Tools for penetration tests

Carlo U. Nicola, HT FHNW
With extracts from documents of: Google; Wireshark; nmap; Nessus.
What is a penetration test?

Goals:

1. Analysis of an IT-environment and search for exploitable vulnerabilities
2. "Simulation" of the behaviour of an attacker
3. The effort spent is an indication of the effort an attacker would have to spend to achieve the same results (i.e. to uncover vulnerabilities)

Phases of a penetration test:

(1) Foot-printing: collecting relevant information about the target environment
(2) Scanning: examine the target networks and hosts in more detail
(3) Analysis: identify the systems that will be analyzed in detail during the following phase
(4) Exploitation: demonstrate proof-of-concept or "real" exploits
(5) Reporting: Prepare a written report and an oral presentation, including concrete recommendations
Foot-printing

Foot-printing means collecting the profile of the target openly accessible in Internet.

What information are we interested in?

A not exhaustive list:

1. Domain names,
2. Contact persons,
3. Interesting pairs (hostnames, IP addresses) connected with the company’s name,
4. IP address blocks,
5. Internal system configurations
Foot-printing: Domain names (1) + (2)

The following tools are most useful in this phase:

1. Google (simple type the company domain name and analyse all additional domains linked to it.)

2. `whois` as command on *nixes or as service via links: `whois.iana.org` (query the TLD (Top Level Domain) field i.e. `ch` for Switzerland. For `.li` and `.ch` TLD `http://www.nic.ch` gives the most authoritative answers.

3. Browsing the company page. Use the search function within the company page with the contact info you extracted from the `whois` search.
An versatile test tool: Google
Google in the early foot-print phase

Below are some questions that are best answered using the operators Google internally uses for its searches:

1. We want to know email addresses and important documents used in the company’s network;
2. We want to check whether important information no longer available is still around perhaps cached somewhere else by Google;
3. We want to map the less obvious connections within the system's domain we are testing;
4. We want to probe which other domains link to the system under investigation;
5. …
# The Google main operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Purpose</th>
<th>Mixes with other operators?</th>
<th>Can be used alone?</th>
<th>Does search work in?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Web</td>
</tr>
<tr>
<td>intitle</td>
<td>Search page title</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>allintitle</td>
<td>Search page title</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>inurl</td>
<td>Search URL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>allinurl</td>
<td>Search URL</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>filetype</td>
<td>Search specific files</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>allintext</td>
<td>Search text of page only</td>
<td>not really</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>site</td>
<td>Search specific site</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>/link</td>
<td>Search for links to pages</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>inanchor</td>
<td>Search link anchor text</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>numrange</td>
<td>Locate number</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>daterange</td>
<td>Search in date range</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>author</td>
<td>Group author search</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>group</td>
<td>Group name search</td>
<td>not really</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>insubject</td>
<td>Group subject search</td>
<td>yes</td>
<td>yes</td>
<td>like intitle</td>
</tr>
<tr>
<td>msgid</td>
<td>Group msgid search</td>
<td>no</td>
<td>yes</td>
<td>not really</td>
</tr>
</tbody>
</table>
# Email addresses

