Protocol
Initiator cookie: 0490174339C81264
Responder cookie: 0000000000000000
Next payload: Security Association (1)
Version: 1.0
Exchange type: Aggressive (4)
Flags: 0x00
  .... ..0 = Not encrypted
  .... ..0 = No commit
  .... .0.. = No authentication
Message ID: 0x00000000
Length: 860
Security Association payload
  Next payload: Key Exchange (4)
  Payload length: 556
  Domain of interpretation: IPSEC (1)
  Situation: IDENTITY (1)
  Proposal payload # 1
    Next payload: NONE (0)
    Payload length: 544
    Proposal number: 1
    Protocol ID: ISAKMP (1)
    SPI Size: 0
    Proposal transforms: 14
    Transform payload # 1
      Next payload: Transform (3)
      Payload length: 40
      Transform number: 1
      Transform ID: KEY_IKE (1)
      Encryption-Algorithm (1): AES-CBC (7)
      Hash-Algorithm (2): SHA (2)
      Group-Description (4): Alternate 1024-bit MODP group (2)
      Authentication-Method (3): XAUTHInitPreShared (65001)
      Life-Type (11): Seconds (1)
      Life-Duration (12): Duration-Value (2147483)
      Key-Length (14): Key-Length (256)
  Key Exchange payload
    Next payload: Nonce (10)
    Payload length: 132
    Key Exchange Data (128 bytes / 1024 bits)
  Nonce payload
    Next payload: Identification (5)
    Payload length: 24
    Nonce Data
  Identification payload
    Next payload: Vendor ID (13)
    Payload length: 24
    ID type: 11
    ID type: KEY_ID (11)
    Protocol ID: UDP (17)
    Port: 500
    Identification Data
  Vendor ID: draft-beaulieu-ike-xauth-02.txt
    Next payload: Vendor ID (13)
    Payload length: 12
    Vendor ID: draft-beaulieu-ike-xauth-02.txt
  Vendor ID: RFC 3706 Detecting Dead IKE Peers (DPD)
    Next payload: Vendor ID (13)
    Payload length: 20
    Vendor ID: RFC 3706 Detecting Dead IKE Peers (DPD)
  Vendor ID: Cisco Fragmentation
    Next payload: Vendor ID (13)
    Payload length: 24
    Vendor ID: Cisco Fragmentation
  Vendor ID: draft-ietf-ipsec-nat-t-ike-02\n
    Next payload: Vendor ID (13)
    Payload length: 20
    Vendor ID: draft-ietf-ipsec-nat-t-ike-02\n
  Vendor ID: CISCO-UNITY-1.0
    Next payload: NONE (0)

```

```

Payload length: 20
Vendor ID: CISCO-UNITY-1.0

No.      Time          Source          Destination      Protocol Info
  7 5.312130    146.176.24.2   192.168.0.20    ISAKMP Aggressive
User Datagram Protocol, Src Port: isakmp (500), Dst Port: 65341 (65341)
Source port: isakmp (500)
Destination port: 65341 (65341)
Length: 456
Checksum: 0x5907 [correct]
  [Good Checksum: True]
  [Bad Checksum: False]
Internet Security Association and Key Management Protocol
Initiator cookie: 0490174339C81264
Responder cookie: F4B6486D172C028B
Next payload: Security Association (1)
Version: 1.0
Exchange type: Aggressive (4)
Flags: 0x00
  .... ...0 = Not encrypted
  .... ..0. = No commit
  .... .0.. = No authentication
Message ID: 0x00000000
Length: 448
Security Association payload
  Next payload: Key Exchange (4)
  Payload length: 56
  Domain of interpretation: IPSEC (1)
  Situation: IDENTITY (1)
  Proposal payload # 1
    Next payload: NONE (0)
    Payload length: 44
    Proposal number: 1
    Protocol ID: ISAKMP (1)
    SPI Size: 0
    Proposal transforms: 1
    Transform payload # 10
      Next payload: NONE (0)
      Payload length: 36
      Transform number: 10
      Transform ID: KEY_IKE (1)
      Encryption-Algorithm (1): 3DES-CBC (5)
      Hash-Algorithm (2): MD5 (1)
      Group-Description (4): Alternate 1024-bit MODP group (2)
      Authentication-Method (3): XAUTHInitPreShared (65001)
      Life-Type (11): Seconds (1)
      Life-Duration (12): Duration-Value (2147483)
  Key Exchange payload
    Next payload: Nonce (10)
    Payload length: 132
    Key Exchange Data (128 bytes / 1024 bits)
  Nonce payload
    Next payload: Identification (5)
    Payload length: 24
    Nonce Data
  Identification payload
    Next payload: Hash (8)
    Payload length: 12
    ID type: 1
    ID type: IPV4 ADDR (1)
    Protocol ID: UDP (17)
    Port: Unused
    Identification data: 146.176.24.2
  Hash payload
    Next payload: Vendor ID (13)
    Payload length: 20
    Hash Data
  Vendor ID: CISCO-UNITY-1.0
    Next payload: Vendor ID (13)
    Payload length: 20
    Vendor ID: CISCO-UNITY-1.0
  Vendor ID: draft-beaulieu-ike-xauth-02.txt
    Next payload: Vendor ID (13)
    Payload length: 12
    Vendor ID: draft-beaulieu-ike-xauth-02.txt
  Vendor ID: RFC 3706 Detecting Dead IKE Peers (DPD)

```

```

Next payload: Vendor ID (13)
Payload length: 20
Vendor ID: RFC 3706 Detecting Dead IKE Peers (DPD)
Vendor ID: draft-ietf-ipsec-nat-t-ike-02\n
Next payload: NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) (130)
Payload length: 20
Vendor ID: draft-ietf-ipsec-nat-t-ike-02\n
NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) payload
Next payload: NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) (130)
Payload length: 20
Hash of address and port: A9D9C6CAEA2D34812E57F925DC636F98
NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) payload
Next payload: Vendor ID (13)
Payload length: 20
Hash of address and port: 4F38ED224B394682D4F05FF14D6F34AF
Vendor ID: Microsoft L2TP/IPSec VPN Client
Next payload: Vendor ID (13)
Payload length: 24
Vendor ID: Microsoft L2TP/IPSec VPN Client
Vendor ID: 0171EF70172D028B237401446015B2D0
Next payload: Vendor ID (13)
Payload length: 20
Vendor ID: 0171EF70172D028B237401446015B2D0
Vendor ID: 1F07F70EAA6514D3B0FA96542A500407
Next payload: NONE (0)
Payload length: 20
Vendor ID: 1F07F70EAA6514D3B0FA96542A500407Covert channels

```

It is through this phase that the main encryption parameters are negotiated.

4.5 Covert channels

A covert channel is a communication channel that allows two cooperating processes to transfer information in a manner that violates the system's security policy (Berg 1998). It is thus a way of communicating which is not part of the original design of the system, but can be used to transfer information to a process or user that, a priori, would not be authorised to access to that information. Covert channels only exist in systems with multilevel security (Proctor and Neumann 1992), which contain and manage information with different sensitivity levels. This it allows different users to access to the same information, at the same time, but from different points-of-view, depending on their requirements to know and their access privileges. The covert channel concept was introduced in 1973 (Lampson 1973), and are now generally, classified based on (Gligor 1993):

- **Scenarios.** In general, when building covert channels scenarios, there is a differentiation between storage and timing covert channels (Lipner 1975). Storage covert channels are where one process uses direct (or indirect) data writing, whilst another process reads the data. It generally uses a finite system resource that is shared between entities with different privileges. Covert timing channels use the modulation of certain resources, such as the CPU timing, in order to exchange information between processes.
- **Noise.** As with any other communication channel, covert channels can be noisy, and vary in their immunity to noise. Ideally, a channel immune to noise is one where the probability of the receiver receiving exactly what the sender has transmitted is unity, and there are no interferences in the transmission. Obviously, in real-life, it is very difficult to obtain these perfect channels, hence it is common to apply error correction codes, which can obviously reduce the bandwidth of the

channel.

- **Information flows.** With conventional lines of transmission, different techniques are applied to increase the bandwidth. A similar method can be achieved in the covert channels. Channels where several information flows are transmitted between sender and receiver are denominated aggregated channels, and depending on how sent variables are initialized, read and reset, aggregations can be classified as serial, parallel, and so on. Channels with a unique information flow are denominated non-aggregated.

The concern for the presence of covert channels is common in high security systems (Figure 4.7), such as military ones, where typically two observed users know that someone wishes to listen to their conversations. Many of the studies done about attacks based on covert channels and its prevention have been done by US government and military bodies, such as the National Security Agency, US Air Force, National Computer Security Centre, and so on. However, in other environments it is also possible the existence of covert channels, especially in protocols like the TCP/IP protocol suite (Route 1996; Rowland 1996).

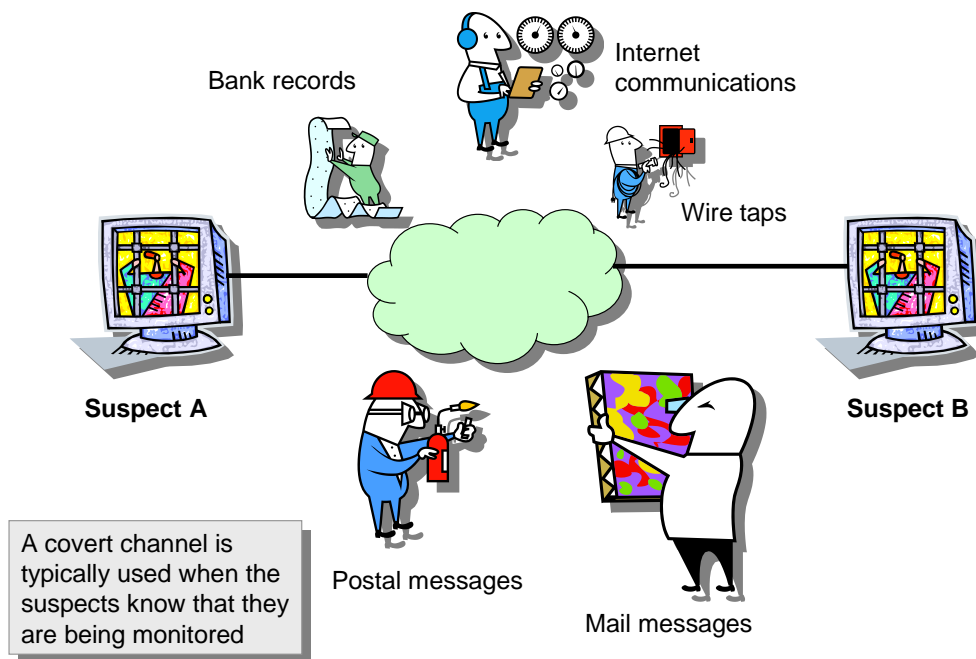


Figure 4.7 Covert channels

In covert channel scenarios *Alice* is often considered to be an inmate of a high security prison. It is assumed that she knows an escape plan from a prison where *Bob* is spending his sentence. *Alice* is trying to send the escape plan to *Bob*, however *Eve*, the governor checks their communication very precisely, thus they employ covert channel know to them to sent the secret messages (Kwecka, 2006). Figure 4.8 illustrates this.

IP and TCP data hiding

The IP and TCP protocols have many fields which are not actually necessary for most types of transmission. They could thus be a source of covert channels, as the addi-

tional fields are typically not checked by any intermediate device. In Figure 4.9 the fields which could contain a covert channel in the IP header includes: Identification; TTL and Fragment Offset. For the Identification, the original RFC (RFC 791) defines that it ensures that the IP data packets have a unique identification number within a given time window. The implementation of the actual generation of the identification numbers has thus been left to the operation system developments. An example from Ubuntu shows that it starts with a random and then takes a jump after a given number of TCP segments:

No.	Time	Source	Destination	Protocol	Info
42	23.937372	192.168.75.138	192.168.75.1	TCP	54064 >
icslap [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=18836 TSER=0 WS=5					
Identification: 0x1640 (5696)					
44	23.943145	192.168.75.138	192.168.75.1	TCP	54064 >
icslap [ACK] Seq=1 Ack=1 Win=5856 Len=0 TSV=18838 TSER=2182531					
Identification: 0x1641 (5697)					
45	23.945922	192.168.75.138	192.168.75.1	TCP	54064 >
icslap [PSH, ACK] Seq=1 Ack=1 Win=5856 Len=133 TSV=18838 TSER=2182531					
Identification: 0x1642 (5698)					
49	23.974294	192.168.75.138	192.168.75.1	TCP	54064 >
icslap [ACK] Seq=134 Ack=225 Win=6912 Len=0 TSV=18845 TSER=2182534					
Identification: 0x1643 (5699)					
50	23.974900	192.168.75.138	192.168.75.1	TCP	54064 >
icslap [ACK] Seq=134 Ack=1673 Win=9824 Len=0 TSV=18845 TSER=2182534					
Identification: 0x1644 (5700)					
51	23.975155	192.168.75.138	192.168.75.1	TCP	54064 >
icslap [ACK] Seq=134 Ack=1807 Win=12704 Len=0 TSV=18845 TSER=2182534					
Identification: 0x1645 (5701)					
53	23.977703	192.168.75.138	192.168.75.1	TCP	54064 >
icslap [FIN, ACK] Seq=134 Ack=1808 Win=12704 Len=0 TSV=18846 TSER=2182534					
Identification: 0x1646 (5702)					
55	23.979951	192.168.75.138	192.168.75.1	TCP	54065 >
icslap [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=18847 TSER=0 WS=5					
Identification: 0x0050 (80)					
57	23.981798	192.168.75.138	192.168.75.1	TCP	54065 >
icslap [ACK] Seq=1 Ack=1 Win=5856 Len=0 TSV=18847 TSER=2182535					
Identification: 0x0051 (81)					
58	23.984743	192.168.75.138	192.168.75.1	TCP	54065 >
icslap [PSH, ACK] Seq=1 Ack=1 Win=5856 Len=133 TSV=18848 TSER=2182535					
Identification: 0x0052 (82)					

View at: http://buchananweb.co.uk/packet_ip_ub.txt

And in Windows it differs as starts with a random value, and then increments each TCP data segment by one each:

No.	Time	Source	Destination	Protocol	Info
3	0.001525	192.168.75.132	192.168.75.1	TCP	afrog > http
[SYN] Seq=0 Win=64240 Len=0 MSS=1460					

No.	Time	Source	Destination	Protocol	Info
Identification: 0x008c (140)					
4	3.019628	192.168.75.132	192.168.75.1	TCP	afrog > http
[SYN] Seq=0 Win=64240 Len=0 MSS=1460					
Identification: 0x008e (142)					
7	8.968288	192.168.75.132	192.168.75.1	TCP	afrog > http
[SYN] Seq=0 Win=64240 Len=0 MSS=1460					
Identification: 0x008f (143)					
... Packets missed out ...					
129	30.598774	192.168.75.132	84.53.138.18	TCP	dcutility >
http [ACK] Seq=4751 Ack=28096 Win=63188 Len=0					
Identification: 0x00d1 (209)					

View at: http://buchananweb.co.uk/packet_ip_windows.txt

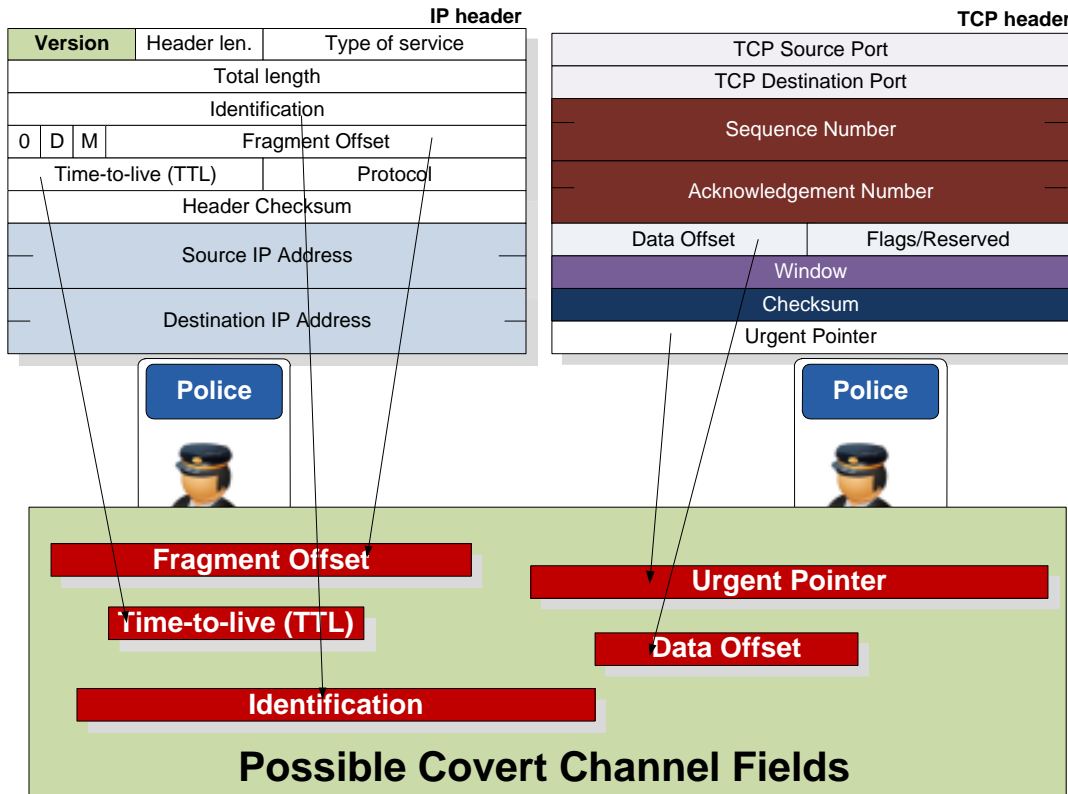


Figure 4.8 Possible fields for data hiding in IP and TCP headers

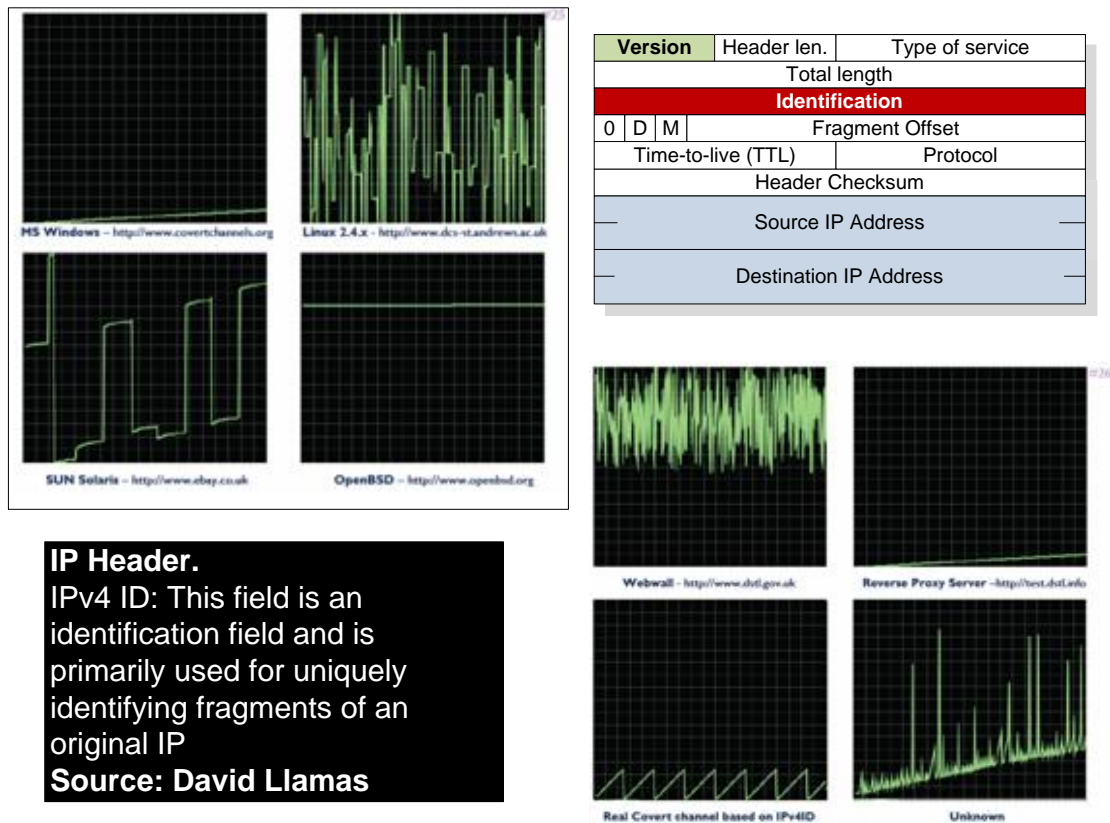


Figure 4.9 Identification field

4.6 Watermarking and Stenography

A digital watermark is either visible or invisible, and is typically a copyright mark which is added to the content. This is normally done with graphical/animation files, where an invisible element to graphics is added. For example, in Figure 4.10, the text “Bill’s Graphic” has been added, but the opacity of the text has been changed from 100% down to 50%. If it was changed to 0% it would be invisible to the user, but the text would still be there. This method, though, can normally be spotted and easily deleted. Also it only works on graphics/video formats which support opacity and vector-based graphics, such as PNGs, and so on. Unfortunately bit-mapped images such as GIF and JPEG images do not support it.

There are literally an endless number of ways that stenography can be used. One example, is to add information into files which can not actually be used, such as in images files. Figure 4.11 shows an example where a GIF file contains a colour table, of which, typically, not all the colours are used in any image. Thus text can be added to the file, which will never actually be seen.

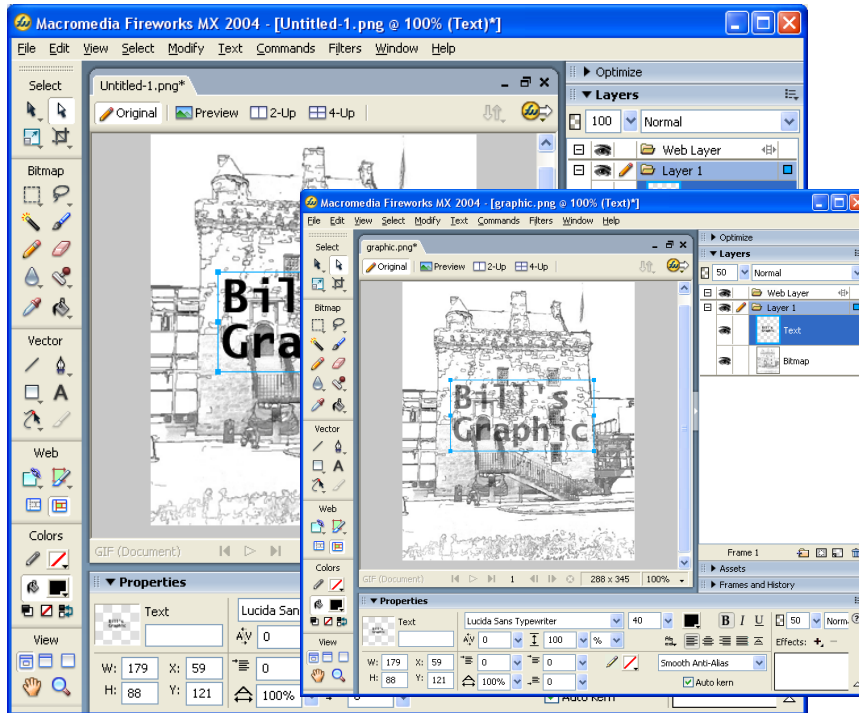


Figure 4.10 Information hiding classifications

Another way is to add information to images, which have no visual effect on the image. This is typically to high information in the high-frequency changes in images, such as in Figure 4.12.

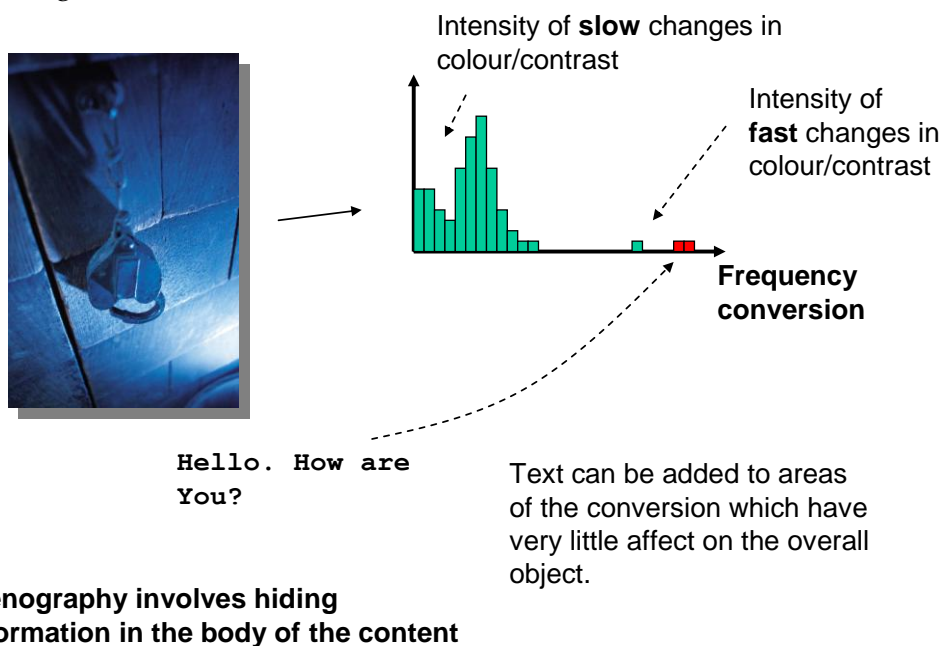


Figure 4.11 Information hiding classifications

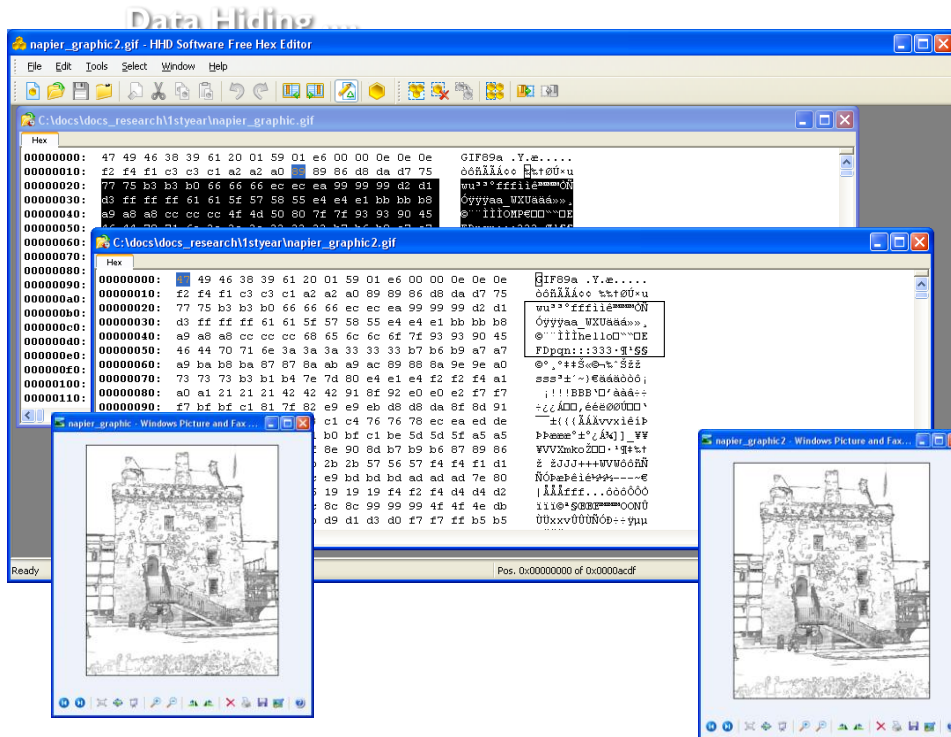


Figure 4.12 Information hiding classifications

4.7 Hiding File contents

Most computer file names are made up of a filename and a file extension, where the extension is used to define the classification of a file, such as for a word processing document, a spreadsheet, and so on. Graphics files, for example, are often used in investigations, thus it is important to identify them on a file system. One method is to search for the file extension, such as JPEG, GIF or PNG through the current folder or subfolders, such as with:

```
dir *.jpg /s
```

from the command prompt, whereas another method is to use the find utility in Windows. A search of the key file formats might include:

Microsoft Word documents	.DOC or .RTF
Image files	.GIF, .JPG, .PNG
Presentation files	.PPT
Spreadsheets	.XLS

One problem with the method of searching for files by their file extension is when the files have been obfuscated in some way, such as where the file extension of the file has been changed, or where the images have been embedded within other documents.

File contents

Files contain binary information which are typically read one byte at a time. In order to make the binary information readable the binary digits are typically interpreted in a hexadecimal format as it is relatively easy to convert from binary to hexadecimal, and vice-versa. With text files, the characters are typically stored as ASCII characters, which can be read directly in a readable format. For example a ZIP file has the following bit sequence at the start of the file:

```
0101 0000 0100 1011 0000 0011 0000 0100
```

which is difficult to remember or to define, thus the hexadecimal equivalent of:

```
50 4B 03 04h
```

is easier to define. Some of the 8-bit binary values will produce a printable character, such as the values from 20h to 7Eh. For example 20h is a space character, 21h is '!', and so on. Thus the hexadecimal value of:

```
50 4B 03 04h
```

when interpreted as ASCII characters is displayed as:

```
PK
```

where \square is a non-printing character. Thus binary files can contain some information which can be interpreted by a viewer which displays each byte as an ASCII character. Unfortunately ASCII is a rather limited character set, and does not support enhanced characters, such as for mathematical symbols. Thus other character sets can be used to save information. A good example is Unicode which extends the character sets with more bits, typically 16-bits for each character. Thus for files stored as 16-bit Unicode, the characters must be interpreted 16 bits at a time. For example, in Figure 4.13 a PowerPoint file has been created and an image of `pics_cookie_transparent_32colors.gif` has been imported into the file. It can be seen in Figure 4.14 that the original name of the file is stored as:

```
00 70 00 69 00 63 00 73 ...
```

which is interpreted as:

```
pics_cookie_transparent_32colors
```

thus the lower 8 bits of the 16-bit character still displayed as an ASCII character, but a search for this name, for example, must have to involve searching 16 bit values, at a time. The similarity between ASCII and Unicode can be seen from:

Char	ASCII (hex)	Unicode
'A'	41h	0041h

'B' 42h 0042h
 'C' 43h 0043h
 and so on.

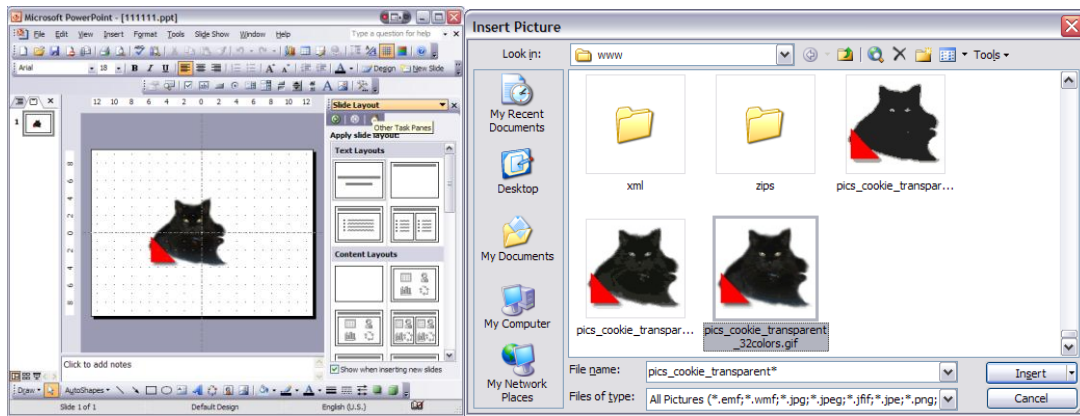


Figure 4.13 PowerPoint example

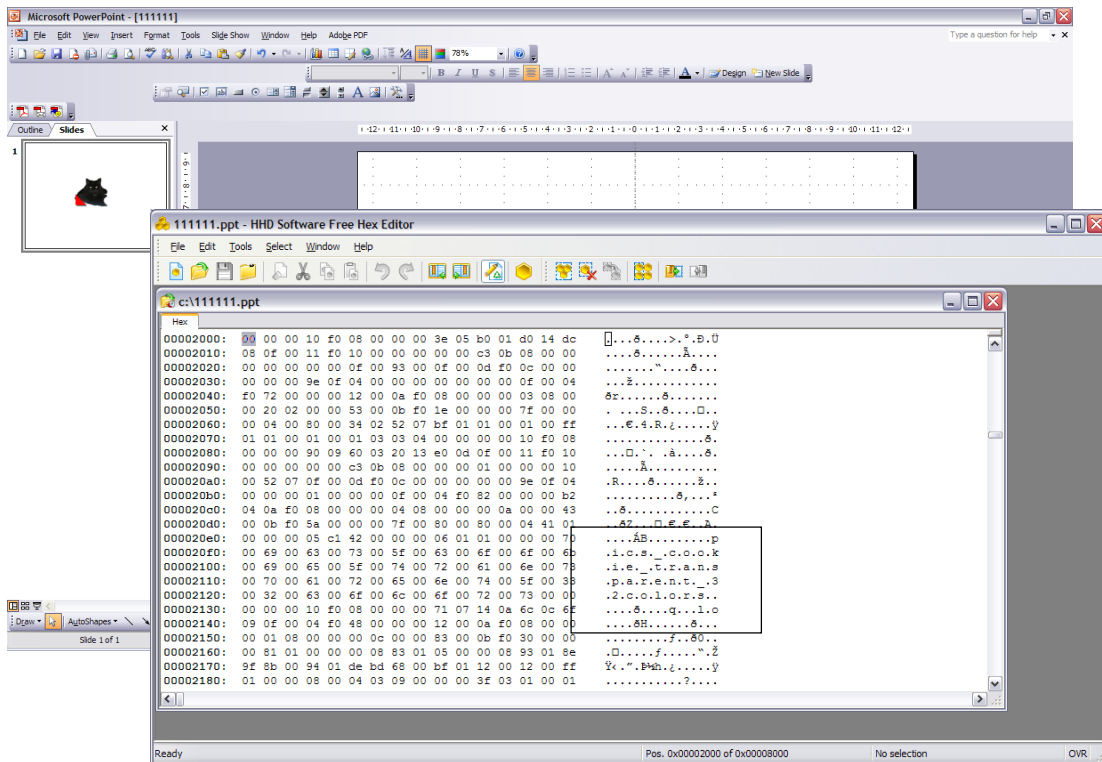


Figure 4.14 PowerPoint file format

The pattern stored is thus in the form:

```
00 63 00 6f 00 6f 00 6b 00 69 00 65 00 5f 00 74     .c.o.o.k.i.e._t
00 72 00 61 00 6e 00 73 00 70 00 61 00 72 00 65     .r.a.n.s.p.a.r.e
```

The search string can be modified so that it looks for the string of "\0c\0o\0o\0k\0i\0e" rather than for "cookie". The following code snippet achieves this:

```

using System;
using System.IO;

namespace ConsoleApplication1
{
    class Class1
    {
        static void Main(string[] args)
        {
            DirectoryInfo di = new DirectoryInfo("c:\\test123");
            FileInfo[] rgFiles = di.GetFiles("*.");
            foreach(FileInfo fi in rgFiles)
            {
                StreamReader f = new StreamReader("c:\\test123\\"+fi.Name);
                string s = f.ReadToEnd();
                string search = "\0c\0o\0o\0k\0i\0e";
                if (s.LastIndexOf(search)>0)
                {
                    Console.WriteLine("Search signature found, name: " + fi.Name);
                }
            }
            Console.WriteLine("Press return to end..");
            Console.ReadLine();
        }
    }
}

```

Another method is to create a byte array with the byte sequence to search for, and then convert it to a string, such as with:

```

byte [] b = {0,(byte)'c',0,(byte)'o',0,(byte)'o',0,
             (byte)'k',0,(byte)'i',0,(byte)'e'};
System.Text.ASCIIEncoding enc = new System.Text.ASCIIEncoding();
string search = enc.GetString(b);

```

The standard Windows search does not cope well with binary searches, but the standard find utility copes better, such as:

```

C:\test123>find /?
Searches for a text string in a file or files.

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]] "string" [[drive:][path]filename[ ...]]

/V          Displays all lines NOT containing the specified string.
/C          Displays only the count of lines containing the string.
/N          Displays line numbers with the displayed lines.
/I          Ignores the case of characters when searching for the string.
/OFF[LINE] Do not skip files with offline attribute set.
"string"    Specifies the text string to find.
[drive:][path]filename
             Specifies a file or files to search.

If a path is not specified, FIND searches the text typed at the prompt
or piped from another command.

C:\test123>find "GIF89a" *.*
----- 111111.PPT
----- 123.JPG
----- AA.GIF
GIF89aã♥3●µ
----- AGENTS02.GIF
GIF89aπ
----- AGENT_GRAPHIC01.GIF
GIF89a█
----- AGENT_GRAPHIC02.GIF
GIF89a(●b

```