<table>
<thead>
<tr>
<th>Query</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Internal Server Error&quot; &quot;server 403&quot;</td>
<td>Apache server error could reveal admin e-mail address</td>
</tr>
<tr>
<td>intitle:&quot;Execution of this script not permitted&quot;</td>
<td>Cgiwrap script can reveal lots of information, including e-mail addresses and even phone numbers</td>
</tr>
<tr>
<td>e-mail address filetype:csv csv</td>
<td>CSV files that could contain e-mail addresses</td>
</tr>
<tr>
<td>intitle:index.of dead.letter</td>
<td>dead.letter UNIX file contains the contents of unfinished e-mails that can contain sensitive information</td>
</tr>
<tr>
<td>inurl:fcgi-bin/echo</td>
<td>fastcgi echo script can reveal lots of information, including e-mail addresses and server information</td>
</tr>
<tr>
<td>filetype:pst pst -from -to -date</td>
<td>Finds Outlook PST files, which can contain e-mails, calendaring, and address information</td>
</tr>
<tr>
<td>intitle:index.of inbox</td>
<td>Generic “inbox” search can locate e-mail caches</td>
</tr>
<tr>
<td>intitle:&quot;Index Of&quot; -inurl:maillog maillog size</td>
<td>Maillog files can reveal usernames, e-mail addresses, user login/logout times, IP addresses, directories on the server, and more</td>
</tr>
<tr>
<td>inurl:email filetype:mdb</td>
<td>Microsoft Access databases that could contain e-mail information</td>
</tr>
<tr>
<td>filetype:xls inurl:&quot;email.xls&quot;</td>
<td>Microsoft Excel spreadsheets containing e-mail addresses</td>
</tr>
<tr>
<td>filetype:xls username password email</td>
<td>Microsoft Excel spreadsheets containing the words username, password, and email</td>
</tr>
<tr>
<td>intitle:index.of inbox dbx</td>
<td>Outlook Express cleanup.log file can contain locations of e-mail information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filetype:eml eml +intext:&quot;Subject&quot; +intext:&quot;From&quot;</td>
<td>Outlook express e-mail files contain e-mails with full headers</td>
</tr>
<tr>
<td>intitle:index.of inbox dbx</td>
<td>Outlook Express e-mail folder</td>
</tr>
<tr>
<td>filetype:wab wab</td>
<td>Outlook Mail address books contain sensitive e-mail information</td>
</tr>
<tr>
<td>filetype:pst inurl:&quot;outlook.pst&quot;</td>
<td>Outlook PST files can contain e-mails, calendaring, and address information</td>
</tr>
<tr>
<td>filetype:mbx mbx intext:Subject</td>
<td>Outlook versions 1–4 or Eudora mailbox files contain sensitive e-mail information</td>
</tr>
<tr>
<td>inurl:cgi-bin/printenv</td>
<td>Printenv script can reveal lots of information, including e-mail addresses and server information</td>
</tr>
<tr>
<td>inurl:forward filetype:forward -cvs</td>
<td>UNIX user e-mail forward files can list e-mail addresses</td>
</tr>
<tr>
<td>Various generic e-mail files</td>
<td>Various generic e-mail files</td>
</tr>
<tr>
<td>&quot;Most Submitted Forms and Scripts&quot; “this section”</td>
<td>WebTrends statistics pages reveal directory information, client access statistics, e-mail addresses, and more</td>
</tr>
<tr>
<td>filetype:reg +intext:&quot;internet account manager&quot;</td>
<td>Windows registry files can reveal information such as usernames, POP3 passwords, e-mail addresses, and more</td>
</tr>
<tr>
<td>“This summary was generated by wwwstat”</td>
<td>Ww wwstat statistics information can reveal directory info, client access statistics, e-mail addresses, and more</td>
</tr>
</tbody>
</table>
Information in cache

If the desired information is not available you can always try to get a cached version. Take the URL and add &strip=1 as the example below shows:

Original:

Cached text only:
Rough network mapping with Google

How is the domain www.fhnw.ch structured?

We want of course to filter out the most obvious combinations, for example all those containing www.

site:fhnw.ch -site:www.fhnw.ch

Or more explicit:

site:fhnw.ch -inurl:www.fhnw.ch
External links

How is the domain www.fhnw.ch linked with other domains?

**link:**www.fhnw.ch

But if you type:

**link:**www.fhnw.ch  -site:fhnw.ch

**link** is treated like a normal search test.
Vulnerabilities: Google Dorks (1)

<Dork> filetype:php inurl:index.php inurl:"module=subjects" inurl:"func=*" (listpages| viewpage | listcat)
</Dork>

<Category>Advisories and Vulnerabilities</Category>
getQuery> filetype:php inurl:index.php inurl:"module=subjects" inurl:"func=*" (listpages| viewpage | listcat)
</Query>

<Comment> Reportedly the PostNuke Modules Factory Subjects module is affected by a remote SQL injection vulnerability.  
    http://securityfocus.com/bid/11148/discussion/
</Comment>
Vulnerabilities: Google Dorks (2)

<Dork> "Online Store - Powered by ProductCart"</Dork>
<Category>Advisories and Vulnerabilities</Category>
<Query>"Online Store - Powered by ProductCart"</Query>
<Comment> ProductCart is "an ASP shopping cart that combines sophisticated ecommerce features with time-saving store management tools and remarkable ease of use. It is widely used by many e-commerce sites". Multiple SQL injection vulnerabilities have been found in the product, they allow anything from gaining administrative privileges (bypassing the authentication mechanism), to executing arbitrary code.
http://www.securityfocus.com/bid/8105
(search SF for more)
</Comment>
Google: SQL vulnerability

Query: "executeQuery(".".getParameter("
Google: DB SQL vulnerability

Query:
"Microsoft OLE DB Provider for ODBC Drivers error '80040e14'" filetype:asp
Foot-printing: IP ranges

The question which IP numbers are available to the company is best answered by searching the database of RIR (Regional Internet Register). In Europe we query the link: http://www.ripe.net and for North America: http://www.arin.net.

For the query one needs only a single correct IP number pertinent to the company. A simple way to get it is via: nslookup fhnw.ch:

```
C:\Documents and Settings\ulisse>nslookup fhnw.ch
...
Non-authoritative answer:
Name: fhnw.ch
Address: 147.86.3.160
```
Foot-printing: DNS query

Name server are responsible for the mapping of DN in IP addresses. A DNS query via dig tells more than that: it reveals names and IP addresses of important parts of the company’s infrastructure (i.e. the of the mail servers)

ulisse@beaver:~$ dig fhnw.ch
; <<>> DiG 9.7.0-P1 <<>> fhnw.ch
...
;; QUESTION SECTION:
;fhnw.ch. IN A
;; ANSWER SECTION:
fhnw.ch. 28 IN A 147.86.3.160
...
DNSSEC:N
Name servers:
ns.inwx.de
ns1.fhnw.ch [147.86.3.20]
nss2.fhnw.ch [147.86.3.21]

Name server are responsible for the mapping of DN in IP addresses. A DNS query via dig tells more than that: it reveals names and IP addresses of important parts of the company’s infrastructure (i.e. the of the mail servers)
Foot-printing: automated DNS query

If we know the IP range for the company, we can easily do a reverse mapping in order to get the DN of the machines with a Perl script:

```perl
#! usr/bin/perl
use socket;
$b_net = "147.86.";
for ($i=0; $i<255; $i++) {
    for ($j=0; $j<255; $j++) {
        $ipaddr = "$b_net.$i.$j";
        $name = gethostbyaddr($ipaddr, AF_INET);
        if ($name) {
            print "${ip}	${name}
";
        }
    }
}
```

TCP or UDP proto
Scanning

Goal of the scanning phase:

1. Determine the network structure. This includes the search of all machines that are reachable from inside or outside the company.
2. Analyse the hosts in detail
3. Identify the OS and versions used
4. Determine the services running in the hosts and the corresponding software products. This can give hints at possible vulnerabilities present on a target system
5. Perform vulnerability scans.
**Scanning: Network structure**

traceroute is the tool of choice to analyse the network structure. Its main output is a list of all **hops** (IP addresses) till the target system.

There are different traceroute variants that either use UDP or ICMP packets:

On *ix systems, UDP is usually the default and ICMP can be used with the `-I` option. Its often a good idea to use both options, especially if one is not successful.

traceroute has many more options, the most important of them are:

a)  `-n` Print hop addresses numerically rather than symbolically and numerically. This saves a NS address-to-name lookup for each gateway found on the path.

b)  `-d` Set the SO_DEBUG socket option.

c)  `-F` Set the "don't fragment" bit.
**Scanning: traceroute (1)**

Below the output of: `traceroute -q 1 www.fhnw.ch`

```
ulisse@beaver:~$ traceroute -q 1 www.fhnw.ch
traceroute to www.fhnw.ch (147.86.3.160), 30 hops max, 60 byte packets
  1 192.168.1.1 (192.168.1.1)  1.952 ms
  2 zhhia00p-ads115.bluewin.ch (85.3.128.1)  13.308 ms
  3 net1701.zhhia00p-rtdi02.bluewin.ch (213.3.247.190)  16.333 ms
  4 net1701.zhhhdz09p-rtdi02.bluewin.ch (213.3.247.189)  18.048 ms
  5 198-0-186-195.bluewin.ch (195.186.0.198)  20.157 ms
  6 i79zhb-025-bun1.bb.ip-plus.net (138.187.129.113)  26.333 ms
  7 i79tix-025-ten1-1.bb.ip-plus.net (138.187.129.82)  24.471 ms
  8 swit-00-ser0.ce.ip-plus.net (164.128.22.130)  26.083 ms
  9 swiEZ2-10GE-1-3.switch.ch (130.59.36.249)  28.200 ms
10  swiBA2-10GE-1-4.switch.ch (130.59.37.106)  31.662 ms
11  unibi7-te-1-2.urz.p.unibas.ch (192.43.192.197)  30.541 ms
12  192.43.192.222 (192.43.192.222)  32.488 ms
13  * * * till hop 30
```