```
----- FLASH_NETWORK_EMULATORS2.JPG
----- PRES01.PPT
----- SRCCODE.ZIP
```

4.7.1 GIF files

The graphics interchange format (GIF) is the copyright of CompuServe Incorporated. Its popularity has increased mainly because of its wide usage on the Internet. Most graphics software support the Version 87a or 89a format (the 89a format is an update the 87a format). Both the basic specification of:

- A header with GIF identification.
- A logical screen descriptor block which defines the size, aspect ratio and color depth of the image place.
- A global color table. Color tables store the color information of part of an image (a local color table) or they can be global (a global table).
- Data blocks with bitmapped images and the possibility of text overlay.
- Multiple images, with image sequencing or interlacing. This process is defined in a graphic-rendering block.
- Compressed bitmapped images.

Blocks can be specified into three groups: control, graphic-rendering and special purpose. Control blocks contain information used to control the process of the data stream or information used in setting hardware parameters. They include:

- GIF Header – which contains basic information on the GIF file, such as the version number and the GIF file signature.
- Logical screen descriptor – which contains information about the active screen display, such as screen width and height, and the aspect ratio.
- Global color table – which contains up to 256 colors from a palette of 16.7M colors (i.e. 256 colors with 24-bit color information).
- Data subblocks – which contain the compressed image data.
- Image description – which contains, possibly, a local color table and defines the image width and height, and its top left coordinate.
- Local color table – an optional block which contains local color information for an image as with the global color table, it has a maximum of 256 colors from a palette of 16.7M.
- Table-based image data – which contains compressed image data.
- Graphic control extension – an optional block which has extra graphic-rendering information, such as timing information and transparency.
- Comment extension – an optional block which contains comments ignored by the decoder.
- Plain text extension – an optional block which contains textual data.
- Application extension – which contains application-specific data. This block can be used by a software package to add extra information to the file.
- Trailer – which defines the end of a block of data.

The key to identifying the GIF file is the six bytes long initial header which identifies

the GIF signature and the version number of the chosen GIF specification. Its format is (Figure 4.15):

- 3 bytes with the characters 'G', 'I' and 'F'.
- 3 bytes with the version number (such as 87a or 89a). Version numbers are ordered with two digits for the year, followed by a letter ('a', 'b', and so on).

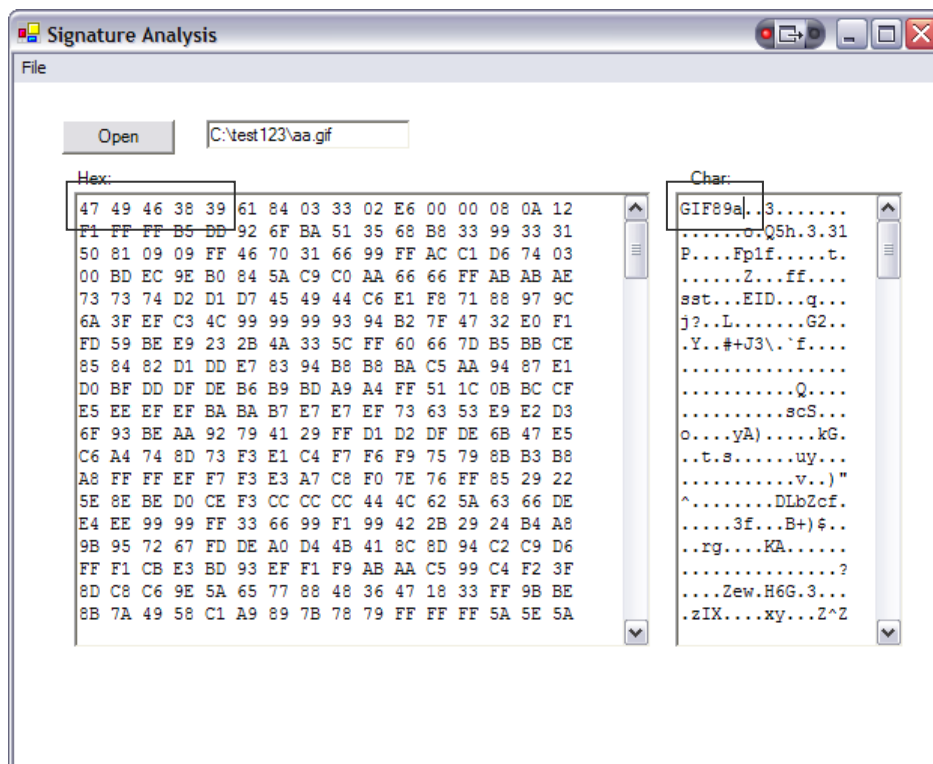


Figure 4.15: GIF file header

4.7.2 JPEG file format

JPEG is a standard compression technique. The files which contain JPEG images normally complies with JFIF (JPEG file interchange format) which is a defined standard file format for storing a gray scale or color image. The data within the JFIF contains segments separated by a 2-byte marker. This marker has a binary value of 1111 1111 (FFh) followed by a defined marker field. If a 1111 1111 (FFh) bit field occurs anywhere within the file (and it isn't a marker), the stuffed 0000 0000 (00h) byte is inserted after it so that it cannot be read as a false marker. The uncompression program must then discard the stuffed 00h byte.

Some of the key markers are:

- Start of image (FFD8h). The segments can be organized in any order but the start-of-image marker is normally the first 2 bytes of the file. Refer to Figure 4.16 and 4.17 for the file format.
- Application-specific type 0 (FFE0h). The JFIF header is placed after this marker.

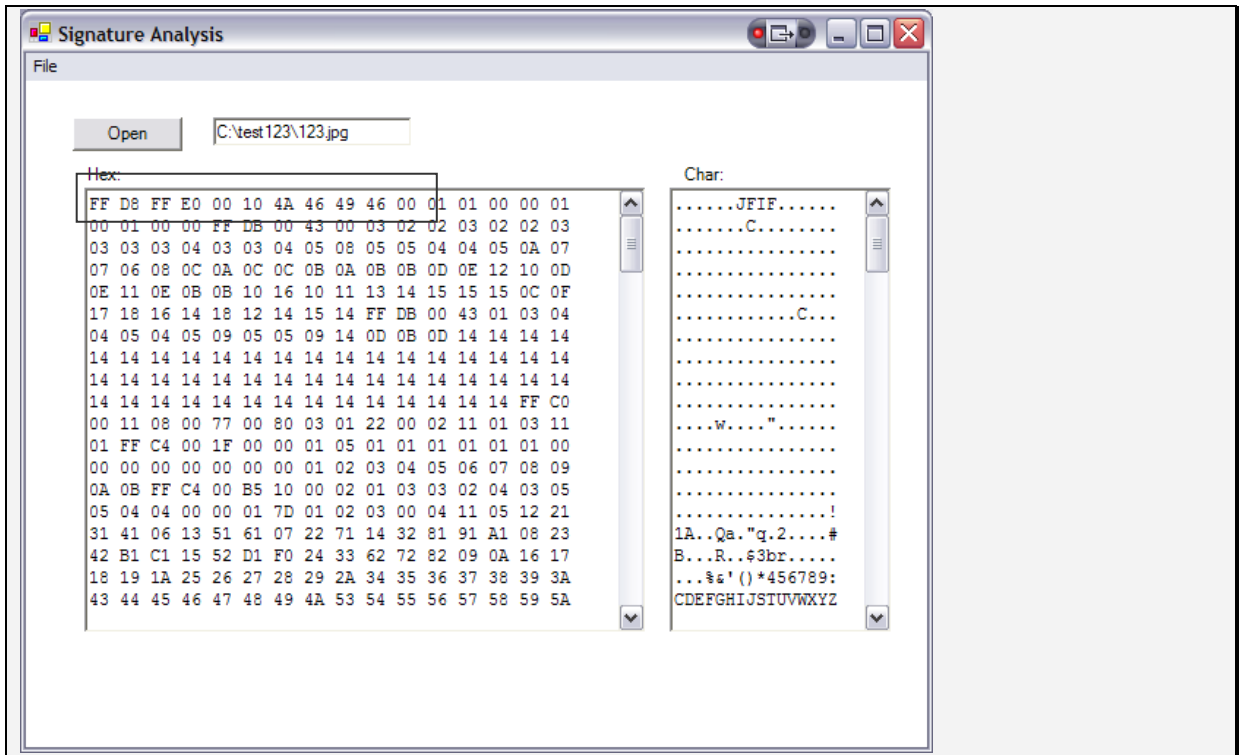


Figure 4.16 JPEG file reading

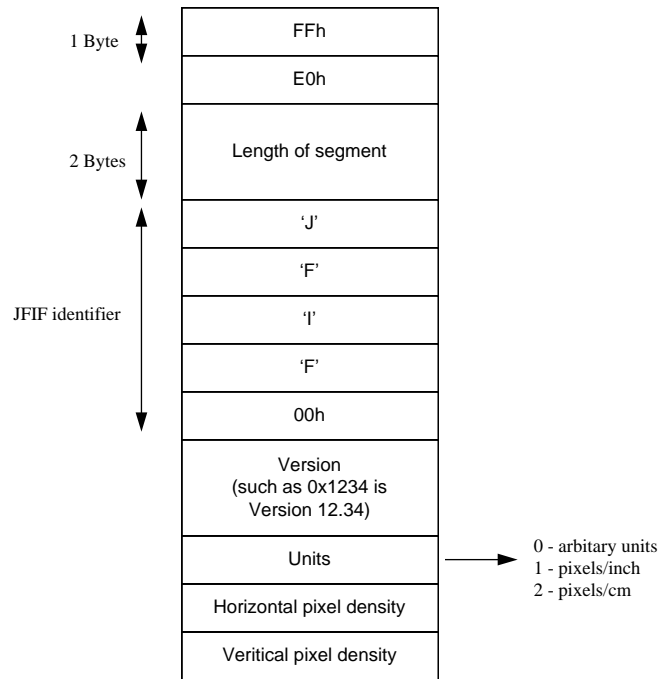
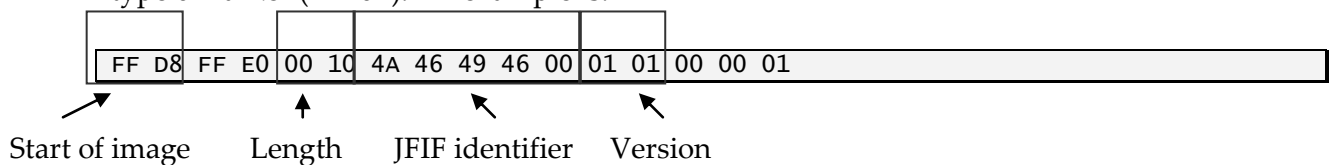


Figure 4.17 JFIF header information

JPEG graphics files have a JFIF header which begins with the application-specific type 0 marker (FFE0h). An example is:



The first 11 bytes can thus be used to identify the start of a JPEG image file, where the hex value of a 'J' is 4Ah, an 'F' is 46h, an 'I' is 49h. Thus the string "JFIF" is represented with the hexadecimal pattern of 4A464946h (Figure 4.18).

4.7.3 ZIP file format

From a forensics point-of-view the detection of a ZIP file can often mean that a file(s) has/have been compressed into a single file. The basic format of its header is:

Byte pos.	Name	No. of bytes	Contents	Description
00	ZIPLOCSIG	4	50 4B 03 04h	File signature
04	ZIPVER	2		Version required for extraction
06	ZIPGENFLG	2		General purpose bit flag
08	ZIPMTHD	2		Compression method
0A	ZIPTIME	2		Time last modified
0C	ZIPDATE	2		Date last modified
0E	ZIPCRC	4		CRC-32
12	ZIPSIZE	4		Compressed size
16	ZIPUNCOMP	4		Uncompressed size
1A	ZIPFNLEN	2		Filename length
1C	ZIPXTRLEN	2		Extra field length
1E	ZIPNAME			Filename

The ZIP file format contains quite an amount of data about the contents of the ZIP file. Apart from the signature, it can be seen from Figure 4.19 that the file names of the files contained within the ZIP file are also contained in the header.

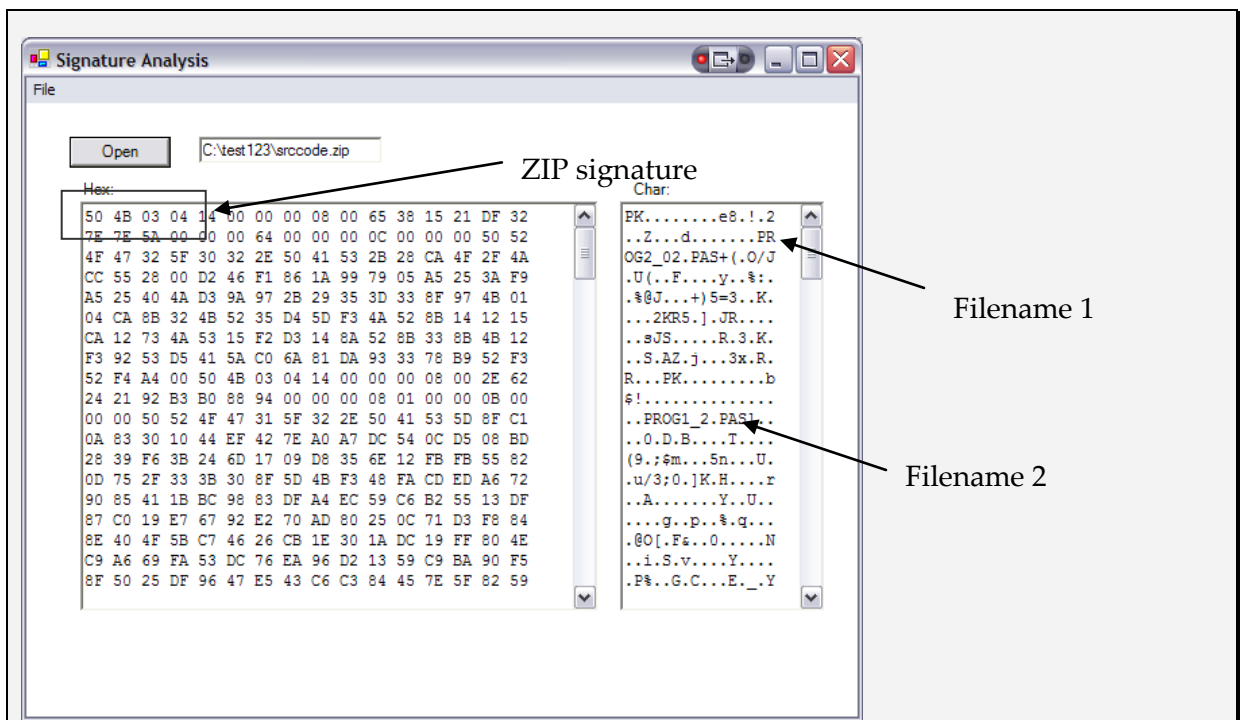


Figure 4.18 ZIP file information

4.8 References

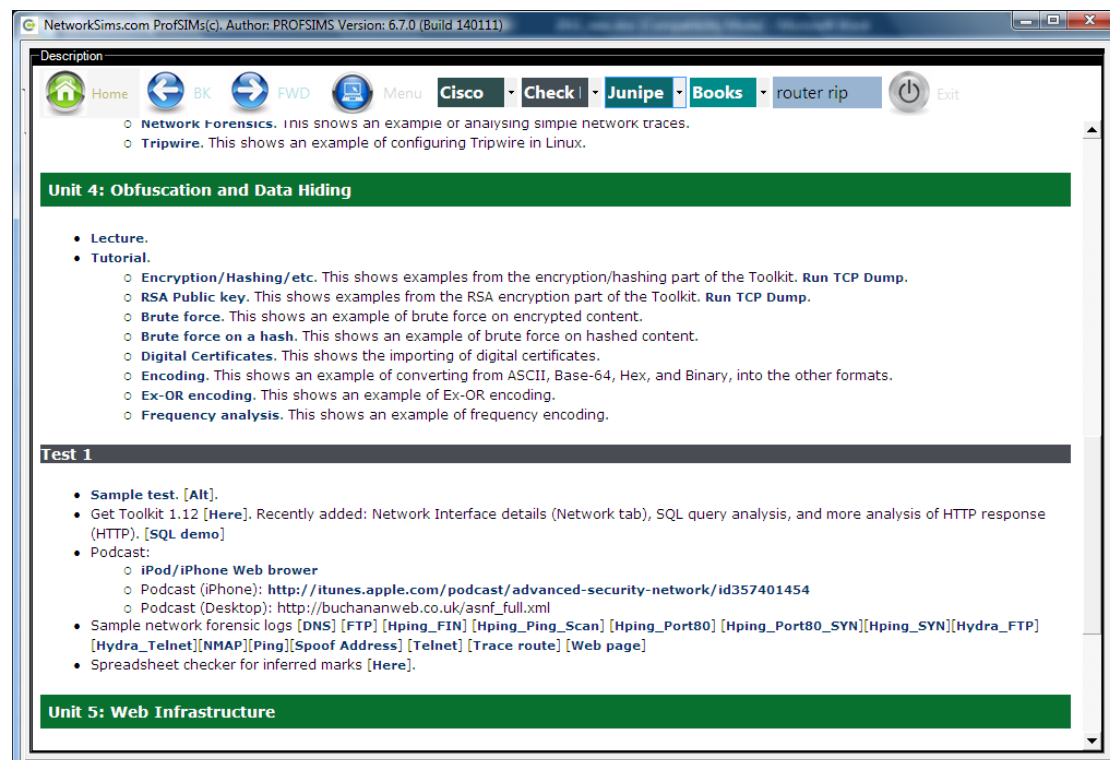
- Berg, S. (1998). Glossary of Computer Security Terms, <http://packetstormsecurity.org/docs/rainbow-books/NCSC-TG-004.txt>.
- Gligor, V. D. (1993). A Guide to understanding Covert Channel Analysis of Trusted Systems. Technical Report NCSC-TG-030, National Computer Security Centre.
- Kwecka Z, (2006), Hons Project Dissertation, Award Winner, Young Software Engineer of the Year Award. <http://www.dcs.napier.ac.uk/~bill/zk.pdf>.
- Lampson, W. (1973). "A note on the Confinement Problem. Communications of the ACM." (16(10)): 613-615.
- Lipner, S. B. (1975). "A note on the Confinement Problem." Operating Systems Review, 9(5): 192-196.
- Llamas D, (2004), Hons Project Dissertation, Award Winner, Young Software Engineer of the Year Award. http://www.dcs.napier.ac.uk/~bill/PROJECTS/2004/ david_llamas.pdf.
- Proctor, N. E. and P. G. Neumann (1992). Architectural implications of Covert Channels. 15th National Computer Security Conference, 28-43.
- Route (1996). "Project Loki: ICMP Tunnelling." Phrack Magazine 7(49).
- Rowland, C. H. (1996). Covert Channels in the TCP/IP Protocol Suite, http://www.firstmonday.dk/issues/issue2_5/rowland/.

4.9 Tutorial

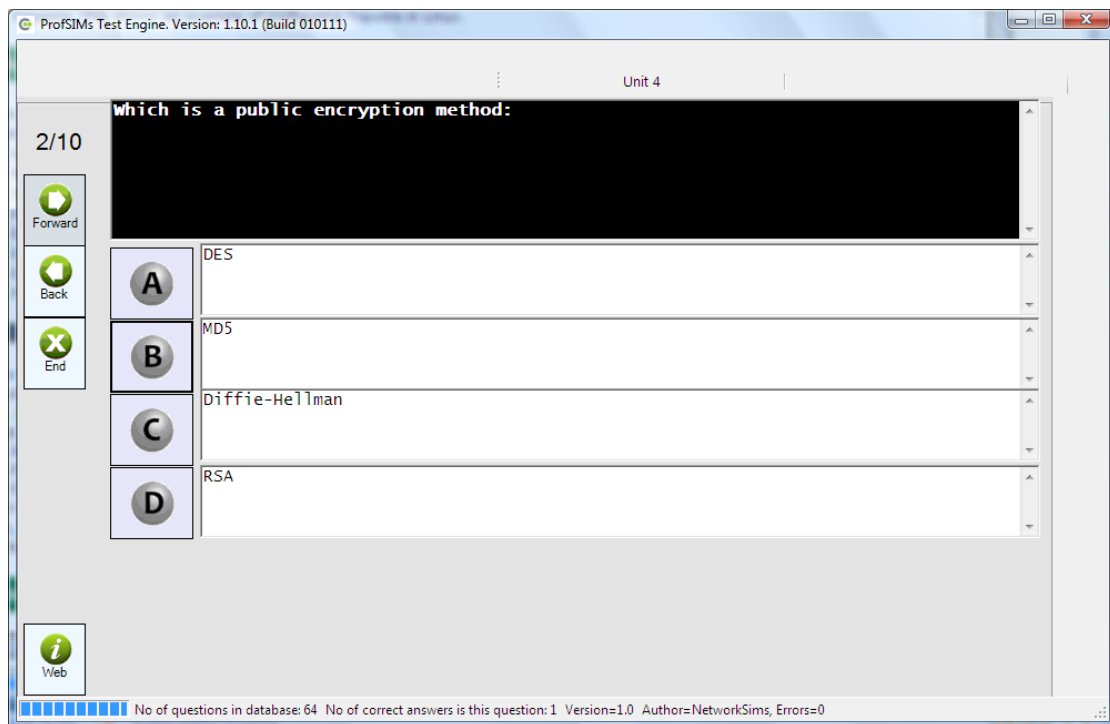
The main tutorial is at:

On-line tutorial: <http://buchananweb.co.uk/adv04.html>

The main tutorial is in NetworkSims ProfSIMs at:



And then select the Tutorial to give:



5 Web Infrastructures



On-line lecture: <http://buchananweb.co.uk/adv/unit05.html>

5.1 Objectives

The key objectives of this unit are to:

- Provide an overview of Web-based architectures, especially in authentication and access control.
- Define key protocols involved in next generation Web-based infrastructures, such as Kerberos and SOAP over HTTP.
- Define scalable authentication infrastructures and protocols.
- Investigate scaleable and extensible architectures, including using LDAP.

5.2 Introduction

The Internet has been built around a wide range of services, such as Web (HTTP), Remote Access (Telnet), File Transfer (FTP), Email (SMTP), and so on, where each of these protocols has used a specific TCP port to identify themselves. This produces complex infrastructures where each service must provide its own authentication and authorization. In the future systems are likely to be built around a Web infrastructure where a common authentication and authorization infrastructure is used to provide access to a wide range of services, each of which can integrate over a wide area.

5.3 Identity 2.0

The Internet was created to be an infrastructure of computers, each with a unique IP address. This scope of the Internet is now increasing where it can support the integration of users, each with their own unique identity. Unfortunately systems have been built where users must log onto each system with a unique identity instance. This makes it difficult for users to manage their own environment, and thus user-centric technologies techniques are being proposed which will allow users to manage their own identity and then to use Information Cards (such as Microsoft CardSpace) or OpenID, to verify their identity. There are many advantages of users controlling their own digital identity in that:

- They can choose a safe repository for it which focuses on keeping this identity secure.
- They can share only the parts of the identity which are relevant to the access.
- They can provide their identity on one occasion, and then automatically sign on using a digital identity card.
- They only have to remember one login and password.

Figure 5.1 shows an example of how a user could control their identity. In this case the user may show their home telephone number and their NI number to a medical practitioner, while their CV and email address would be exposed to their employer (or future employers, of course).

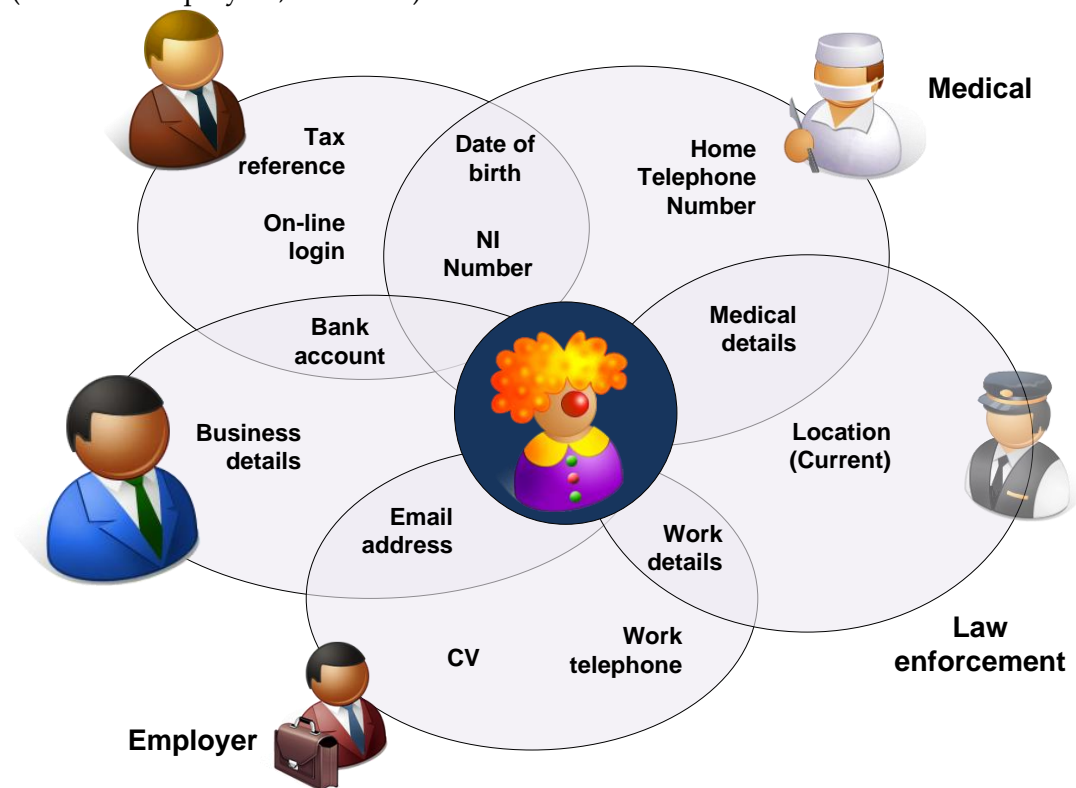


Figure 5.1 Identity 2.0

5.4 SOAP over HTTP

SOAP (Simple Object Access Protocol) is a method of exchanging messages in a Web Service infrastructure. It uses XML, and typically uses Remote Procedure Call (RPC) or HTTP for message negotiation and transmission. It is thus used to send messages and objects over infrastructures built on different types of systems. SOAP over HTTP allows for messages to be transferred through HTTP, which will typically pass over a firewall.

SOAP uses XML to create a message, which is contained within an envelope, along within an optional Header element. It then contains a Body element, and an optional Fault element (which contains the reason why an error occurred in the processing of a SOAP message). An example is:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <Message xmlns="http://www.software.org" />
  </Body>
</Envelope>
```

The first line is the xmlsoap namespace, which identifies the envelope as a SOAP Envelope. It is also possible to make namespaces explicit. Most SOAP messages do not use the default namespace, but using an explicit one. An example of this is:


```
<soap:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Message xmlns:m="http://www.soapware.org/"/>
  </soap:Body>
</soap:Envelope>
```

Normally there are arguments added to an element (which is <message> in this case):

```
<soap:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <message xmlns:m="http://www.soapware.org/">
      <title>hello</title>
      <content>This is the message</content>
    </message>
  </soap:Body>
</soap:Envelope>
```

In SOAP, elements which are not supported are ignored, and the server will continue to process the other elements. Along with this the envelope element can contain other information, such as the encoding method:

```
<soap:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <message xmlns:m="http://www.soapware.org/">
      <title>hello</title>
      <content>This is the message</content>
    </message>
  </soap:Body>
</soap:Envelope>
```

The format of data can also be defined, such as:

```
<soap:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <message xmlns:m="http://www.soapware.org/">
      <title>hello</title>
      <content>This is the message</content>
      <msgid xsi:type="xsd:int">1234</msgid>
    </message>
  </soap:Body>
</soap:Envelope>
```

which defines that msgid is a 32-bit integer. An example of a SOAP request is:

```
<soap:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CalcRootResponse xmlns="http://MyMath.com/math">
      <CalcRootResult>9</CalcRootResult>
    </CalcRootResponse>
  </soap:Body>
</soap:Envelope>
```

and the response could be:

```
<double xmlns="http://MyMath.com/math">3</double>
```

5.5 LDAP

LDAP (Lightweight Directory Access Protocol) is an application protocol which is used with TCP/IP to query/modify directory services. It uses the form of a directory which is a set of objects with attributes, each of which are organised in a logical and hierarchical manner. This hierarchy is based on X.500, and is based on c (country), st (state) dc (Domain Component), o (organisation), ou (Organisational Unit), l (location) and cn (Common Name) and uid (User ID), where dn is a distinguishing name (as is the name of the entity). In the example in Figure 5.2, distinguishing name is made up of domain components (napier, ac,uk), an organisation unit (Comp) and a common name (Bill).

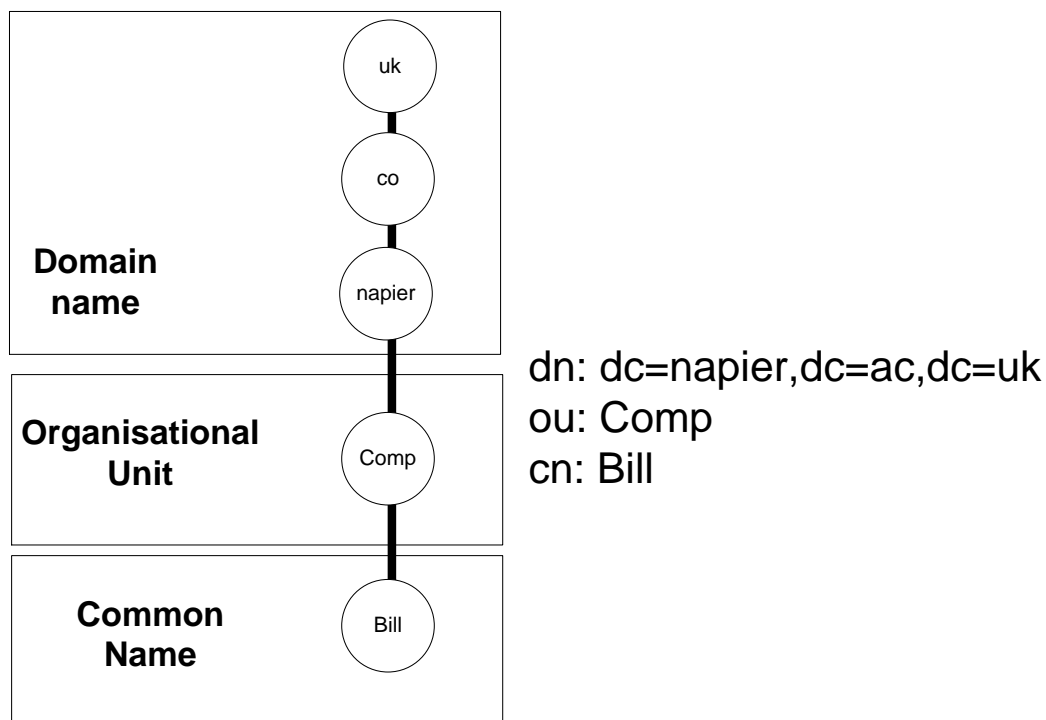


Figure 5.2 Example of LDAP

An LDAP URL can then be used to refer to objects, such as for:

```
ldap://ldap.example.com/cn=Bill,dc=napier,dc=ac,dc=uk
```

creates a reference to all the user attributes for Bill within napier.ac.uk.

G3.1.1 X.500

X.500 directory services allows resources to be mapped into a logical and hierarchal structure, which is not dependent on the actual domain or server which they connect to. In this way the directory is global to the infrastructure, thus a user can log into all authorized network-attached resources, rather than requiring to log into each separate server. It uses resources by objects, properties, and values, with:

- **Leaf objects** – which are network resources such as disk volumes, printers, printer queues, and so on.

- **Container objects** – which are cascadable organization units that contain leaf objects. A typical organizational unit might be a company, department or group.