Router unib

Firewall

NS HS11 21
Scanning: traceroute to mail server

Output of: traceroute -n -q 1 mxmuu11.fhnw.ch

traceroute to mxnu11.fhnw.ch (147.86.3.24), 30 hops max, 60 byte packets
1 192.168.1.1  4.601 ms  4.462 ms  4.336 ms
2 85.3.128.1  12.362 ms  14.110 ms  16.254 ms
3 213.3.247.190  14.538 ms  15.304 ms  16.790 ms
4 213.3.247.189  33.385 ms  33.317 ms  33.226 ms
5 195.186.0.198  33.103 ms  33.006 ms  32.944 ms
6 138.187.129.113  32.771 ms  41.614 ms  40.523 ms
7 138.187.129.82  40.179 ms  36.939 ms  35.468 ms
8 164.128.22.130  29.013 ms  31.980 ms  34.119 ms
9 130.59.36.249  33.521 ms  18.498 ms  20.060 ms
10 130.59.37.106  24.019 ms  26.989 ms  27.469 ms
11 192.43.192.197  40.458 ms  40.365 ms  30.570 ms
12 192.43.192.222  30.451 ms  20.893 ms  20.714 ms
13 * * *
14 * * * until hop 30
Scanning: traceroute from internal host

Output of: traceroute  -n -q 1 www.fhnw.ch

ulisse@beaver:~$ traceroute  -q 1 www.fhnw.ch
traceroute to www.fhnw.ch (147.86.3.160), 30 hops max, 60 byte packets
  1  10.212.136.1 (10.212.136.1)  1.287 ms
  2  nd41u101-sta-vl3213.net.fhnw.ch (10.212.16.33)  3.039 ms
  3  nc40u101-sta-vl3113.net.fhnw.ch (10.212.16.17)  1.516 ms
  4  nca0e001-sta-vl3113.net.fhnw.ch (10.218.0.17)  2.147 ms
  5  nda0e001-sta-vl3113.net.fhnw.ch (10.218.0.18)  1.986 ms
  6  *
  7  *
Scanning: traceroute from int. host to external

Output of: traceroute -q 1 www.ethz.ch

ulisse@beaver:~$ traceroute -q 1 www.ethz.ch
traceroute to www.ethz.ch (129.132.19.220), 30 hops max, 60 byte packets
1  10.212.136.1 (10.212.136.1)  6.655 ms
2  nda0e001-sta-vl3113.net.fhnw.ch (10.212.16.17)  2.153 ms
3  nca0e001-sta-vl3113.net.fhnw.ch (10.218.0.17)  2.618 ms
4  nda0e001-sta-vl3113.net.fhnw.ch (10.218.0.18)  2.515 ms
5  nfa0e002-sta.net.fhnw.ch (10.218.0.252)  2.402 ms
    nda0e001-sin-vl4064.net.fhnw.ch (193.73.125.14)  12.274 ms
6  unibi7-vl-501.urz.p.unibas.ch (192.43.192.213)  3.674 ms
7  swiba2.urz.p.unibas.ch (192.43.192.196)  5.537 ms
8  swiez2-10ge-5-4.switch.ch (130.59.37.105)  5.375 ms
9  rou-gw-rz-tengig-to-switch.ethz.ch (192.33.92.1)  5.281 ms
10 rou-fw-rz-rz-gw.ethz.ch (192.33.92.169)  4.453 ms
11 * * till hop 30
**Scanning: hping3 (1)**

hping3 allows to trace a route to a machine using UDP, TCP and ICMP. Thus we can trace the route to using a TCP SYN port 80 probe.

```
ulisse@beaver:~$ hping3 --ttl 1 --traceroute --destport 80 --syn lis.technik.fhnw.ch
HPING lis.technik.fhnw.ch (eth0 147.86.20.21): S set, 40 headers + 0 data bytes
hop=1 TTL 0 during transit from ip=192.168.1.1 name=UNKNOWN
hop=1 hoprtt=0.6 ms
hop=2 TTL 0 during transit from ip=85.3.128.1 name=zhxia00p-adsl15.bluewin.ch
hop=2 hoprtt=13.0 ms
hop=3 TTL 0 during transit from ip=213.3.247.190 name=net1701.zhhsa00p-rtdi02.bluewin.ch
hop=3 hoprtt=12.8 ms
hop=4 TTL 0 during transit from ip=213.3.247.189 name=net1701.zhhdz09p-rtdi02.bluewin.ch
hop=4 hoprtt=12.6 ms
hop=5 TTL 0 during transit from ip=195.186.0.198 name=198-0-186-195.bluewin.ch
hop=5 hoprtt=12.7 ms
hop=6 TTL 0 during transit from ip=138.187.129.113 name=i79zhb-025-bun1.bb.ip-plus.net
hop=6 hoprtt=15.1 ms
```
Scanning: hping3 (2)

hping3 option in the example:

1. --ttl Start with ttl set to 1 s
2. --traceroute Increment ttl for every subsequent attempt
3. --destport Set the destination port
4. --syn Set the SYN flag in TCP header.

The example shows that another router 138.187.129.113 is responsible for the route to lis.technik.fhnw.ch. Only 6 hops separate the external machine from lis.technik.fhnw.ch.
FHNW network so far

147.86.20.21

10.51.2.32 :DNS
10.212.16.33
10.212.16.17
10.218.0.17
10.218.0.18
10.218.0.252

10.212.136.0/24

138.187.129.113

192.43.192.222

193.73.125.14

Internet

Switch
nmap
nmap: network exploration tool

Purpose: "Nmap ("Network Mapper") is a free and open source utility for network exploration or security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime."

Download: http://www.insecure.org/nmap/

Functionality: (1) Network mapping, (2) port scanning (3) OS detection and ping sweeps (and much more besides …)
**nmap: functionality (1)**

*nmap* uses the following scan techniques:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>UDP scan</td>
</tr>
<tr>
<td>(2)</td>
<td>TCP connect()</td>
</tr>
<tr>
<td>(3)</td>
<td>TCP SYN (half open)</td>
</tr>
<tr>
<td>(4)</td>
<td>ftp proxy (bounce attack)</td>
</tr>
<tr>
<td>(5)</td>
<td>Reverse-Identification</td>
</tr>
<tr>
<td>(6)</td>
<td>ICMP (ping sweep)</td>
</tr>
<tr>
<td>(7)</td>
<td>TCP FIN scan</td>
</tr>
<tr>
<td>(8)</td>
<td>TCP ACK sweep</td>
</tr>
<tr>
<td>(9)</td>
<td>Xmas Tree (FIN, URG, PSH flags set)</td>
</tr>
<tr>
<td>(10)</td>
<td>TCP SYN sweep</td>
</tr>
<tr>
<td>(11)</td>
<td>IP Protocol</td>
</tr>
<tr>
<td>(12)</td>
<td>Null Scan (FIN, URG, PSH, RST, ACK,SYN flags not set)</td>
</tr>
</tbody>
</table>
**nmap: functionality (2)**

*nmap* uses the following OS detection techniques:

→ TCP/IP fingerprinting
→ stealth scanning
→ dynamic delay and retransmission calculations
→ parallel scanning (-Pn)
→ detection of down hosts via parallel pings
→ decoy scanning
→ port filtering detection
→ direct (non-port mapper) RPC scanning
→ fragmentation scanning
→ flexible target and port specification.
**nmap: Labor Bedingungen**

1. Die nmap Scan-Rate auf ca 10/s reduzieren;

2. Ports Scan nur an maximal 3 FHNW-Server durchführen;

3. Bevor Sie zu scannen beginnen, schreiben Sie ein Email an network.services@fhnw.ch mit folgenden Inhalt:
   a) Aktuelle IP-Nummer Ihres Laptops;
   b) IP-Nummer Bereich, den Sie scannen wollen;
   c) Zeitfenster des Scans (ungefähr von ... bis ... )
Scanning: network’s mapping

Goal # 1: Network Mapping

Why: To determine the topology of the network.

How:

1. Manually using tools like ping, traceroute, (Windows: tracert)
2. Automatically with tools like Google based TouchGraph network mapping tool
3. Semi-automatically with nmap ≥ 5.5 (Zenmap GUI tool)
Goal # 2: **Port Scanning**

**Why:** To find open ports in order to exploit them.

**How:** With `nmap`.

- **TCP Connect:** attempt to complete the 3-way handshake, look for SYN-ACK. This scan is easy to detect.