The top of the tree is the root object, to which there is only a single root for an entire global database. Servers then use container objects to connect to branches coming off the root object. This structure is similar to the organization of a directory file structure and can be used to represent the hierarchical structure of an organization. Figure 5.3 illustrates an example with root, container and leaf objects. In this case, the organization splits into four main containers: Electrical, Mechanical, Production and Administration. Each of these containers has associated leaf objects, such as disk volumes, printer queues, and so on.

To improve fault tolerance, the branches of the tree (or partitions) are often stored on multiple file servers. These mirrors are then synchronized to keep them up to date. Another advantage of replicating partitions is that local copies of files can be stored so that network traffic is reduced.

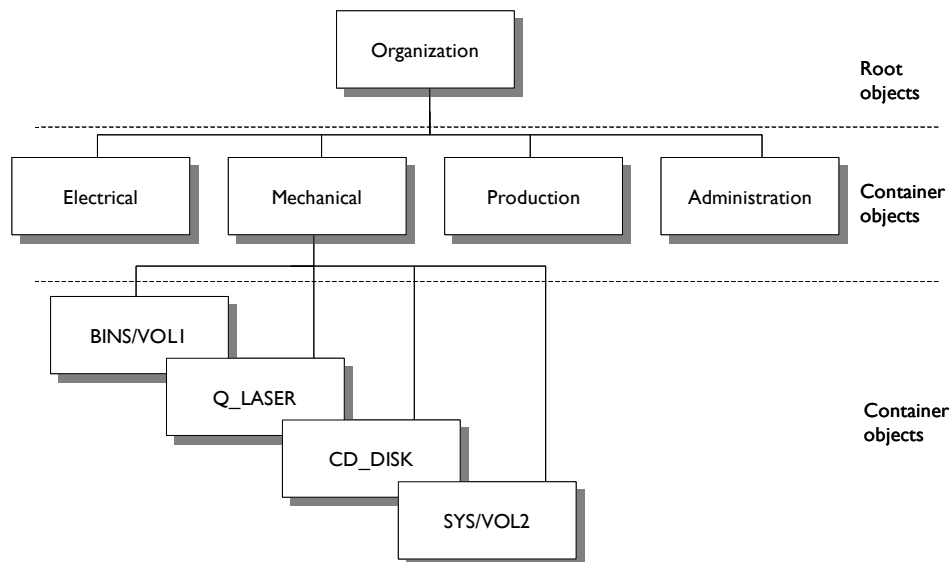


Figure 5.3 Example structure

The container objects are:



[ROOT]. This is the top level of the inverted tree and contains all the objects within the organizational structure.



Organization. This object class defines the organizational name (such as FRED_AND_CO). It is normally the next level after [ROOT] (or below the C=Country object).



User. This object defines an individual user.



Volume. This identifies the mounted volume for file services. A network file system data links to the Directory tree through Volume objects.

The most commonly used objects are:



Organizational unit. This object represents the OU part of the NDS tree. These divide the NDS tree into subdivisions, which can represent different geographical sites, different divisions or workgroups. Different divisions might be PRODUCTION, ACCOUNT, RESEARCH, and so on. Each Organizational Unit has its own login script.



Organization role. This object represents a defined role within an organization object. It is thus easy to identify users who have an administrative role within the organization.



Group. This object represents a grouping of users. All users within a group inherit the same access rights.

Figure 5.4 shows the top levels of the NDS tree. These are:

- **[ROOT].** This is the top level of the tree. The top of the NDS tree is the [ROOT] object.
- **C=Country.** This object can be used, or not, to represent different countries, typically where an organization is distributed over two or more countries. If it is used then it must be placed below the [ROOT] object. Most LDAP applications do not normally use the Country object and uses the Organization Unit to define the geographically located sites, such as SALES_UK.[ROOT], SALES_USA.[ROOT], and so on.
- **L=Locality.** This object defines locations within other objects, and identifies network portions. The Country and Locality objects are included in the X.500 specification, but they are not normally used, because many applications ignore this. When used, it must be placed below the [Root] object, Country object, Organization object, or Organizational Unit object.
- **O=Organization.** This object represents the name of the organization, a company division or a department. Each NDS Directory tree has at least one Organization object, and it must be placed below the [Root] object (unless the tree uses the Country or Locality object).
- **OU=Organization Unit.** This object normally represents the name of the organizational unit within the organization, such as Production, Accounts, and so on. At this level, User objects can be added and a system level login script is created. It is normally placed below the Organizational object.

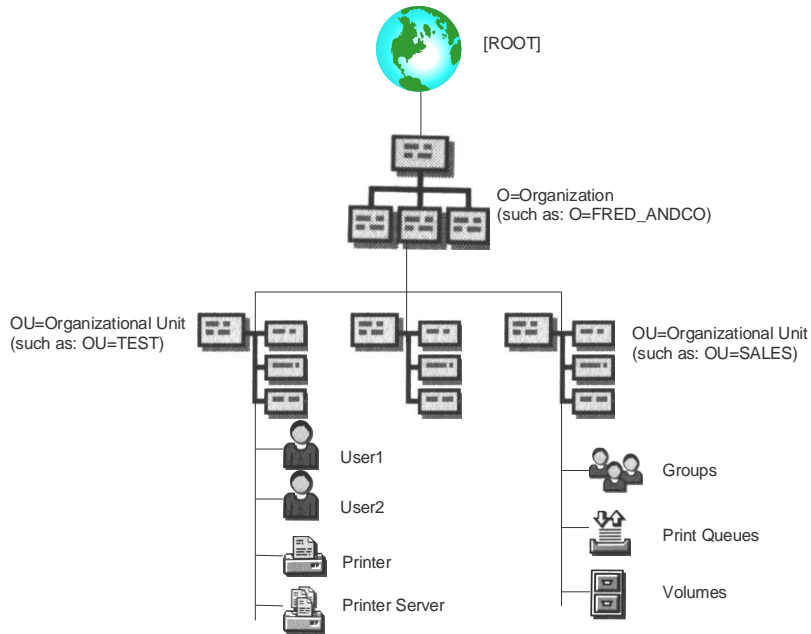


Figure 5.4 LDAP Example

A few examples are:

Access to Fred's folder	cn=Fred Folder,ou=people,dc=fake,dc=com
Identifier for Fred login	uid=fred,ou=people,dc=fake,dc=com
Identifier for Fred	cn=fred,ou=people,dc=fake,dc=com

The LDAP record stores information within the object using attribute pairs, such as (case of the letters is stored, but are not used for searches):

```

dn: ou=people,dc=fake,dc=com
   objectClass: organizationalUnit
   ou: people

dn: ou=groups,dc=fake,dc=com
   objectClass: organizationalUnit
   ou: groups

dn: uid=fred, ou= people, dc=fake, dc=com
   objectClass: inetOrgPerson
   objectClass: posixAccount
   objectClass: shadowAccount
   uid: fred
   givenname: Fred
   sn: Fredaldo
   cn: Freddy Fredaldo
   telephonenumber: 45511332
   roomnumber: C.63
   o: Fake Inc
   mailRoutingAddress: f.smith@fake.com
   mailhost: smtp.fake.com
   userpassword: {crypt}ggHi99x
   uidnumber: 5555
   gidnumber: 4321
   homedirectory: /user/fred
   loginshell: /usr/local/bin/bash

dn: cn=example,ou=groups, dc=fake,dc=com
   objectClass: posixGroup
   cn: example
   gidNumber: 10000

```

5.6 Authentication Infrastructures

Authentication can normally be done using a local device, such as a switch or access point, but this method does not scale well for larger-scale infrastructures. It has the advantage, though that a failure in parts on the infrastructure will still allow users and devices to authenticate locally. In most systems, though, authentication is centralised, in order to synchronize user names, identity provision, and so on. As this is a key service, there are normally backup and failover devices, as a lack of authentication from the central resource may lead to a complete failure of the infrastructure.

For a centralised model, as illustrated in Figure 5.5, normally a device or person (known as a supplicant) asks an access device (such as a switch or a wireless access point) to connect to the system, which will then forward the request to the central authentication server, which will then respond back to the access device with the required credentials, such as for a username/password, a digital certificate, a MAC address, or any other type of identification method (such as for a fingerprint, iris scan, and so on). The user/device then responds with its credentials, which are checked against an identity provider (such as a PKI server) and/or to a domain server (such as for a Windows or a Samba domain one). Typical methods to verify user credentials include SQL, Kerberos, LDAP, and Active Directory servers. A particular problem is then how to then map the identity and authentication to the actual access rights to the system, and thus to other external trusted systems.

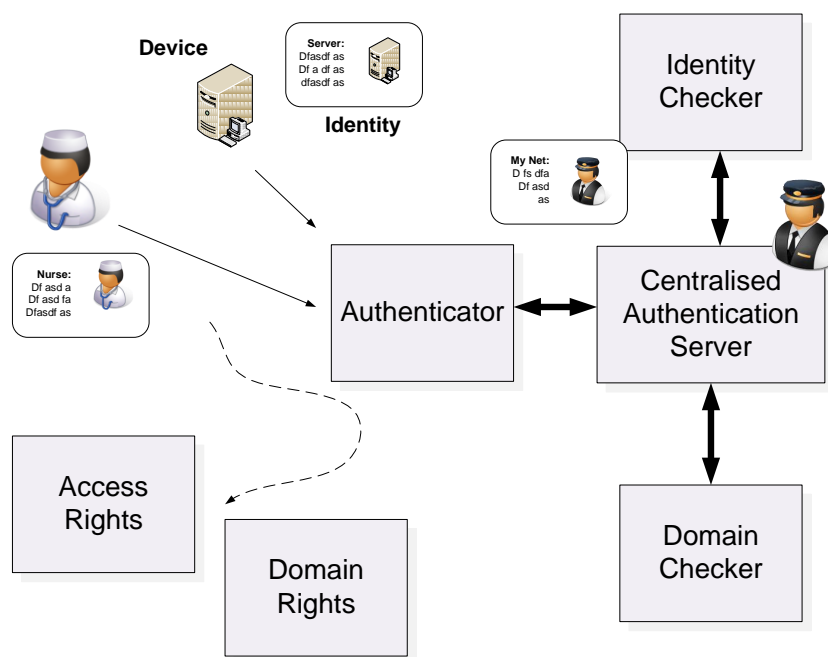


Figure 5.5 Generalised authentication infrastructure

5.7 802.1x Authentication Infrastructure

The 802.1x standard supports the authentication of users and devices onto the network at the point of their connection. With this a **supplicant** connects to an **authenticator**, such as a switch or a wireless access point. It then is setup to send the request for authentication to an **authentication server** such as a RADIUS or Tacacs+ server (Figure 5.6). If the user/device is authenticated it sends an acceptance message

back to the authenticator, which then allows the user/device onto the network. The authentication server is kept synchronised with the correct authentication details, such as synchronising with a Windows domain server for usernames and passwords, or with a PKI server for digital certificates. The 802.1x standard has many advantages including that it connects to many different types of networks including 802.11 (wireless), 802.3 (Ethernet) and PPP (Serial), and support a wide range of authentication methods, such as LEAP (username and password), PEAP (username/password or digital certificate), and so on. A great advantage is that users and devices are not allowed onto the network unless they have the required credentials, even though they have a physical or wireless connection. In the future more network infrastructures will embed 802.1x so that no user or device connects, unless they have the required authentication. For smaller networks, the authenticator server could be built into the authenticator by using a local authentication server. This is defined as local authentication.

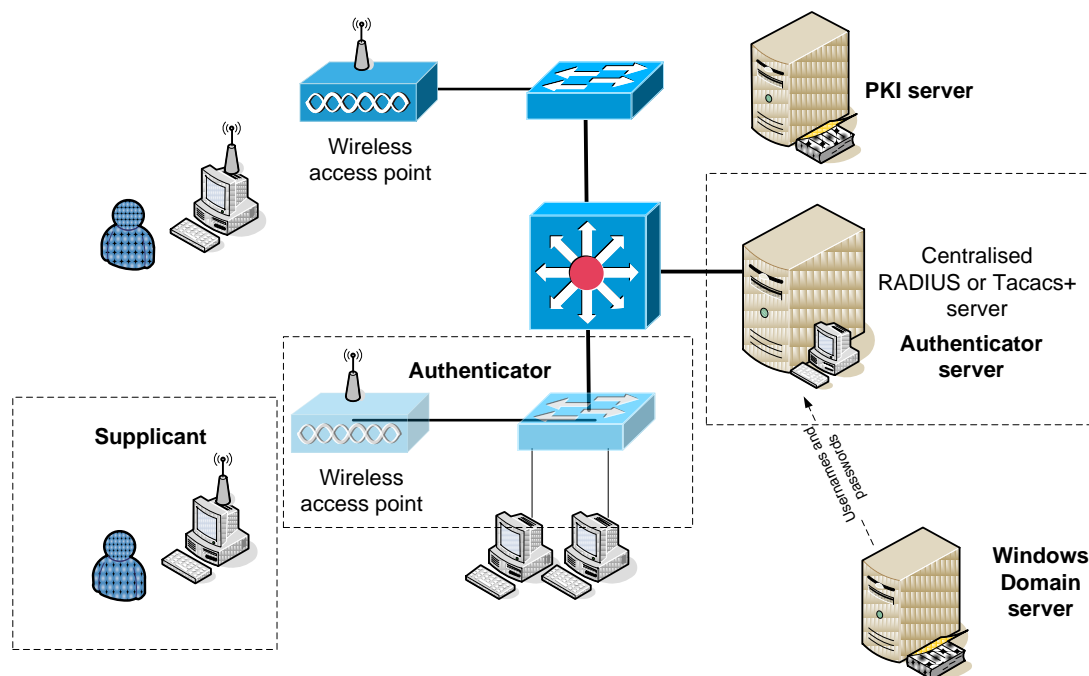


Figure 5.6 802.1x authentication infrastructure

Authentication techniques

It has been seen that standard 802.11 authentication methods can be easily overcome. There are several standard authentication methods, some of which have been developed by vendors, such as Cisco Systems, while others are international standards. Basically authentication consists of an authentication framework, an authentication algorithm and an encryption technique. The proposed enhanced authentication method tries to split these up with:

- **801.1x² authentication.** This defines the authentication framework which can support many authentication types. Ethernet network have developed so that it is

² Note 802.1x – Port-based authentication, and is not to be confused with 802.1q with VLAN tagging and is used to provide a trunk between switches, or with 802.11x

now the standard method of connecting to a wired network. The IEEE 802.1x standard aims to extend Ethernet onto wireless networks and dialup connections. It uses a port authentication method that could be used on a range of networks, including 802.3 (Ethernet), 802.11 (wireless) and PPP (serial connections). IEEE 802.1x thus defines authentication and key management, while 802.11i defines extended security. At the present the WiFi Alliance (WFA) has published the 802.11 security specification, which is known as Wi-fi Protected Access (WPA).

- **EAP** (Extensible Authentication Protocol). This defines the actual implementation of the authentication method. It thus provides centralized authentication and dynamic key distribution. It has been developed by the IEEE 802.11i Task Group as an end-to-end framework and uses 802.1x with:
 - **Authentication**. This is of both the client and the authentication server (such as a RADIUS server).
 - **Encryption keys**. These are dynamically created after authentication. They are not common to the whole network.
 - **Centralized policy control**. A session time-out generates a reauthentication and the generation of new encryption keys.
- **Encryption**. This replaces WEP with TKIP (Temporal Key Integrity Protocol), which is based on WEP but which overcomes its major weaknesses.

Figure 5.7 shows that the 802.1x framework provides an interface between many different network types and a number of differing authentication methods (such as LEAP, EAP-TLS, and so on). It can be seen that 802.1x gets in-between the Layer 3 protocol and the link layer, which means that the device cannot directly communicate with the network unless it has been authenticated. The framework supports a wide range of authentication methods, and also network technologies, and is seen as a single standard for the future of authenticated systems. As previously mentioned, 802.1x uses three main entities:

- **Supplicant**. This operates on the station client.
- **Authenticator**. This operates on the access point.
- **Authenticator server**. This operates on a RADIUS server.

Figure 5.8 shows the basic message flow for 802.1x authentication, where the supplicant sends its identity to the access point, which is then forwarded to a RADIUS server. The RADIUS server then authenticates the client, and vice-versa. If these are successful the RADIUS server sends a RADIUS-ACCEPT message to the access point, which then allows the client to join the network.

which is any existing or developing standard in the 802.11 family.

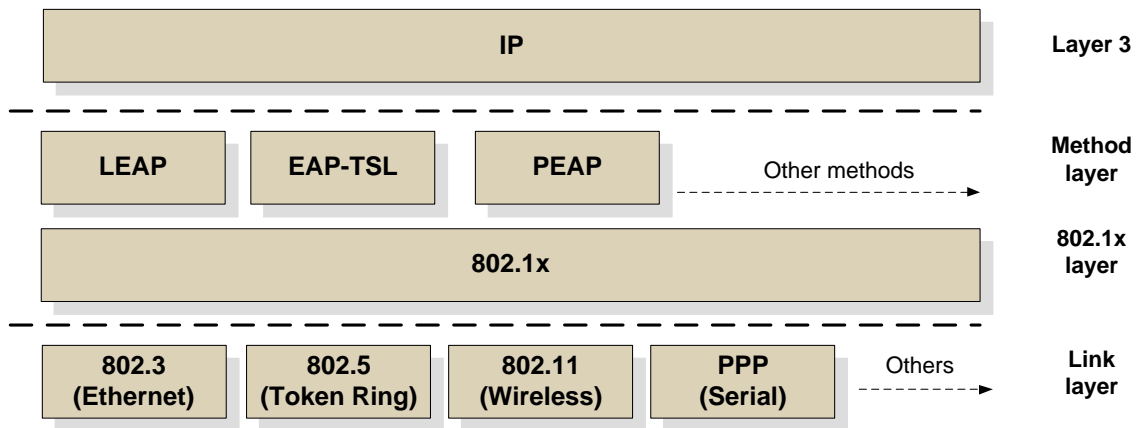


Figure 5.7 802.1x layers

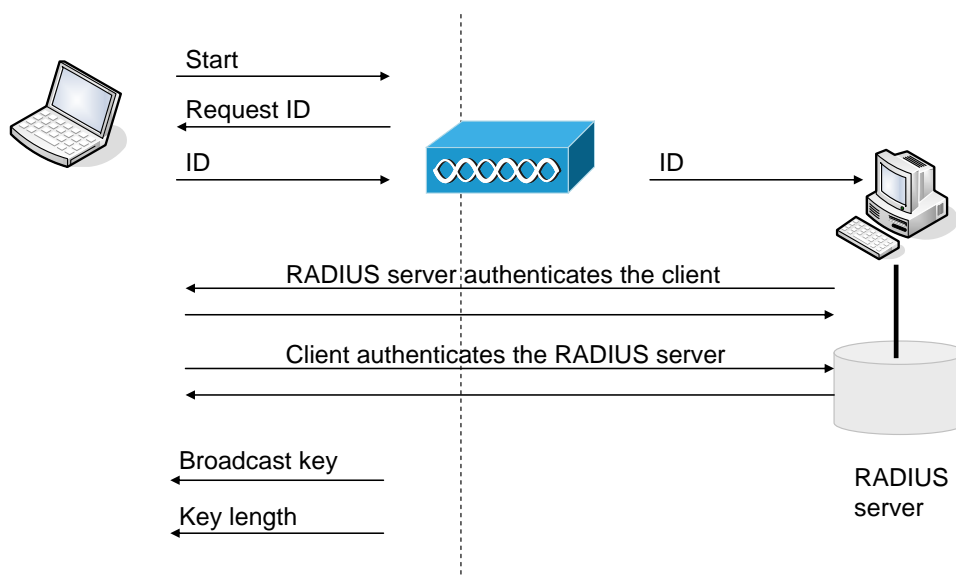


Figure 5.8 Basic message flow for 802.1X

Security weaknesses of RADIUS

A RADIUS servers provides a useful authentication method but suffers from many weaknesses, and work well within organizational infrastructure, but not between differing domains. RADIUS is especially weak, as it uses stateless UDP protocol, which allows for easier packet forging and spoofing. RADIUS uses UDP port 1812 for Authentication and 1813 for Accounting, and uses a shared secret key between the authenticator and the server. Particular problems for RADIUS include:

- **Brute-forcing of user credentials.** A malicious user can continually access the RADIUS server with a range of user ID and associated passwords, and RADIUS may eventually return a success authentication if a match is found.
- **Denial of service.** RADIUS uses UDP, which is connectionless, thus it is difficult to determine malicious from non-malicious UDP packets on ports 1812 and 1813.

- **Session replay.** There is very little authentication of the messages involved in RADIUS, thus malicious users can reply valid ones back into the next at future times.
- **Spoofed packet injection.** There is very little authentication of data packets built into RADIUS, and it can thus suffer from spoofed packet injection.
- **Response Authenticator Attack.** RADIUS uses an MD5-based hash for the Response Authenticator, thus if an intruder captures a valid Access-Request, Access-Accept, or Access-Reject packet sequence, they can launch a brute force attack on the shared secret. This is because the intruder can compute the MD5 hash for (Code+ID+Length+RequestAuth+Attributes), as most of the parts of the Authenticator are known, and can thus focus on the shared secret key.
- **Password Attribute-Based Shared Secret Attack.** Intruders can determine the share secret key but attempting to authenticate using a known password and then capturing the resulting Access-Requestpacket. After this they can then XOR the protected portion of the User-Password attribute with the password that they have used. A brute-force attack can then be done on the shared secret key
- **Shared Secret.** The basis methodology of RADIUS is that the same shared secret by many clients. Thus weakly protected clients could reveal the secret key.

Other weaknesses include:

- **User Password-Based Attack.**
- **Request Authenticator-Based Attacks.**
- **Replay of Server Responses.**

5.8 OpenID

OpenID is one method of creating a federated identity management system. It now includes Google mail profile, along with major organizations such as AOL, BBC, PayPal and Verisign. It uses a URL to identify the user, such as:

<http://billbuchanan.myopenid.com/>

The site hosting the identity then has a list of the accesses for the identity, and sites visited. It is an open system, and can be used as a single-sign-on for access to Web sites. Along with this it supports multiple forms of authentication, such as for smart cards, passwords and biometrics. This is set not by the Web site, or by the protocol, but by the OpenID provider. Figure 5.9 shows an example of the creation of an OpenID account, and Figure 5.10 shows how this can be used to log into a site which supports OpenID.