- **TCP SYN Scan:** “half-open” scan, look for SYN-ACK, then send RESET, in this case the target system will not record the attempted connection. It is faster than the TCP connect scan.

- **TCP FIN, Xmas Tree, Null Scans:** scans that violate the protocol: the closed ports send RESET, the open ones send nothing (Windows does not respond to these scans).
Scanning: ports open/closed (2)

- **TCP ACK Scan**: may be useful to get past packet filters (believes it is a response to a request from inside a firewall), if one receives a RESET, one knows that this port is open through the firewall.

- **FTP Bounce Scan**: request a server to send a file to a victim machine inside its network (most servers though, have this service disabled).

- **UDP Scan**: if receive ICMP Port Unreachable, it assumes that port closed, otherwise open. (Unreliable).

- **Ping Sweep**: can use ICMP or TCP packets to identify active hosts within the target network.
Scanning: ports open/closed (3)

Additional goals of a network scan:

→ **Decoys**: insert false IP addresses in scan packets.

→ **Ping Sweeps**: identify active hosts on a target network.

→ **Find RPCs**: connect to each open port looking for common RPC services (send a NULL RPC commands).
Scanning: OS detection (1)

Goal # 3: Operating System Detection

Why: To determine which Operating System is in use in order to exploit known vulnerabilities.

Also known as *TCP stack fingerprinting*. It takes advantage of the ambiguity of how to handle illegal combinations of TCP code bits that is found in the RFCs.

Experience teaches that each OS responds to illegal combinations in different ways. Therefore one determines the OS by examining the system's responses.
**Scanning: OS detection (2)**

**How:** With `nmap` you examine how the OS manipulates specific parameters of a TCP-packet.

- **Window Size:** most *ix OS keep the same window size throughout a session. Windows OS tend to change the window size during a session.

- **Time to Live (TTL):** FreeBSD or Linux typically use 64; Windows OS typically uses 128.

- **Do Not Fragment Flag:** most OS leave this flag set, OpenBSD leaves it unset.
Scanning: vulnerability assessment

Goal # 4: Vulnerability Assessment

Why: To determine what known (or unknown?) vulnerabilities exist on a given network

Vulnerabilities come from:

→ Default configuration weakness
→ Configuration errors
→ Security holes in specific versions of applications and protocols
→ Failure to download security patches!
Scanning: vulnerability checkers

Vulnerability checkers consists of:

→ Database of known vulnerabilities
→ Configuration tool
→ Scanning engine
→ Knowledge base of current scan
→ Report generation tool
Vulnerability checker tool: Nessus

Purpose: “A software which will audit remotely a given network and determine whether bad guys (aka 'crackers') may break into it, or misuse it in some way.”

Available platforms: UNIX for client and server
Windows for client only

Available at: http://www.nessus.org/
What does Nessus do?

(1) It iteratively tests if a target system (or systems) is vulnerable to **known** exploits.

(2) It uses a separate plug-in (written in C or in Nessus scripting language) for each security test the user wants to execute.

(3) It can test multiple hosts concurrently.

(4) It produces a thorough vulnerability assessment report at the conclusion of the vulnerability scan.
What does Nessus check for?

→ Backdoors
→ CGI abuses
→ Denial of Service
→ Finger abuses
→ FTP
→ Gain a shell remotely
→ Gain root remotely

→ Port scanners
→ Remote file access
→ RPC
→ SMTP problems
→ Useless services
→ Windows loopholes
→ and more...
Traffic shaping and Intrusion Detection Systems (IDS)
Traffic shaping

Traditional firewall is a binary system:
→ Allow traffic or disallow traffic

Traffic shaping is a more subtle technique:
→ it limits certain kinds of traffic;
→ it can differentiate by host address, by protocol, etc
→ Multi-Protocol Label Switching (MPLS):
  Label traffic flows at the edge of the network and let core routers identify the required class of service

With traffic shaping one can solve a fastidious problem known on every school's campus:
→ P2P file sharing takes a lot of bandwidth
→ On average 1/3 of a university's network bandwidth is consumed by BitTorrent (you know what I mean …)
Academic computer users' patterns

Personal computing activities in hours/week

Percentage of Stanford undergraduates
Traffic shaping functions

1. Classify and analyze traffic:
   Classify by IP address and port number
   Use application-specific information (layer 7)

2. Control traffic:
   Selectively slow certain classes of traffic

3. Monitor network performance:
   Collect performance data, used to improve policies

4. Network resilience:
   Active traffic management can provide resilience to DoS attacks, at least within the enterprise network
PacketShaper classification

Classify more than 400 apps at OSI Layers 2-7

Peer-to-Peer Apps:
- Aimster
- AudioGalaxy
- CuteMX
- DirectConnect
- Gnutella
- Hotline
- iMesh
- KaZaA/Morpheus
- Napster
- ScourExchange
- Tripnosis….