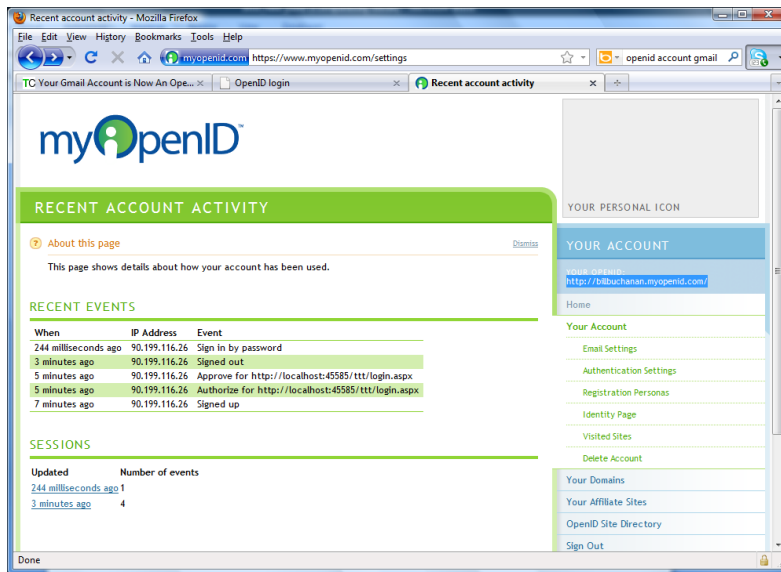


Figure 5.9 myOpen ID account



Figure 5.10 myOpen ID login

5.9 Kerberos

The major problem with current authentication systems is that they are not scalable, and they lack any real form of proper authentication. A new authentication architecture is now being proposed, which is likely to be the future of scalable authentication infrastructures – Kerberos. It uses tickets which are gained from an Identity Provider (IP – and also known as an Authentication Server), which are trusted to provide an identity to a Relying Party (RP). The basic steps are:

Client to IP:

- A user enters a username and password on the client.
- The client performs a one-way function on the entered password, and this becomes the secret key of the client.

- The client sends a cleartext message to the IP requesting services on behalf of the user.
- The IP checks to see if the client is in its database. If it is, the IP sends back a session key encrypted using the secret key of the user (MessageA). It also sends back a ticket which includes the client ID, client network address, ticket validity period, and the client/TGS (Ticket Granting Server) session key encrypted using the secret key of the IP (MessageB).
- Once the client receives messages A and B, it decrypts message A to obtain the client/TGS session key. This session key is used for further communications with IP.

Client-to-RP:

- The client now sends the ticket to the RP, and an authentication message with the client ID and timestamp, encrypted with the client session key (MessageC).
- The RP then decrypts the ticket information from the secret key of the IP, of which it recovers the client session key. It can then decrypt MessageD, and sends it back a client-to-server ticket (which includes the client ID, the client network address, validity period, and the client/server session key). It also sends the client/server session key encrypted with the client session key.

The Kerberos principle is well-known in many real-life authentication, such as in an airline application, where the check-in service provides the authentication, and passes a token to the passenger (Figure 5.11). This is then passed to the airline security in order to board the plane. There is thus no need to show the form for the original authentication, as the passenger has a valid ticket. Figures 5.12 and 5.13 show the detail of the Kerberos protocol which involves an Authentication Server, and a Ticket Grant Server.

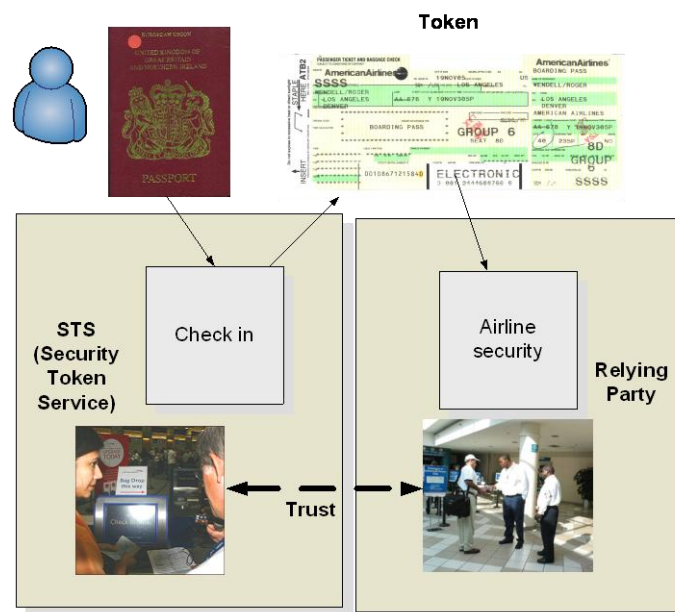


Figure 5.11 Ticketing authentication

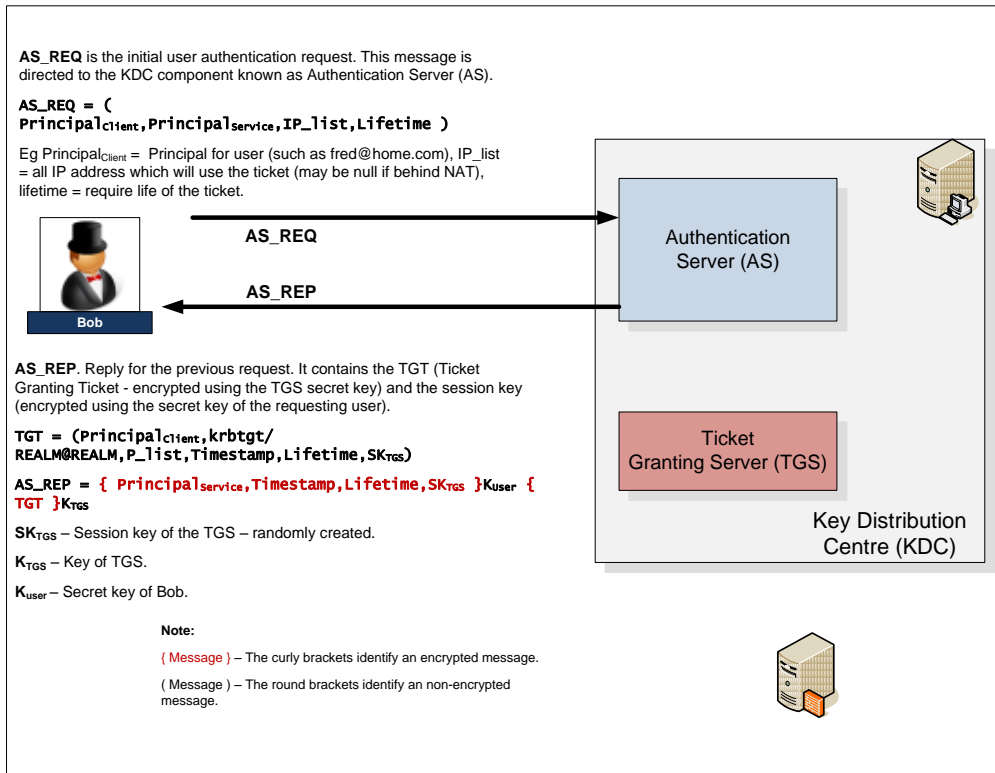


Figure 5.12 Kerberos protocol

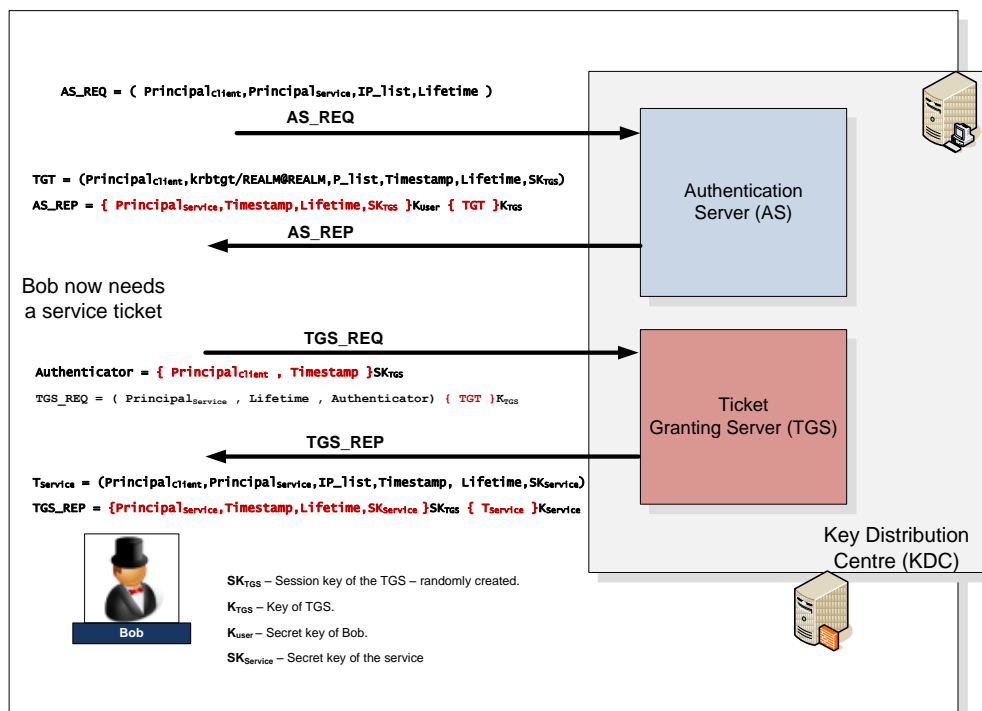


Figure 5.13 Kerberos protocol

Microsoft CardSpace

The Microsoft .NET 3.0 framework has introduced the CardSpace foundation framework, which uses Kerberos as its foundation. For this it defines a personal card,

which is encrypted and created by the user, and contains basic users details on the user, such as their name, address, email address, and so on. A **managed card** is created by an IP (Identity Provider) and validates the user. The managed card thus does not keep any personal details on login parameters and bank card details (as these are kept off-site). The user can thus migrate one from machine to another, and migrate their card (Figure 5.14). A personal card, of course, does not require an IP, and a card can be passed directly to the RP (Figure 5.15).

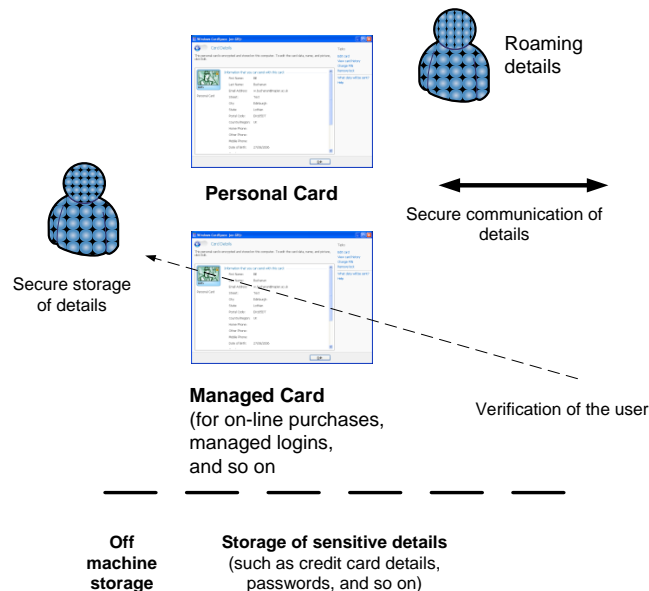


Figure 5.14 Personal and managed cards

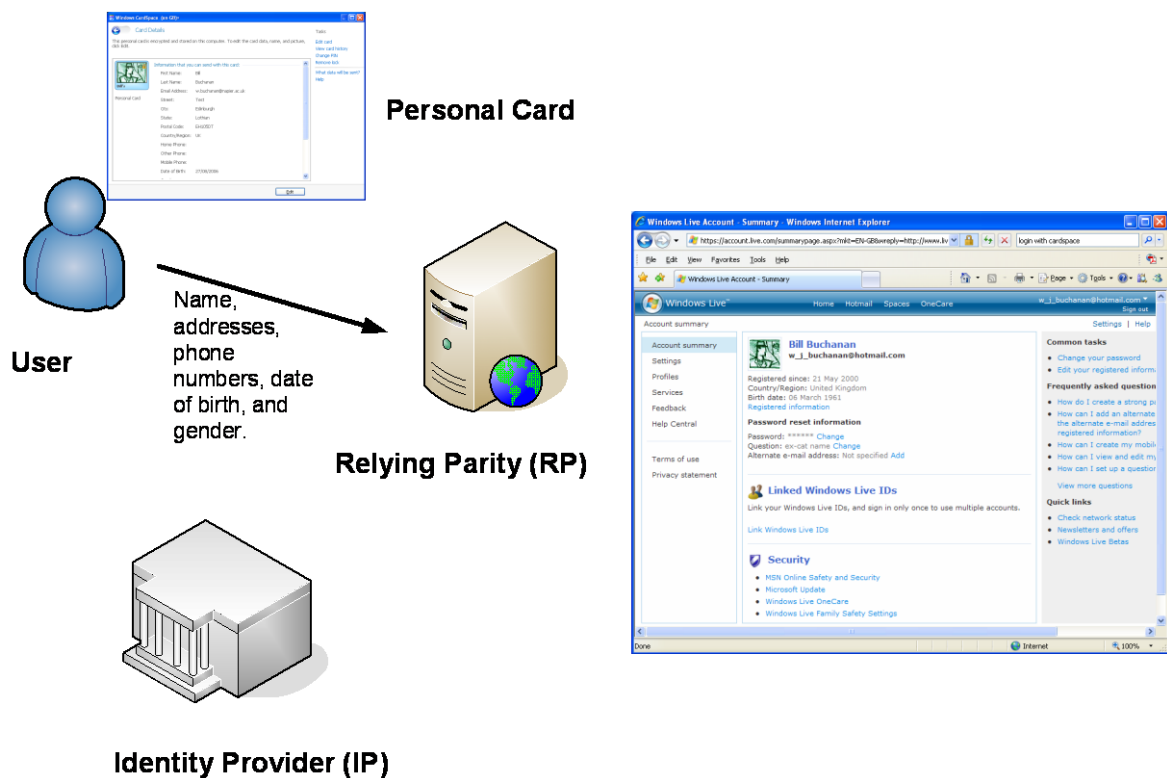


Figure 5.15 Personal cards

5.10 WS-*

A major problem with the interconnection of differing types of systems involves the transfer to data between them, and in the differences of the protocols used. To overcome this the WS-* infrastructure is being proposed as a way for the interconnection of systems using an open standard. This is illustrated in Figure 5.16. It can be seen that the infrastructure can use Kerberos or the traditional PKI method for identification.

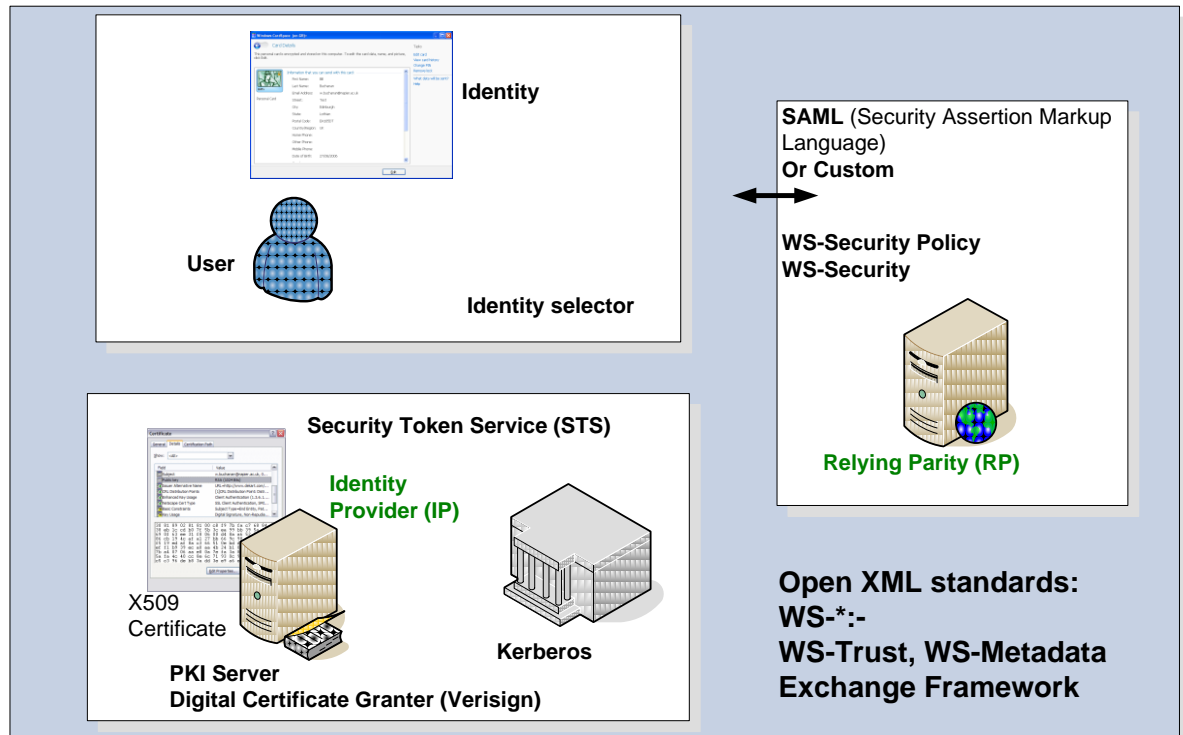


Figure 5.16 Standardized protocols

5.6.1 SAML

Security Assertion Markup Language (SAML) uses XML, and is a proposed method for interconnected between authentication and authorization infrastructures over multiple domains. It also focuses on providing a SSOs (Single Sign-On). The SAML the user is defined as the principal, and has an at least one identity provider. After the identity has been provided, access control can then be defined based on this. Figure 5.17 shows this type of infrastructure.

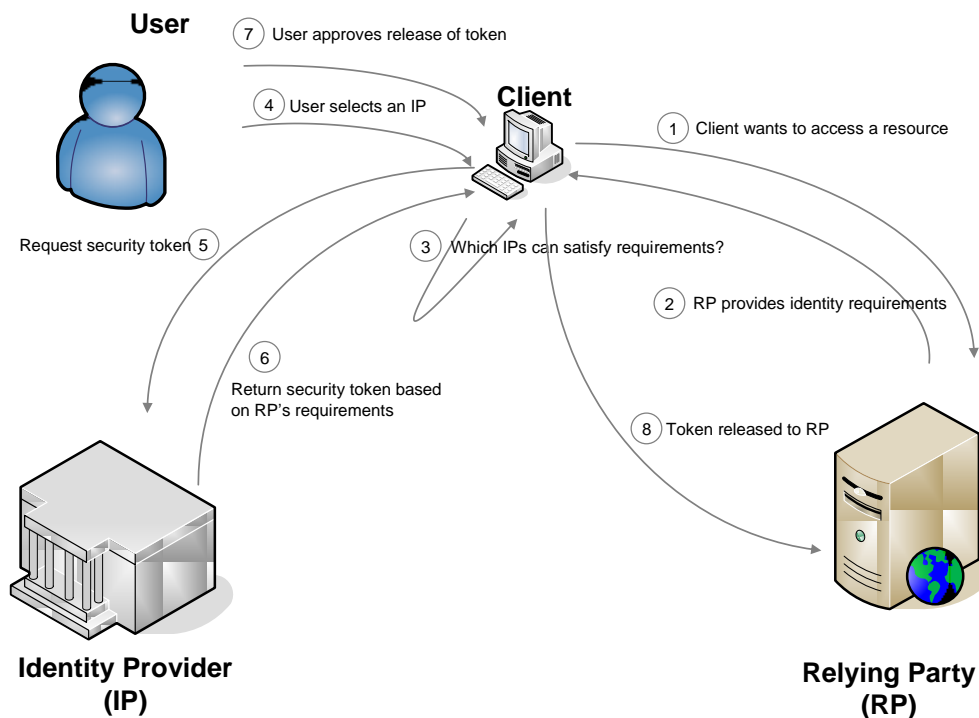


Figure 5.17 Managed cards

SAML uses XML-based messages to detail whether users are authenticated and the types of rights, roles and level of access that they have. It supports a wide range of protocols including HTTP, SMTP, FTP and SOAP, and so on. The language defines three main elements:

- **SAML Assertions.** These are: **Authentication** assertions (which assert that the user have proven their identity); **Attribute** assertions (which contains information about the user, such as when their limits are); and **Authorization** decision assertions (these define when the user can actually do).
- **Protocol.** This defines method that SAML uses to get gets assertions, such as using SOAP over HTTP (which is the most common method at the present).
- **Binding.** This defines how SAML message are exchanged, such as with SOAP messages.

Figure 5.18 shows an example of a SSO with a service provider. In this case the user connects to a service (such as for email), and the Service Provide creates a SAML request, and sends back the SSO URL for the user to authenticate to. The browser is then redirected to the SSO URL, where the Identity Provider parses the received SAML and generates a SAML response, which is then sent back to the user's browser. This is then sent to the Service Provider, and should then provide access to the service. An Assertion Consumer Service is used at Step 8 to check the assertions.

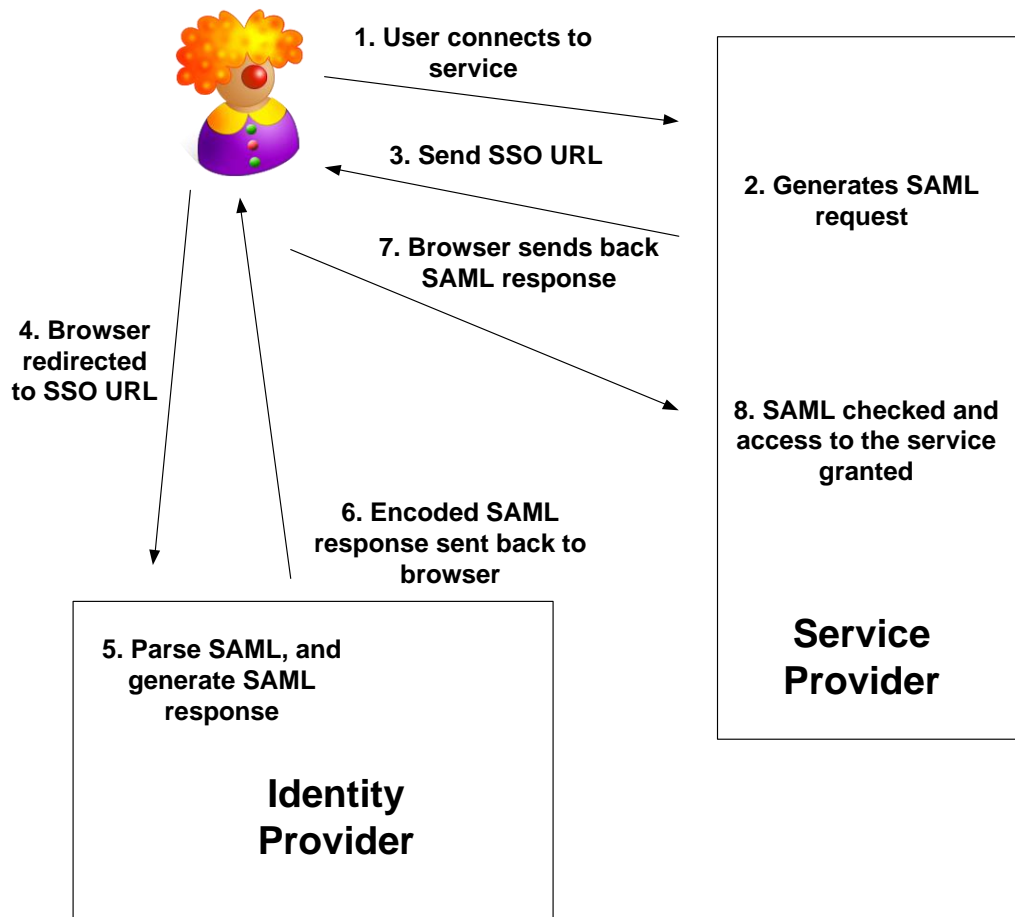


Figure 5.18 SSO

To authenticate from one site to another:

- A user authenticates to WebSite1.
- The user goes to WebSite2 (<http://www.WebSite2.com>), but is redirected to the SAML server for WebSite1.
- The SAML service adds on a partner ID (known as an artifact parameter name) and creates a secure connection (https) with SAML code which defines the identity of the user and their rights:

<https://www.WebSite2.com?SAMLart=>

- The received URL is then directed to the SAML server on WebSite2, and the service communicates with the SAML server on WebSite1, and thus verifies the identity and rights.