Some Other Apps:
- H.323
- RTP-I/RTCP-I
- PASV FTP
- HTTP
- Real
- WinMedia
- Shoutcast
- MPEG
- Quicktime
- RTSP
- Chatting Apps
- Games
PacketShaper controls

A partition:

→ Creates a virtual pipe within a link for each traffic class;
→ Provides a minimal and maximal bandwidth for each class;
→ So it enables an efficient bandwidth's use.

Rate shaped P2P capped at 300kbps

Rate shaped HTTP/SSL to give better performance
PacketShaper report: HTTP

Outside Web Server Normalized Network Response Times

Inside Web Server Normalized Network Response Times
Host and network intrusion detection

Intrusion prevention:
1. Network firewall:
   → Restrict flow of packets (see firewall slides);
2. System security:
   → Find buffer overflow vulnerabilities and remove them!

Intrusion detection:
1. Discover system modifications:
   → Tripwire
2. Look for attack in progress:
   → Network traffic patterns
   → System calls, other system events
Tripwire

Standard modus operandi of a cracker's attack:
   a) Gain user access to system;
   b) Gain root access;
   c) Replace system binaries to set up backdoor;
   d) Use backdoor for future activities.

Tripwire detects an attack by examining the system's binaries:
   → It computes hash of key system binaries;
   → it compares the actual hash to the hash it stored earlier;
   → It reports a problem if the hash is different;
   → It stores the reference hash codes on a read-only medium.
How to outsmart Tripwire

Cracker's attack with a new twist:
1. Gain access;
2. Install backdoor:
   This can be stored in main memory, not on disk!!
3. Use it.

What can Tripwire do in this case?
→ Not much because this attack doesn’t change the system files stored on the hard disk!
→ Nevertheless using Tripwire is always a good idea.
→ It detects a compromised system's file after the attack has happened.
How to detect modified binary in memory

Can use system-call monitoring techniques. For example (see Wagner, Dean; *IEEE Security and Privacy Conf.* 2001):

→ Build automaton of expected (typical) system calls  
  
    – Can be done automatically from source code.

→ Monitor system calls from each program;

→ Compare with automaton and eventually catch violation.
Code's example and its relevant automaton

f(int x) {
    x ? getuid() : geteuid();
    x++
}
g() {
    fd = open("foo", O_RDONLY);
    f(0); close(fd); f(1);
    exit(0);
}

If code behavior is inconsistent with the automaton then something is wrong.
General intrusion detection

Many intrusion detection systems that are available, are roughly divided in three categories: (i) Network-based, (ii) host-based, or (iii) a combination of (i) and (ii).

Two basic models:

→ **Misuse** detection model:
  - Maintain data on known attacks;
  - Look for activity with corresponding signatures.

→ **Anomaly** detection model:
  - Try to figure out what is “normal” (hard) and then
  - Report anomalous behavior

Fundamental problem: too many **false** alarms.

http://www.snort.org/
Example of misuse that leads to detection: rootkit

A typical rootkit sniffs networks for passwords:
- It is a collection of programs that allow attacker to install and operate a packet sniffer (on Unix machines).

rootkit attack:
- Use stolen password or dictionary attack to get access as a legitimate user;
- Get root access using vulnerabilities in rdist, sendmail, /bin/mail, loadmodule, rpc.ypupdated (NIS Network Information Service data base), lpr, or passwd.
- Via ftp the rootkit is uploaded to the host, unpacked, compiled, and installed.
- It then collects more username/password pairs and then moves on.
Rootkit covers its tracks

Modifies netstat, ps, ls, du, ifconfig, login
  – Modified binaries hide the new files used by rootkit
  – Modified login allows attacker to return for fishing new passwords

Rootkit fools simple Tripwire checksum
  – Modified binaries have the same checksum as the correct one.
  – But a better hash than MD5 would make a rootkit attack more difficult (e.g. SHA-1024).
Detecting rootkit on system

Not the best way:
- Disk is full of sniffer logs.