An assertion is defined between:

```
<saml:Assertion ...>
</saml:Assertion>
```

And a sample is (Ref: <http://identitymeme.org/doc/draft-hodges-learning-saml->

00.html):

•

```
1 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3   xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
4   <Issuer>
5     example.com
6   </Issuer>
7   <Subject>
8     <NameID
9       Format=
10        "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
11       Alice@example.com
12     </NameID>
13     <SubjectConfirmation
14       Method="urn:oasis:names:tc:SAML:2.0:cm:sender-vouches"/>
15   </Subject>
16   <Conditions NotBefore="2003-04-17T00:46:02Z"
17     NotOnOrAfter="2003-04-17T00:51:02Z">
18     <AudienceRestriction>
19       <Audience>
20         example2.com
21       </Audience>
22     </AudienceRestriction>
23   </Conditions>
24   <AttributeStatement>
25     <saml:Attribute
26       xmlns:x500=
27        "urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
28       NameFormat=
29        "urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
30       Name="urn:oid:2.5.4.20"
31       FriendlyName="telephoneNumber">
32       <saml:AttributeValue xsi:type="xs:string">
33
34         +1-888-555-1212
35       </saml:AttributeValue>
36     </saml:Attribute>
37   </AttributeStatement>
38 </Assertion>
```

5.11 Access Control

There are three main methods of access control on systems (Figure 5.19):

- **Role-based access control.** This involves defining roles in the organisation, and then defining rights based on the role. Users can thus join several groups, and be assigned rights based on their current role(s).
- **Mandatory access control.** This involves allowing the operating system to constrain access.
- **Discretionary access control.** This involves the owner of the entity defining the rights for access control.

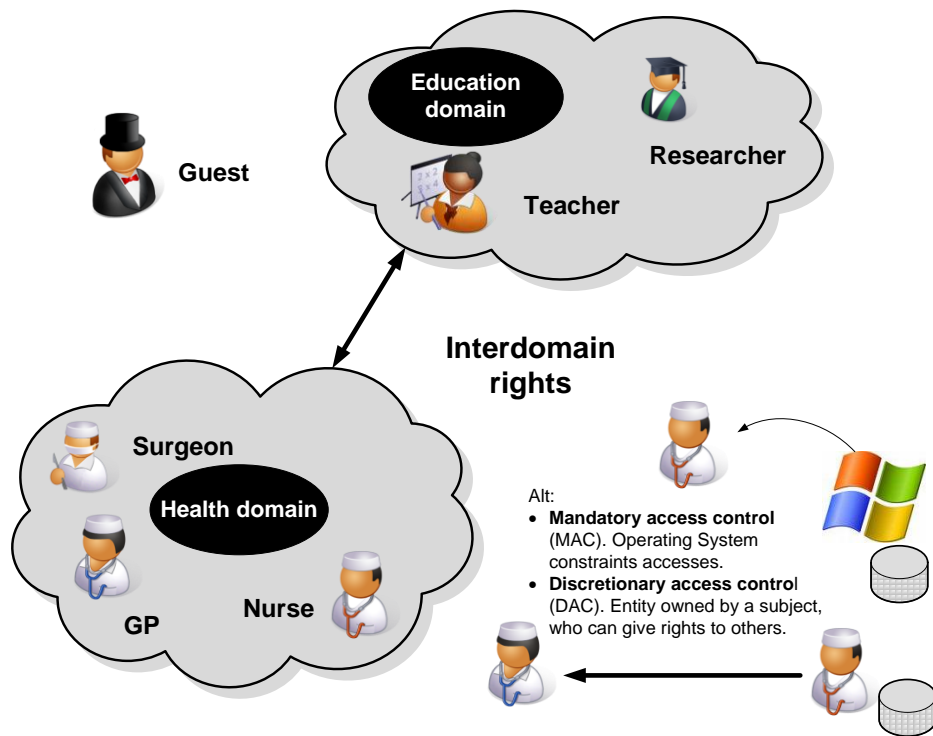


Figure 5.19 Access control

5.10.1 XACML

XACML (eXtensible Access Control Markup Language) defines a method of defining the access control policy language in an XML, along with a processing engine which defines how policies are interpreted. It contains:

- Policy Administration Point (PAP). This manages the policies
- Policy Decision Point (PDP). This evaluates the policy and then issues authorization decisions.
- Policy Enforcement Point (PEP). This intercepts user's access request to a resource and enforces the decisions made by the PDP.
- Policy Information Point (PIP). This is used to provide important attribute data that can be used by the PDP.

An example of a XACML file which permits the administrator to any operation on any Fedora repository service (<http://www.fedora-commons.org/download/2.2/userdocs/server/security/AuthorizationXACML.htm#DEFAULT>):

```
<Policy PolicyId="administrator"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-
applicable">
  <Description> </Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">administrator</AttributeValue>
          <SubjectAttributeDesignator AttributeId="fedoraRole"
MustBePresent="false" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
</Policy>
```

```

    </Subject>
  </Subjects>
<Resources>
  <AnyResource/>
</Resources>
<Actions>
  <AnyAction/>
</Actions>
</Target>
<Rule RuleId="1" Effect="Permit"/>
</Policy>

```

5.12 Tutorial

The main tutorial is at:

On-line tutorial: <http://buchananweb.co.uk/adv05.html>

The main tutorial is in NetworkSims ProfSIMs at:

NetworkSims.com ProfSIMs(c). Author: PROFSIMS Version: 6.7.0 (Build 140111)

Home BK FWD Menu Cisco Check Juniper Books router rip Exit

Test 1

- **Sample test. [Alt].**
- **Get Toolkit 1.12 [Here].** Recently added: Network Interface details (Network tab), SQL query analysis, and more analysis of HTTP response (HTTP). [SQL demo]
- **Podcast:**
 - iPod/iPhone Web browser
 - Podcast (iPhone): <http://itunes.apple.com/podcast/advanced-security-network/id357401454>
 - Podcast (Desktop): http://buchananweb.co.uk/asnf_full.xml
- **Sample network forensic logs [DNS] [FTP] [Hping_FIN] [Hping_Ping_Scan] [Hping_Port80] [Hping_Port80_SYN][Hping_SYN][Hydra_FTP] [Hydra_Telnet][NMAP][Ping][Spoof Address] [Telnet] [Trace route] [Web page]**
- **Spreadsheet checker for inferred marks [Here].**

Unit 5: Web Infrastructure

- **Lab.**
- **Lecture.**
- **Tutorial.**

Unit 6: Cloud

- **Lab [Network Forensics Trace] [Toolkit].**
- **Lecture. [PPT - with additional slides].**
 - Demo of S3CMD for Amazon S3.
 - EC2 tools
 - Real-life demo of instance creation.
 - Creating multiple instances from a user-generated AMI.
 - Creating Linux image demo (Lab 10). [Lab 10]

6 Cloud/Grid Computing

On-line lecture: <http://buchananweb.co.uk/adv/unit06.html>

6.1 Objectives

The key objectives of this unit are to:

- Provide an introduction to cluster, grid and cloud infrastructures.
- Define an example of grid computing, and its advantages.
- Show an example of using a Cloud Infrastructure.

6.2 Introduction

The computing power of computers increases by the year, as the number of transistors that can be fitted onto a piece of silicon increase. This has led to vast processing potential, and massive amount of local memory and storage space. Each machine, though, is limited in its resources, with a limit on the actual processing throughput, a limit on their local memory, and a limit on their storage space. The Internet, though, has given us access to a great deal of resources which might be local to the organisation, such as on local servers, or on remote systems. This also matches with the changes in architecture for systems, which have moved from thick-clients, where most of the resources are installed locally, towards thin-clients, where most of the software and data storage occurs on a remote system, and the local client is used to access the resources, and thus most of the computing is on the remote server. This type of architecture has many advantages, including making systems more robust, and in backing-up data on a regular basis.

The major recent changes have also included computers with multiple processors, where each processor can simultaneously run a task, and even a separate operating system. This allows processes to be moved around a network, and find the computing resources required, at any given time. Thus applications, operating systems, and even complete machines, are not actually physically tied-down to any specific hardware, and can thus move around the network, with fewer constraints on the actually running of the software. Figure 6.1 shows an example of different types of this distribution of processing and storage. In a clustered architecture, computers of the same type of operating system and architecture are brought together, with an interface layer which allows all the computers to be interfaced to as a single entity, and then the cluster management software (such as VMWare ESX) manages the resources for processing and storage, depending on the resources within the cluster. Figure 6.1 illustrates some of the main architectures which are used to distribute the processing of tasks.

Clusters are often specialized computing infrastructures. An alternative to this is to use the resources on a wide range of computers which have resources to spare. For example many computers spend most of their time with a less than 5% CPU utilization, and could thus offer some of the spare CPU resource to another application (as

long as it was trusted, of course). This is the main concept behind grid computing, where a distributed computer can be setup, with a range of computer architectures and operating systems, and spread over a large geographical area. Normally with grid computing the required resources would be mapped to physical computers, which had resources free at a given time. The advantage with **Cloud** infrastructures is that the actual resource does not need to exist, until it is actually required. In this way computing resources can be created, consumed and, even deleted, whenever they are required. In this way, a hardware, network, server, or service instance can be created at any given time, and then deleted when not required anymore.

Cluster, grid and cloud infrastructures give the opportunity for providing computing resources as a utility, and thus provided as a pay-as-you-go service, in the same way that other utilities do, such as for electricity, water and gas.

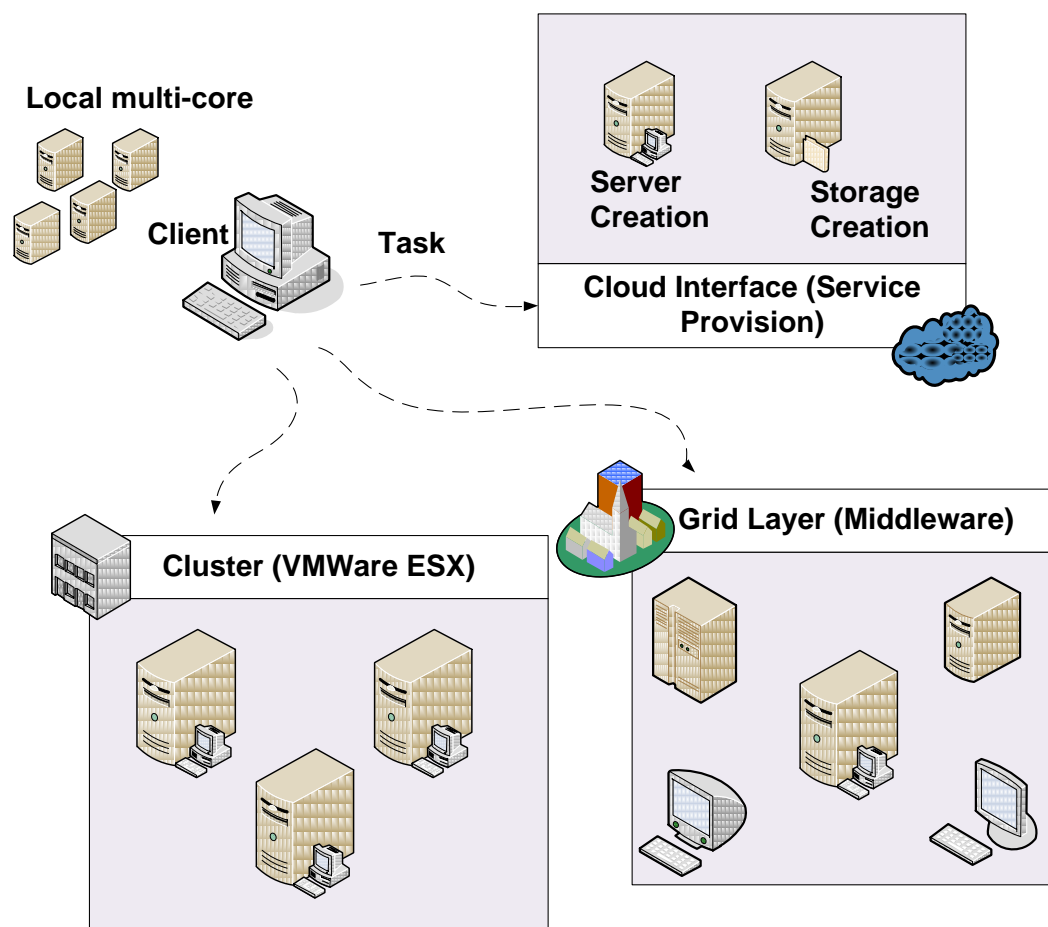


Figure 6.1 Cluster, grid and cloud

6.3 Grid Computing

Grid computing involves using the computing resources from multiple domains to implement a given task, that might be difficult on a local computer. For example a grid network could be setup to brute force an encrypted file, using a wide range of computers, each which could be given a part of the key space. This type of task would take a lengthy amount of time on a stand-alone computer, but the time taken can be considerably reduced if it is done on a parallel basis. It is thus key to be able to

split a task into a number of sub-tasks, each of which are fairly independent from each other, or where there is defined synchronization between the sub-tasks. For example if we take a very simple equation:

$$f(x) = (x^2+5) \times (x^3+x^2+1) \times (4x^4+7x^2)$$

We could split this into three tasks, each which could be processed independently:

$$T1 = (x^2+5)$$

$$T2 = (x^3+x^2+1)$$

$$T3 = (4x^4+7x^2)$$

The result would then have to be gather back from all the processing elements to produce the result.

The distributed infrastructure can either be localized within an organization, or can be distributed across different domains, and thus create a trusted infrastructure.

6.3.2 Grid middleware

Grid applications are normally built around grid middleware, including the Globus Toolkit (which has been used to simulate the gravitational effects of black hole collisions), NGrid, gLite (which focuses on Linux systems), and UNICORE (Uniform Interface to Computing Resources). For example the Alchemi framework contains a Manager, which runs a service on a machine on port 9000, which a client can connect to.

6.3.1 Grid computing applications

Grid computing offers an almost infinite computing resource, using clustered computers which intercommunicate to perform large scale tasks, typically ones which involved intensive scientific, mathematical or search-type operations. In a clustered environment we normally have an array of closely coupled computers, which are constrained with a local environment, and are typically of the same type (homogeneous), whereas grid computing typically uses more loosely coupled computer, which are often not of the same type (heterogeneous) and are often widely dispersed. Grid computing also differs from clustered environments, in that each instance of them tend to focus on the one type of application, such as for cancer drug analysis, brute force search for encryption keys, and so on.

Distributed.net

One of the most scalable applications in terms of processing is the brute force analysis of encrypted text. As an example, let's try a 64-bit encryption key which gives us: 1.84×10^{19} combinations (2^{64}). If we now assume that we have a fast processor that tries one key every billionth of second (1GHz clock), then the average³ time to crack the code will be:

³ The average time will be half of the maximum time

$$T_{average} = 1.84 \times 10^{19} \times 1 \times 10^{-9} \div 2 \approx 9,000,000,000 \text{ seconds}^4$$

It will thus take approximately 2.5 million hours (150 million minutes or 285 years) to crack the code, which is likely to be strong enough in most cases. Unfortunately as we have seen, the computing power often increases by the year, so if we assume a doubling of computing power, then:

Date	Hours	Days	Years
0	2,500,000	104,167	285
+1	1,250,000	52,083	143
+2	625,000	26,042	71
+3	312,500	13,021	36
+4	156,250	6,510	18
+5	78,125	3,255	9
+6	39,063	1,628	4
+7	19,532	814	2
+8	9,766	407	1
+9	4,883	203	1
+10	2,442	102	0.3
+11	1,221	51	0.1
+12	611	25	0.1
+13	306	13	0
+14	153	6	0
+15	77	3	0
+16	39	2	0
+17	20	1	0

we can see that it now only takes 17 years to crack the code in a **single day!** If we then apply parallel processing, the time to crack reduces again. In the following an array of 2x2 (4 processing elements), 4x4 (16 processing elements), and so on, are used to determine the average time taken to crack the code. If, thus, it currently takes 2,500,000 minutes to crack the code, it can be seen that by Year 6, it takes less than one minute to crack the code, with a 256x256 processing matrix.

Processing Elements	Year 0 (minutes)	Year 1 (min)	Year 2 (min)	Year 3 (min)	Year 4 (min)	Year 5 (min)	Year 6 (min)	Year 7 (min)
1	2500000	1250000	625000	312500	156250	78125	39062.5	19531.3
4	625000	312500	156250	78125	39062.5	19531.3	9765.7	4882.9
16	156250	78125	39062.5	19531.3	9765.7	4882.9	2441.5	1220.8
64	39063	19531.5	9765.8	4882.9	2441.5	1220.8	610.4	305.2
256	9766	4883	2441.5	1220.8	610.4	305.2	152.6	76.3
1024	2441	1220.5	610.3	305.2	152.6	76.3	38.2	19.1
4096	610	305	152.5	76.3	38.2	19.1	9.6	4.8
16384	153	76.5	38.3	19.2	9.6	4.8	2.4	1.2
65536	38	19	9.5	4.8	2.4	1.2	0.6	0.3

The ultimate in distributed applications is to use unused processor cycles of ma-

⁴ 9,223,372,036 seconds to be more precise

chines connected to the Internet. For this applications such as **distributed.net** allow the analysis of a key space when the screen saver is on (Figure 6.2). It has since used the method to crack a number of challenges, such as in 1997 with a 56-bit RC5 Encryption Challenge. It was cracked in 250 days, and has since moved on, in 2002, to crack 64-bit RC5 Encryption Challenge in 1,757 days (with 83% of the key space tested). The current challenge involves a 72-bit key.

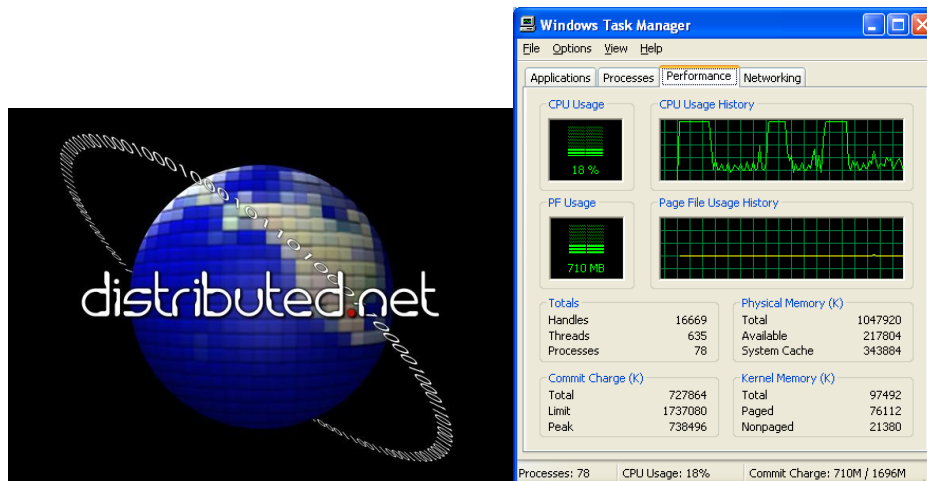


Figure 6.2 Distributed.net

6.4 Cloud Computing

Cloud computing is a natural extension of a range of technologies, including the use of cluster, virtualization and grid computing. With this services can be created and consumed when they are required, including processing, storage and authentication services, each of which can be integrated to create software-as-a-software (SaaS). The major advantage of this is that users do not actually need, at any specific time, where the resources are, and how to consume them. The services themselves can be services actually currently running on a server, or can be a virtualized version. Many applications which consume the cloud infrastructure are based around a Web or a console interface.

The key characteristics of Cloud Computing are (Src: NIST):

- On-demand self-service
- Ubiquitous network access
- Location independent resource pooling
- Rapid Elasticity
- Measured Service

6.4.1 Cloud abstraction

The interface to the Cloud can happen at a number of levels:

- **Hardware as a Service (HaaS)**. At the lowest level hardware can be provided as a service, such as for the provision of a cluster, which can reduce the investment in capital and operation, while increasing the reliability of the infrastructure.

- **Infrastructure as a Service (IaaS).** This involves creating an infrastructure for the application, such as in provide the basic infrastructure for servers required for an application. This includes processing, storage, networks, and other basic computing resources, which are under the control of the user. Figure 6.4 illustrates IaaS, and shows the integration with Amazon Web services.
- **Platform-as-a-Service (PaaS).** This layer provides a hooks for APIs which can be used to build applications, such as integrating security, defining the user interface, and providing a data storage service. Figure 6.5 illustrates PaaS, and show the integration with Microsoft Azure, Google Application Engine and Amazon EC2. This layer supports the deployment of a consumer-created application into a cloud infrastructure using standard tools such as Java, Python, and .Net).
- **Software-as-a-Service (SaaS).** This layer provide the software application directly to the user, who can then customize it as they require. It includes applications such as Google mail, Twitter, Hotmail, and so on.

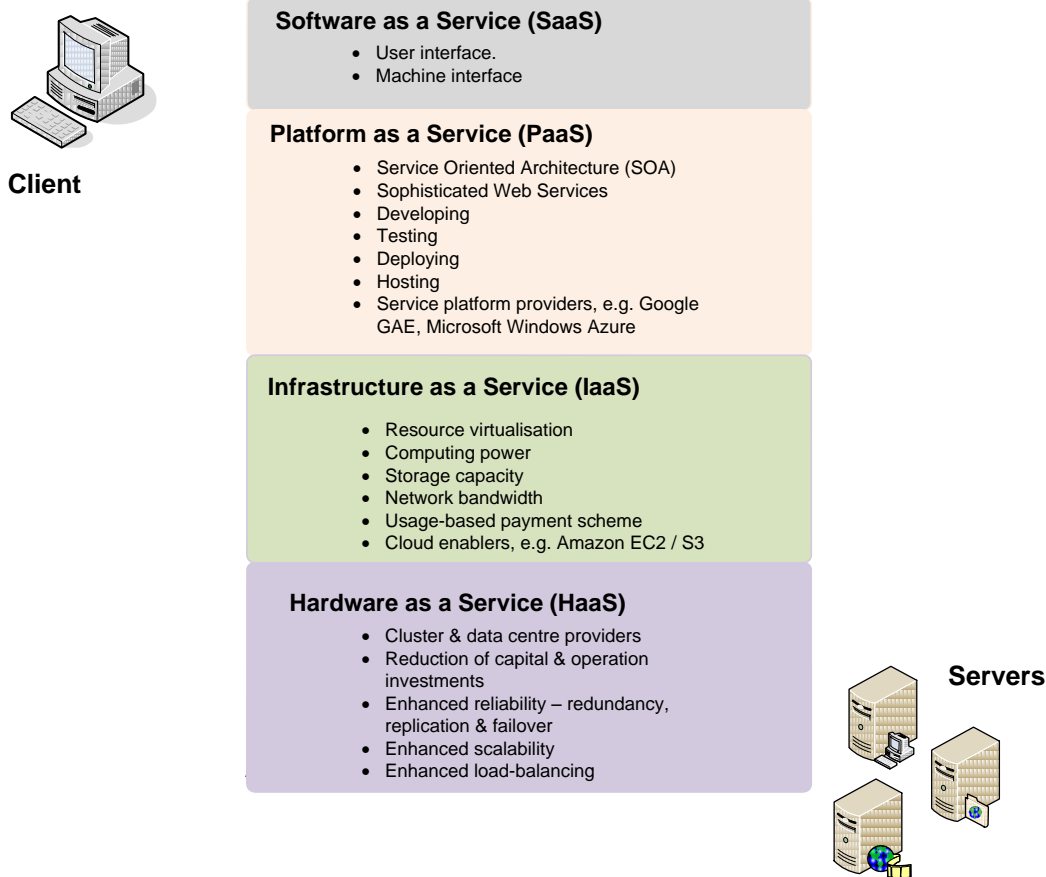


Figure 6.3 Cloud abstraction layers

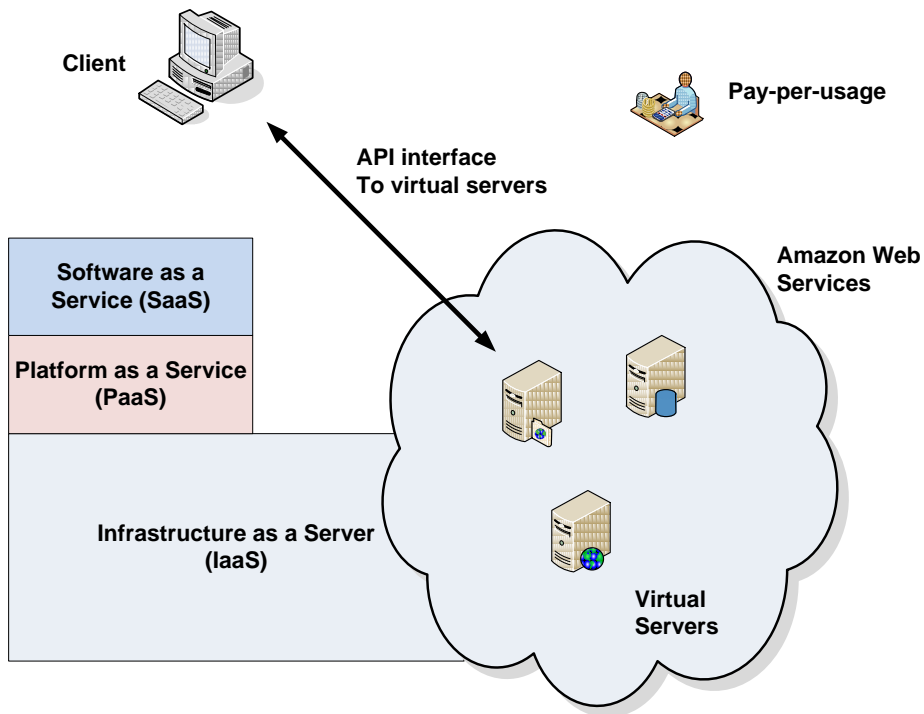


Figure 6.4 IaaS

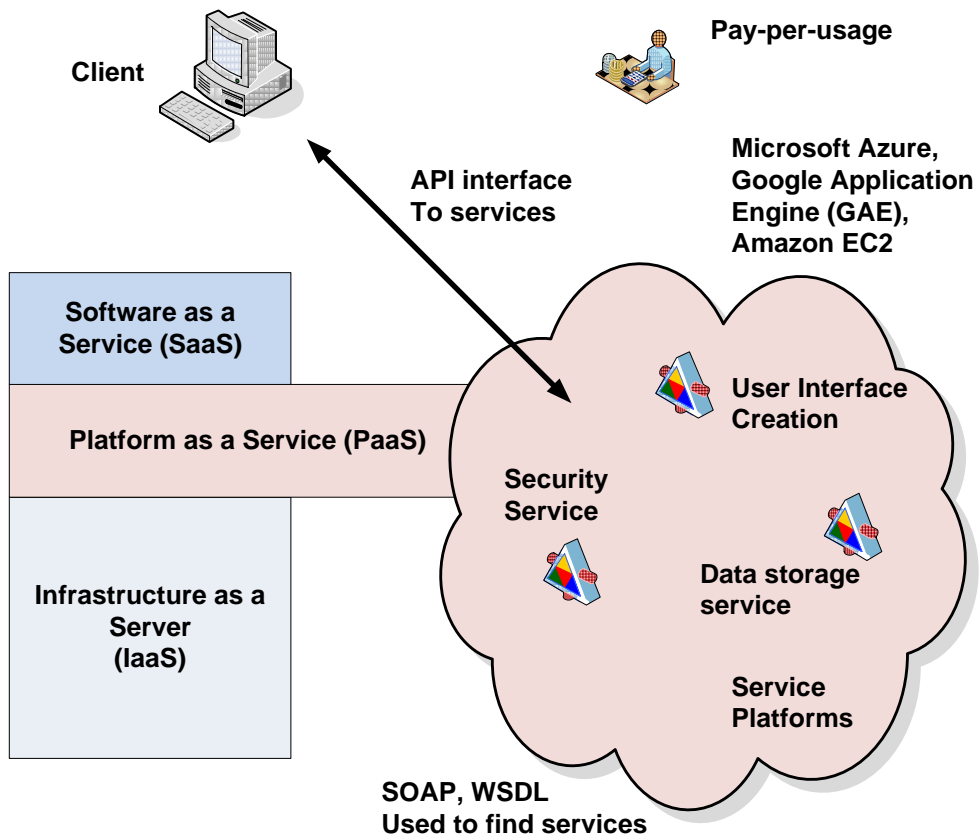


Figure 6.5 PaaS

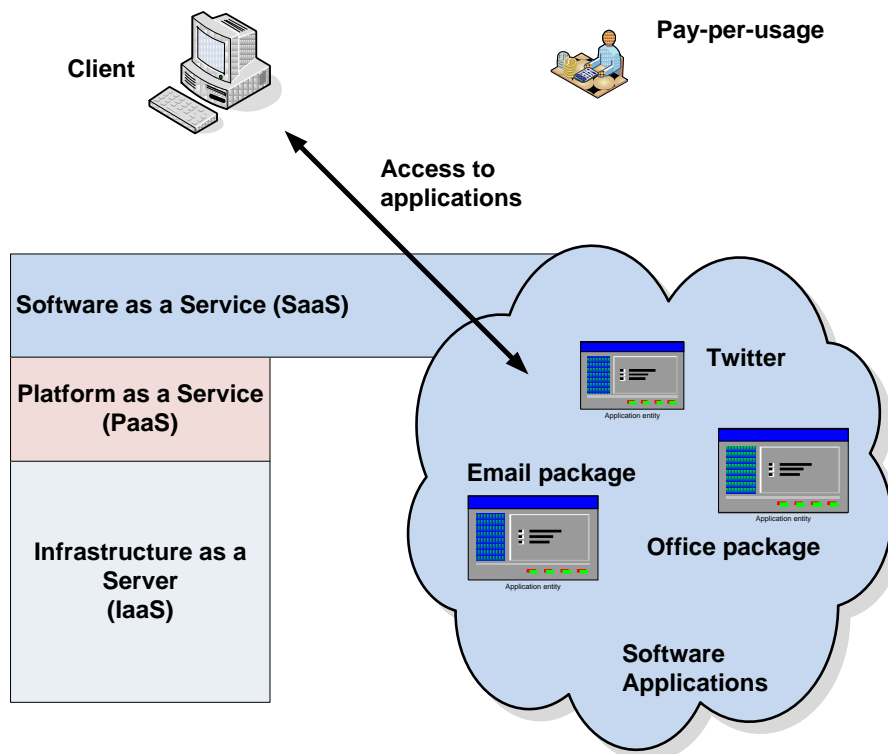


Figure 6.6 SaaS

6.5 Amazon Web Services

Amazon are one of the most advanced providers of Web services. This includes:

- **Amazon Elastic Cloud Compute** (Amazon EC2). This is the core of the Amazon Cloud, and provides a Web services API to create, manage and delete virtual servers within the Amazon Cloud. This includes US and European data centres, and uses the Xen hypervisor for the management of the servers.
- **Amazon Simple Storage Service** (Amazon S3). This provides data storage with web services through APIs. It differs from normal filesystems in that it does not have a hierarchal structure. Instead it uses buckets, which are unique namespaces across all of the Amazon customers. It is thus not a filesystem, and is a Web service, thus applications need to be written which specifically store data into the S3 Cloud.
- **Amazon CloudFront**. This allows content to be placed close to the places where it is to be consumed, the content thus gets moved to the edge of the cloud to support rapid delivery of content.
- **Amazon Simple Queue Service** (Amazon SQS). This supports a grid infrastructure, where message can be passed to a queue, and then consumed by any subscribers.
- **Amazon SimpleDB**. This produces a mixture of structured data storage with the reliability of a traditional database.

6.5.1 Amazon E3

Amazon E3 uses buckets to store data. Initially a bucket is created with (in this case

the bucket is bill.bucket):

```
s3cmd mb s3://bill.bucket
```

where the name of the bucket which should be unique across the Amazon Cloud. The bucket can then be listed with:

```
s3cmd ls  
s3cmd ls s3://bill.bucket/
```

Next an object can be copied to the bucket with:

```
s3cmd put myfile.mp3 s3://bill.bucket/myfile.mp3
```

and got back with:

```
s3cmd get s3://bill.bucket/myfile.mp3 myfile.mp3
```

and finally to delete an object from a bucket, and to delete the bucket we have:

```
s3cmd del s3://bill.bucket/myfile.mp3  
s3cmd rb s3://bill.bucket
```

6.5.2 Amazon EC2

Amazon EC2 provides an excellent PaaS, and allows for the creation of data storage, virtual images, and so on, on a pay-per-usage basis. It initially creates a predefined AMI (Amazon machine image), which can be then customized to the given requirement. The storage can then be tied to the virtual image (that is deleted when the virtual server is deleted), or can be done as block storage, which will exist even when virtual images are deleted.

Figure 6.7 shows an example of a user creating a Windows 2008 virtual image. In this case the public DNS given is defined as:

```
ec2-204-236-199-96.compute-1.amazonaws.com
```

and this can be connected to by the Remote Desktop to allow access to the server image, such as with:

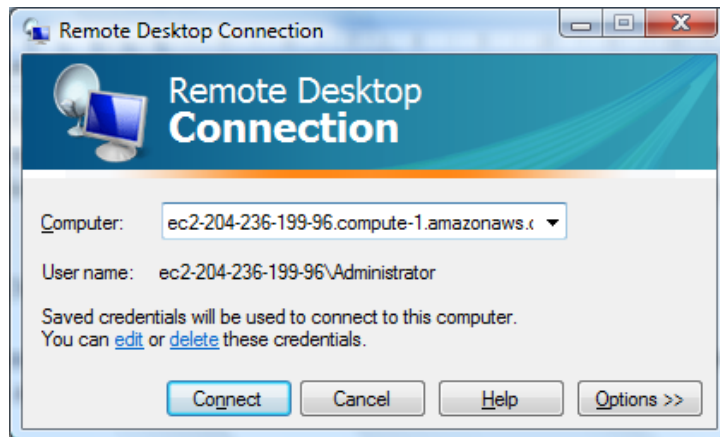


Figure 6.7 Remote connection to the virtual image

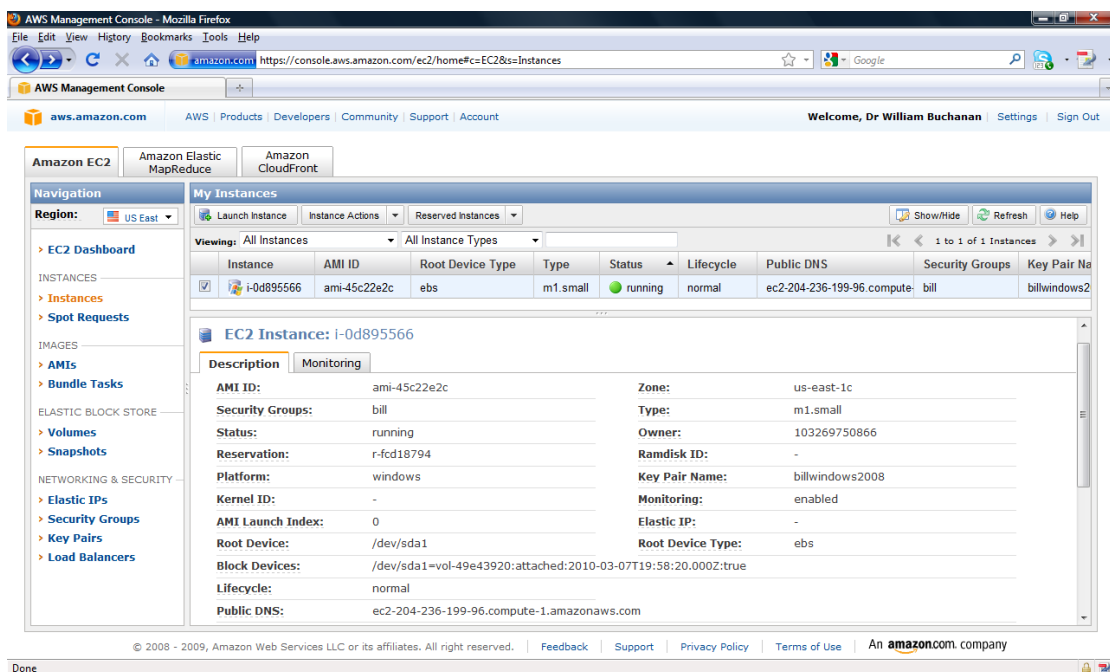


Figure 6.8 EC2 example

The remote desktop is then viewed, as in Figure 6.9. The user may then want to create an IIS server, and this is created, in Windows 2008 with an Add Role, and IIS Web Server. After this the `http://localhost` will work on the virtual image, but there will be no external Web access. This is because the virtual image is firewalled so that there is not external access. Figure 6.10 shows that HTTP access can then be enabled for port 80. Once this is setup, the Web server can then be accessed with:

`http://ec2-204-236-199-96.compute-1.amazonaws.com`

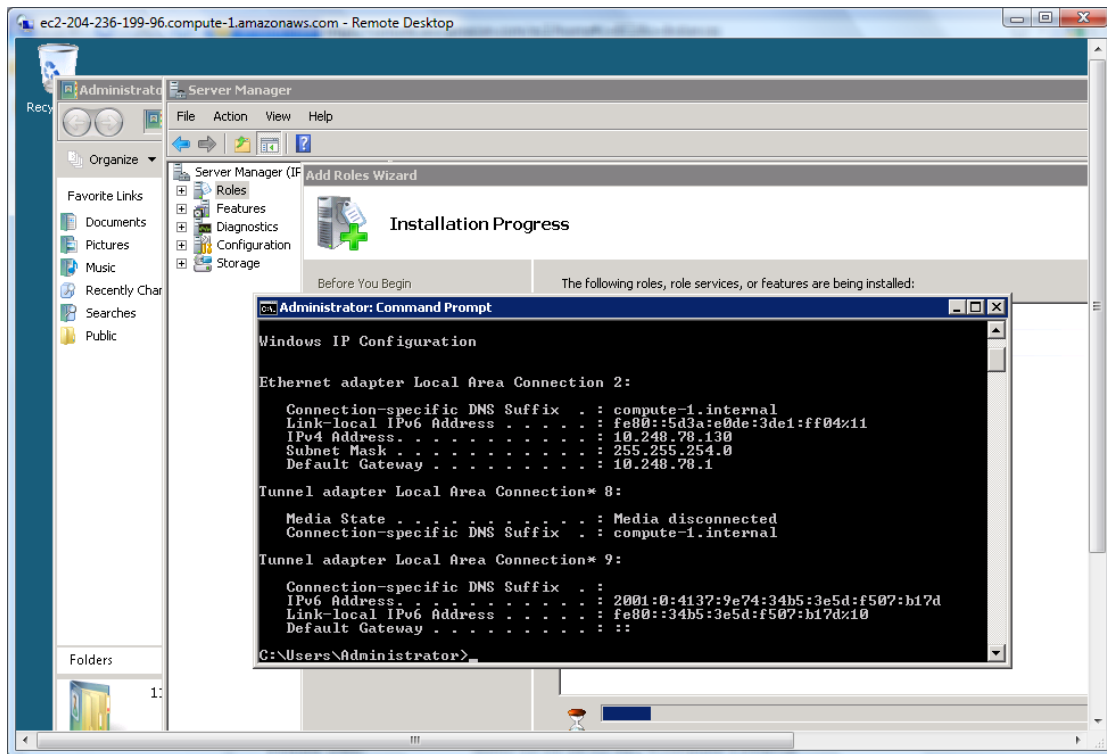


Figure 6.9 Amazon EC2 virtual image

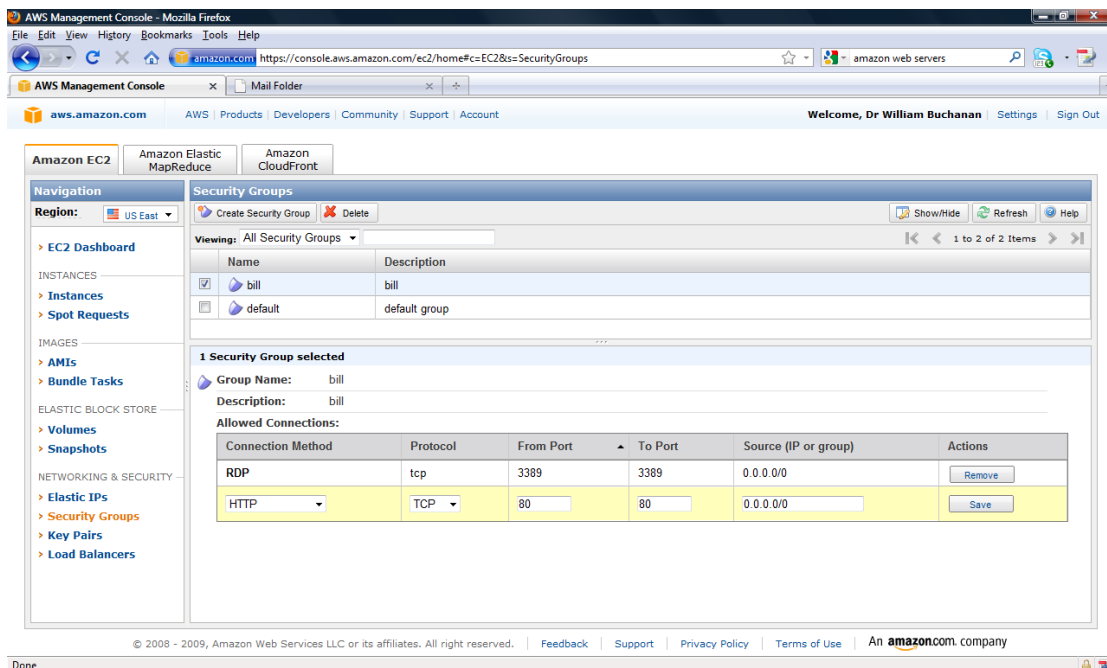


Figure 6.10 Enabling HTTP access

6.6 Installing EC2 and S3 command tools

In Ubuntu, the S3 command tools are installed with:

```
napier@ubuntu:~$ sudo apt-get install s3cmd
```

and then configured with:

```
napier@ubuntu:~$ s3cmd --configure
```

where the Amazon Web Service ID, and secret password are then defined. See the online video for more details, but a sample run is:

```
napier@ubuntu:~$ s3cmd ls
2010-03-08 20:44 s3://akiaiwumttazyst2i2aa-test-bucket12083
2010-03-09 08:17 s3://akiaiwumttazyst2i2aa-test-bucket13695
2010-03-09 08:18 s3://akiaiwumttazyst2i2aa-test-bucket17966
2010-03-08 20:42 s3://akiaiwumttazyst2i2aa-test-bucket25165
2010-03-08 20:43 s3://akiaiwumttazyst2i2aa-test-bucket27774
2010-03-08 20:44 s3://akiaiwumttazyst2i2aa-test-bucket31130
2010-03-08 20:42 s3://akiaiwumttazyst2i2aa-test-bucket39674
2010-03-09 12:16 s3://bill.bucket

napier@ubuntu:~$ s3cmd mb s3://bill.bucket2
Bucket 's3://bill.bucket2/' created

napier@ubuntu:~$ s3cmd put test.txt s3://bill.bucket2
test.txt -> s3://bill.bucket2/ [1 of 1]
 15 of 15 100% in 0s 27.65 B/s done
ERROR: S3 error: 400 (MalformedXML): The XML you provided was not well-formed or did
not validate against our published schema

napier@ubuntu:~$ s3cmd put test.txt s3://bill.bucket2/test.txt
test.txt -> s3://bill.bucket2/test.txt [1 of 1]
 15 of 15 100% in 0s 25.08 B/s done

napier@ubuntu:~$ s3cmd ls s3://bill.bucket2
2010-03-09 18:06 15 s3://bill.bucket2/test.txt

napier@ubuntu:~$ s3cmd ls s3://bill.bucket2
2010-03-09 18:06 15 s3://bill.bucket2/test.txt

napier@ubuntu:~$ s3cmd get s3://bill.bucket2/test.txt 1.txt
s3://bill.bucket2/test.txt -> 1.txt [1 of 1]
 15 of 15 100% in 0s 24.16 B/s done

napier@ubuntu:~$ s3cmd del s3://bill.bucket2/test.txt
File s3://bill.bucket2/test.txt deleted

napier@ubuntu:~$ s3cmd rb s3://bill.bucket2
Bucket 's3://bill.bucket2/' removed

napier@ubuntu:~$ s3cmd ls
2010-03-08 20:44 s3://akiaiwumttazyst2i2aa-test-bucket12083
2010-03-09 08:17 s3://akiaiwumttazyst2i2aa-test-bucket13695
2010-03-09 08:18 s3://akiaiwumttazyst2i2aa-test-bucket17966
2010-03-08 20:42 s3://akiaiwumttazyst2i2aa-test-bucket25165
2010-03-08 20:43 s3://akiaiwumttazyst2i2aa-test-bucket27774
2010-03-08 20:44 s3://akiaiwumttazyst2i2aa-test-bucket31130
2010-03-08 20:42 s3://akiaiwumttazyst2i2aa-test-bucket39674
2010-03-09 12:16 s3://bill.bucket
```

The EC2 command tools are installed with:

```
napier@ubuntu:~$ sudo apt-get install ec2-api-tools
```

Next the PEM files are created from the files created when the AWS account is created:

```
napier@ubuntu:~$ ls
1.txt      Documents      examples.desktop  Pictures      slap.1      test.txt
cert.pem   Downloads     hydra-5.4-src    private.pem  Templates   Videos
Desktop   en_GB.UTF-8   Music            Public       test2.txt
```