Manual confirmation:
- Reinstall a clean version of ps and see what processes are running.

Automatic detection:
- rootkit does not alter the data structures normally used by netstat, ps, ls, du, ifconfig only their output is faked;
- Thus a host-based intrusion detection can find rootkit files, as long as an update version of rootkit does not disable your intrusion detection system ...
Detecting network attack (Sept. 2003)

Symantec *honeypot* running Red Hat Linux 9.

Attack
- Samba ‘call_trans2open’ Remote Buffer Overflow (BID 7294)
- Attacker installed a copy of the SHV4 Rootkit

Snort NIDS generated alerts against this attack from its standard rule signature:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 \  
msg:"NETBIOS SMB trans2open buffer overflow attempt"; \  
flow:to_server,established; \  
content:"|00|"; offset:0; depth:1; \  
content:"|ff|SMB|32|"; offset:4; depth:5; \  
content:"|0014|"; offset:60; depth:2; \  
...
```

More info: Symantec030929-Analysis-SHV4Rootkit.pdf on the course's webpage.
Misuse's example: port sweep

Attacks can be OS specific:
- Bugs in specific OS implementations can be exploited to mount an attack;
- Oversights in default configuration's files open a path that can be easily exploited.

Attacker sweeps network to find vulnerabilities:
- Port sweep tries many ports on many IP addresses
- If the characteristic behavior is detected, then it mounts an attack
  - SGI IRIX responds to TCPMUX port (TCP on port 1)
  - If a machine responds, then SGI IRIX vulnerabilities can be tested and used to break in

Port sweep activity is easily detected.
Anomaly Detection

Basic idea:
→ Monitor network traffic, system calls;
→ Compute statistical properties;
→ Report errors if statistics lies outside an empirical established range.

Example: IDES (Denning, SRI)
→ For each user, store the daily count of certain activities
  • E.g., Fraction of hours spent reading email.
→ Maintain a list of counts for several days;
→ Report anomaly if count is outside weighted norm.

The crux is that the most unpredictable user is the most dangerous.
Anomaly: sys calls' sequences

Build traces during normal run of program:

- Example of program's (good) behavior (sys calls), open read write open mmap write fchmod close
- Sample traces are stored in file (as 4-calls sequences):
  open read write open
  read write open mmap
  write open mmap write
  open mmap write fchmod
  mmap write fchmod close
- Report anomaly if for example the following sequence is observed:
  open read read open mmap write fchmod close

Compute # of mismatches to get mismatch rate.

(See papers of Hofmeyr, Somayaji, Forrest)
Difficulties in intrusion detection

Lack of training data:
→ Lots of “normal” network and system call data but too little data containing realistic attacks, anomalies

Data drift:
→ Statistical methods are used to detect changes in behavior
→ That means that the cracker can attack gradually and incrementally thus defeating a statistical analysis.

Main characteristics not well understood:
→ By many measures, attack may be within bounds of “normal” range of activities

False positive are very costly:
→ System administrators spend many hours examining bogus evidence.
Example: strategic intrusion assessment (Lunt 1999)

Test over two-week period:

→ AFIWC’s (US Air Force Information Warfare Center) intrusion detectors at 100 AFBs alarmed on 2 million sessions
→ Manual review identified 12,000 suspicious events
→ Further manual review ⇒ four actual incidents

Conclusion:

→ Most alarms are false positives;
→ Most true positives are trivial incidents;
→ Of the significant incidents, most are isolated attacks to be dealt with locally.

Appendices
Appendix A: How to attach a IDS in a LAN

Drawing by
Jeff Nathan <jeff@wwti.com>
Brian Caswell <bmc@snort.org>
Appendix B: Packet analysis (1)

**Goal:** Capture and decode the header and body information used in different Internet protocols.

**Why:** The careful study of the packet flow in a network permits (i) to understand whether all components work as expected and (ii) to detect anomalies in the traffic due to an attack on the system.

**How:**

- Wireshark for both Unices and Windows: [www.wireshark.org](http://www.wireshark.org)
- On Linux: tcdump
Appendix B: Packet analysis (2)

The packet list pane displays a summary of each packet captured.

The packet details pane displays the packet selected in the list pane with more details.

The packet bytes pane displays the ASCII data from the selection in the first pane.
Appendix B: Packet analysis (3)