Where private.pem is the private key, and cert.pem is the digital certificate. An example of these is:

```
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAIBNJQqhKfqrvcOVCKQRpR5cFz1m
8PMWEZW3H/S9R3PV4QbLGJftczKj73iOnVN2oJ3Fcv0WNM1NSAwIdHwMYskFNVu1c7r3aDPIVG8R
BXD/...+JEiiQJAPD5EyMZA7oLB
awLNC7/lu3AdufVgc9CCFKDdlt+tKAw+lzSfungnzMPIrM/6OVLZ0I/7jVn0w+5wBhz5eI2KkQJA
GwANdPo8xH6GI0bLAuvNvPN0QS9k6fgEWtr07raGFFJp2j0cYGx4Z5NGPUUmpq6qBxXOONmhgule
ID8deq3IwA==
-----END PRIVATE KEY-----
```

Next the .bashrc file can be created with:

```
export EC2_HOME=~/.ec2
export PATH=$PATH:$EC2_HOME/bin
export EC2_PRIVATE_KEY=private.pem
export EC2_CERT=cert.pem
```

The ec2 command can then be used, such as for viewing the instances:

```
napier@ubuntu:~$ ec2-describe-instances
RESERVATION r-fcd18794 103269750866 bill
INSTANCE i-0d895566 ami-45c22e2c                stopped billwindows2008      0
           m1.small 2010-03-09T16:27:01+0000      us-east-1c                  windows mon-
itoring-enabled
```

```
napier@ubuntu:~$ ec2-run-instances ami-45c22e2c
RESERVATION r-702b7818 103269750866 default
INSTANCE i-95cd14fe ami-45c22e2c                pending                    0m1.small
           2010-03-09T18:26:48+0000 us-east-1d                  windows monitoring-disabled

napier@ubuntu:~$ ec2-describe-instances
RESERVATION r-fcd18794 103269750866 bill
INSTANCE i-0d895566 ami-45c22e2c                stopped billwindows2008      0
           m1.small 2010-03-09T16:27:01+0000      us-east-1c                  windows mon-
itoring-enabled
RESERVATION r-702b7818 103269750866 default
INSTANCE i-95cd14fe ami-45c22e2c                pending                    0m1.small
           2010-03-09T18:26:48+0000 us-east-1d                  windows monitoring-disabled
```

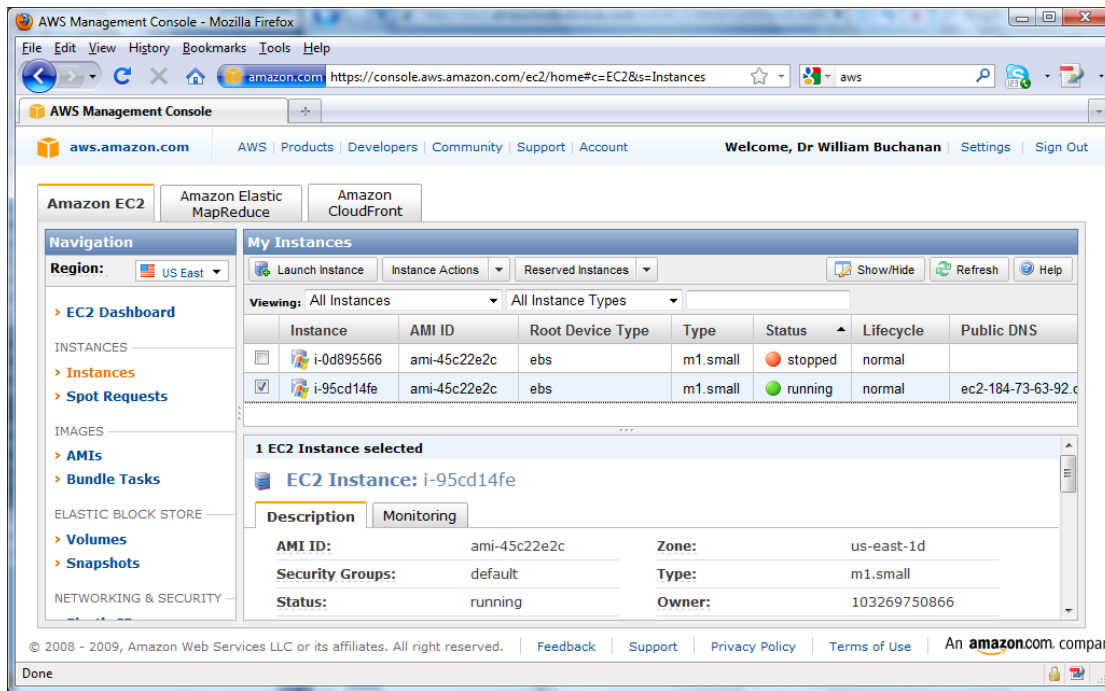
After a while we can see that the new instance (i-95cd14fe) has been created:

```
napier@ubuntu:~$ ec2-describe-instances
RESERVATION r-fcd18794 103269750866 bill
INSTANCE i-0d895566 ami-45c22e2c                stopped billwindows2008      0
           m1.small 2010-03-09T16:27:01+0000      us-east-1c                  windows mon-
itoring-enabled
RESERVATION r-702b7818 103269750866 default
INSTANCE i-95cd14fe ami-45c22e2c                ec2-184-73-63-92.compute-1.amazonaws.com ip-
10-242-63-161.ec2.internal running              0                          m1.small      2010-03-
09T18:26:48+0000 us-east-1d                  windowsmonitoring-disabled
```

And then to get rid of the instance:

```
$ ec2-terminate-instances i-0d895566
INSTANCE i-0d895566 running shutting-down
```

And the console also reports this new instance:



We can then open-up ports to allow the image to communicate with HTTP (for the default security group):

```

napier@ubuntu:~$ ec2-authorize --help
SYNOPSIS
    ec2auth (ec2-authorize)
    ec2auth [GENERAL OPTIONS] GROUP [SPECIFIC OPTIONS]
GENERAL NOTES
    Any command option/parameter may be passed a value of '-' to indicate
    that values for that option should be read from stdin.
DESCRIPTION
    Grant selected permissions to a specified group.
    The GROUP parameter is name of the group to grant this permission to.
GENERAL OPTIONS
    -K, --private-key KEY
        Specify KEY as the private key to use. Defaults to the value of the
        EC2_PRIVATE_KEY environment variable (if set). Overrides the default.
    -C, --cert CERT
        Specify CERT as the X509 certificate to use. Defaults to the value
        of the EC2_CERT environment variable (if set). Overrides the default.
    -U, --url URL
        Specify URL as the web service URL to use. Defaults to the value of
        'https://ec2.amazonaws.com' or to that of the EC2_URL environment
        variable (if set). Overrides the default.
    --region REGION
        Specify REGION as the web service region to use.
        This option will override the URL specified by the "-U URL" option and
        EC2_URL environment variable.
    -v, --verbose
        Verbose output.
    -?, --help
        Display this help.
    -H, --headers
        Display column headers.
    --debug
  
```

```

    Display additional debugging information.

--show-empty-fields
    Indicate empty fields.

--connection-timeout TIMEOUT
    Specify a connection timeout TIMEOUT (in seconds).

--request-timeout TIMEOUT
    Specify a request timeout TIMEOUT (in seconds).

SPECIFIC OPTIONS

-P, --protocol PROTOCOL
    tcp, udp or icmp (must be lower case). If not specified, the protocol
    defaults to tcp if source subnet is specified (or implied by default),
    or all-protocols if source group is specified (to ensure backwards
    compatibility)

-p, --port-range PORT-RANGE
    Range of ports to open. If the tcp or udp protocol are specified (or
    implied by default), then the range of ports to grant access to may
    optionally be specified as a single integer, or as a range (min-max).

-t, --icmp-type-code TYPE:CODE
    icmp type and code. If the icmp protocol is specified, then icmp type
    and code may optionally be specified as type:code, where both type and
    code are integers and compliant with RFC792. Type or code (or both) may
    be specified as -1 which is a wildcard covering all types or codes.

-o, --source-group SOURCE-GROUP [--source-group...]
    Network source from which traffic is to be authorized, specified as
    an EC2 security group name, e.g. default. This may be specified more
    than once to allow network traffic from multiple security groups.

-u, --source-group-user SOURCE-GROUP-USER [--source-group-user...]
    The owner of the security group specified using -o. If specified only
    once, the same user will be used for all specified groups. However, if
    specified once per -o, each user is mapped to a group in order.
    Anything else is invalid.

-s, --source-subnet SOURCE-SUBNET
    The network source from which traffic is to be authorized, specified
    as a CIDR subnet range, e.g. 205.192.8.45/24. This may be specified
    more than once to allow traffic from multiple subnets.

napier@ubuntu:~$ ec2-authorize default -p 80
GROUP      default
PERMISSION default  ALLOWS tcp    80    80    FROM  CIDR  0.0.0.0/0

```

We can then view the ports which are open with:

```

napier@ubuntu:~$ ec2-describe-group default
GROUP 103269750866 default default group
PERMISSION 103269750866 default ALLOWS tcp 0 65535 FROM USER
103269750866 GRPNAME default
PERMISSION 103269750866 default ALLOWS udp 0 65535 FROM USER
103269750866 GRPNAME default
PERMISSION 103269750866 default ALLOWS tcp 22 22 FROM CIDR
0.0.0.0/0
PERMISSION 103269750866 default ALLOWS tcp 80 80 FROM CIDR
0.0.0.0/0
PERMISSION 103269750866 default ALLOWS tcp 3389 3389 FROM CIDR
0.0.0.0/0

```

If we want to get rid of an open port we can use:

```

napier@ubuntu:~$ ec2-revoke default -p 80
GROUP      default
PERMISSION default  ALLOWS tcp    80    80    FROM  CIDR  0.0.0.0/0
napier@ubuntu:~$ ec2-describe default
ec2-describe: command not found

```

```

napier@ubuntu:~$ ec2-describe-group default
GROUP 103269750866 default default group
PERMISSION 103269750866 default ALLOWS tcp 0 65535 FROM USER
103269750866 GRPNAME default
PERMISSION 103269750866 default ALLOWS udp 0 65535 FROM USER
103269750866 GRPNAME default
PERMISSION 103269750866 default ALLOWS tcp 22 22 FROM CIDR
0.0.0.0/0
PERMISSION 103269750866 default ALLOWS tcp 3389 3389 FROM CIDR
0.0.0.0/0

```

The range of ec2 command are:

```

napier@ubuntu:~$ ls /usr/bin/ec2*
/usr/bin/ec2-add-group
/usr/bin/ec2addgrp
/usr/bin/ec2addkey
/usr/bin/ec2-add-keypair
/usr/bin/ec2addsnap
/usr/bin/ec2addvol
/usr/bin/ec2allocaddr
/usr/bin/ec2-allocate-address
/usr/bin/ec2assocaddr
/usr/bin/ec2-associate-address
/usr/bin/ec2-attach-volume
/usr/bin/ec2attvol
/usr/bin/ec2auth
/usr/bin/ec2-authorize
/usr/bin/ec2bundle
/usr/bin/ec2-bundle-instance
/usr/bin/ec2-cancel-bundle-task
/usr/bin/ec2cbun
/usr/bin/ec2-cmd
/usr/bin/ec2-confirm-product-instance
/usr/bin/ec2cpi
/usr/bin/ec2-create-snapshot
/usr/bin/ec2-create-volume
/usr/bin/ec2daddr
/usr/bin/ec2datt
/usr/bin/ec2daz
/usr/bin/ec2dbun
/usr/bin/ec2-delete-group
/usr/bin/ec2-delete-keypair
/usr/bin/ec2-delete-snapshot
/usr/bin/ec2-delete-volume
/usr/bin/ec2delgrp
/usr/bin/ec2delkey
/usr/bin/ec2delsnap
/usr/bin/ec2delvol
/usr/bin/ec2dereg
/usr/bin/ec2-deregister
/usr/bin/ec2-describe-addresses
/usr/bin/ec2-describe-availability-zones
/usr/bin/ec2-describe-bundle-tasks
/usr/bin/ec2-describe-group
/usr/bin/ec2-describe-image-attribute
/usr/bin/ec2-describe-images
/usr/bin/ec2-describe-instances
/usr/bin/ec2-describe-keypairs
/usr/bin/ec2-describe-regions
/usr/bin/ec2-describe-reserved-instances
/usr/bin/ec2-describe-reserved-instances-offerings
/usr/bin/ec2-describe-snapshots
/usr/bin/ec2-describe-volumes
/usr/bin/ec2-detach-volume
/usr/bin/ec2detvol
/usr/bin/ec2dgrp
/usr/bin/ec2dim
/usr/bin/ec2din
/usr/bin/ec2disaddr
/usr/bin/ec2-disassociate-address
/usr/bin/ec2dkey
/usr/bin/ec2dre
/usr/bin/ec2dri

```

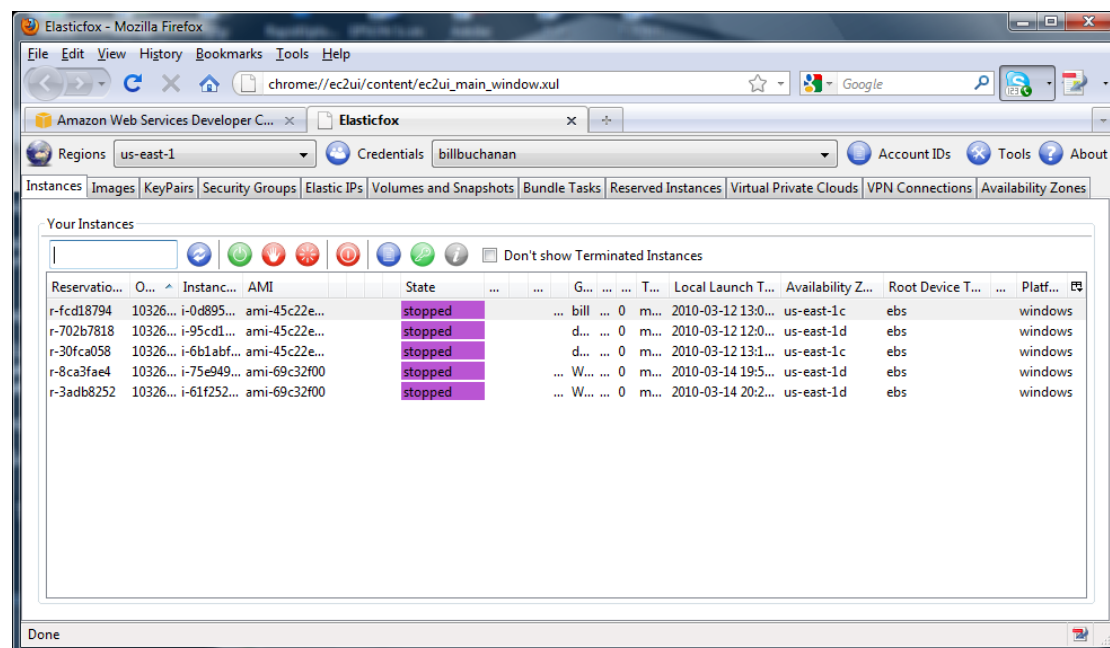
```

/usr/bin/ec2drio
/usr/bin/ec2dsnap
/usr/bin/ec2dvol
/usr/bin/ec2-fingerprint-key
/usr/bin/ec2fp
/usr/bin/ec2gcons
/usr/bin/ec2-get-console-output
/usr/bin/ec2-get-password
/usr/bin/ec2gpass
/usr/bin/ec2kill
/usr/bin/ec2matt
/usr/bin/ec2-migrate-image
/usr/bin/ec2mim
/usr/bin/ec2min
/usr/bin/ec2-modify-image-attribute
/usr/bin/ec2-monitor-instances
/usr/bin/ec2prio
/usr/bin/ec2-purchase-reserved-instances-offering
/usr/bin/ec2ratt
/usr/bin/ec2reboot
/usr/bin/ec2-reboot-instances
/usr/bin/ec2reg
/usr/bin/ec2-register
/usr/bin/ec2reladdr
/usr/bin/ec2-release-address
/usr/bin/ec2-reset-image-attribute
/usr/bin/ec2revoke
/usr/bin/ec2-revoke
/usr/bin/ec2run
/usr/bin/ec2-run-instances
/usr/bin/ec2-terminate-instances
/usr/bin/ec2umin
/usr/bin/ec2-unmonitor-instances
/usr/bin/ec2ver
/usr/bin/ec2-version

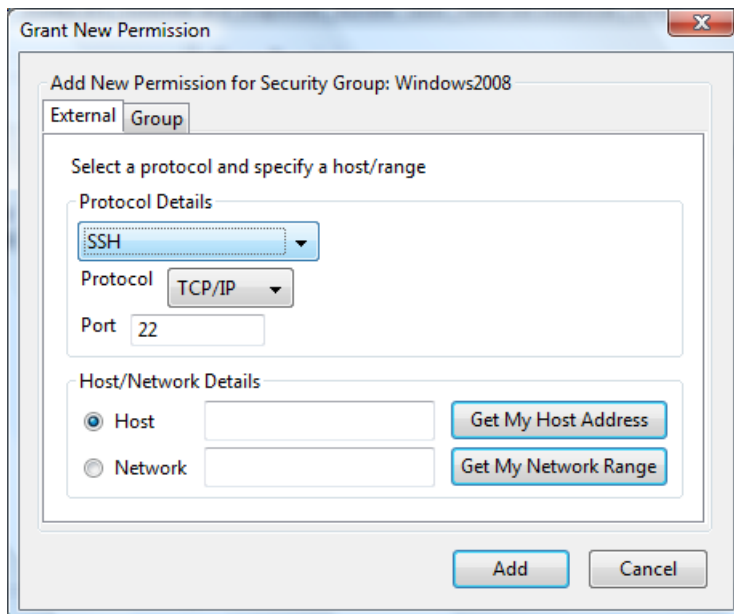
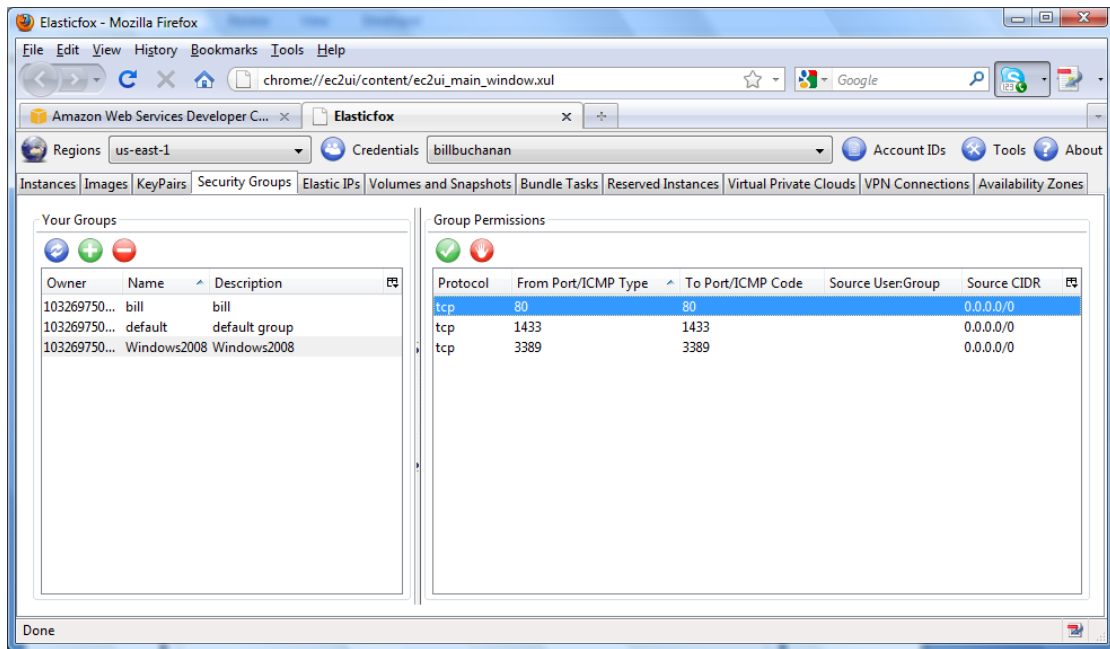
```

ElasticFox

ElasticFox is a plug-in for Firefox, and can be used to access the EC2 cloud. The following shows an example of the interface, where the instances are shown:



and is particularly useful in defining the security groups, such as:



6.7 Tutorial

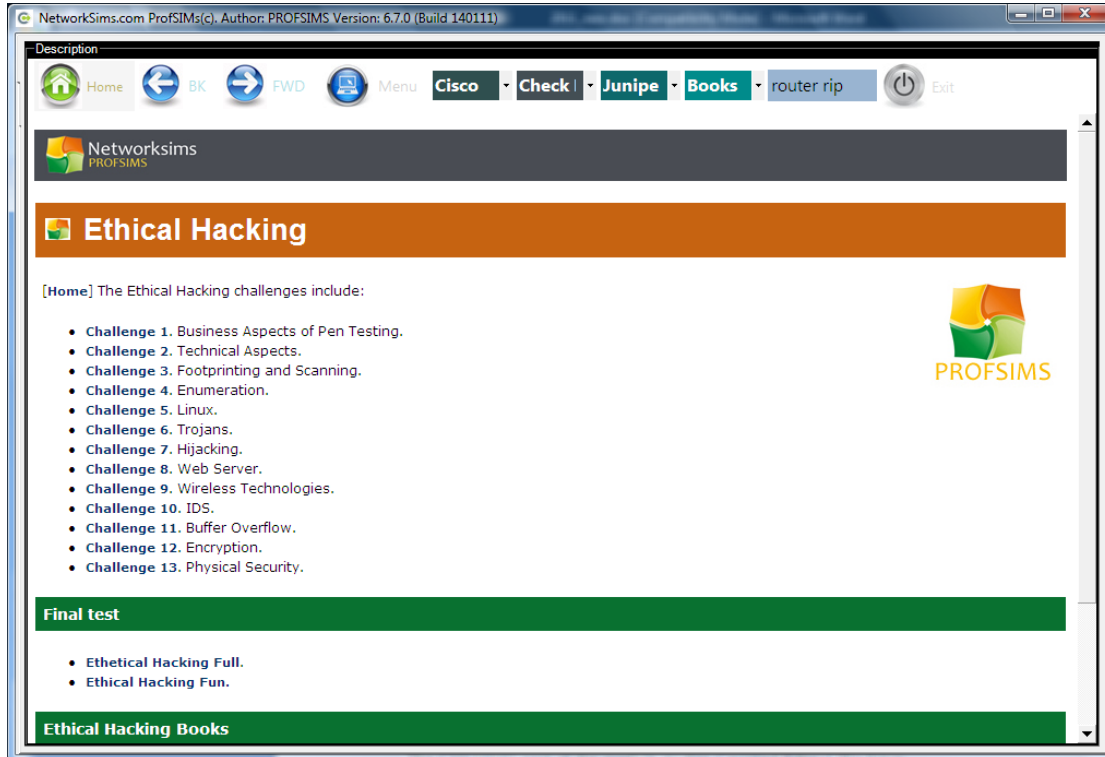
An outline tutorial is at:



On-line tutorial: <http://buchananweb.co.uk/adv06.html>

7 Professional Certification

The CSN10102 part of the module studies **Certified Ethical Hacking**:



8 References

[1] Introduction to Security and Network Forensics, William J Buchanan, Auerbach Publications, ISBN: 084933568X.