VIRUSES AND MALWARE

Ben Livshits, Microsoft Research

Overview of Today's Lecture

- Viruses
- Virus/antivirus coevolution paper discussed
- Intrusion detection
  - Behavioral detection
  - Firewalls
  - Application firewalls
- Advanced attack techniques
  - Heap spraying
  - Heap feng shui
  - JIT spraying

What is a Virus?

- a program that can infect other programs by modifying them to include a, possibly evolved, version of itself

Fred Cohen, 1983

Computer Viruses

Theory and Experiments

Fred Cohen
Dept. of Computer Science and Electric Engineering, Lehigh University, Bethlehem, PA 18215, USA, and The Foundation for Computer Integrity Research, Pittsburgh, PA 15217, USA

This paper introduces "computer viruses" and examines their potential for causing widespread damage to computer systems. Basic theoretical results are presented, and the infeasibility of viral defense in large classes of systems is shown. Defensive schemes are presented and several experiments are described.

Keywords: Computer Viruses, System Integrity, Data Integrity

Fred Cohen received a B.S. in Electrical Engineering from Carnegie-Mellon University in 1977, an MS in Information Science from the University of Pittsburgh in 1981 and a Ph.D. in Electrical Engineering from the University of Southern California in 1986.
He has worked as a freelance consultant since 1977, and has designed and implemented numerous devices and systems. He is currently a professor of Computer Science and Electrical Engineering at Lehigh University, Chairman and Director of Engineering at the Foundation for Computer Integrity Research, and President of Legal Software Incorporated.
He is a member of the ACM, IEEE, and IACR. His current research interests include computer viruses, information flow model, adaptive systems theory, genetic models of computing, and evolutionary systems.

North-Holland
Computers & Security 6 (1987) 22–35

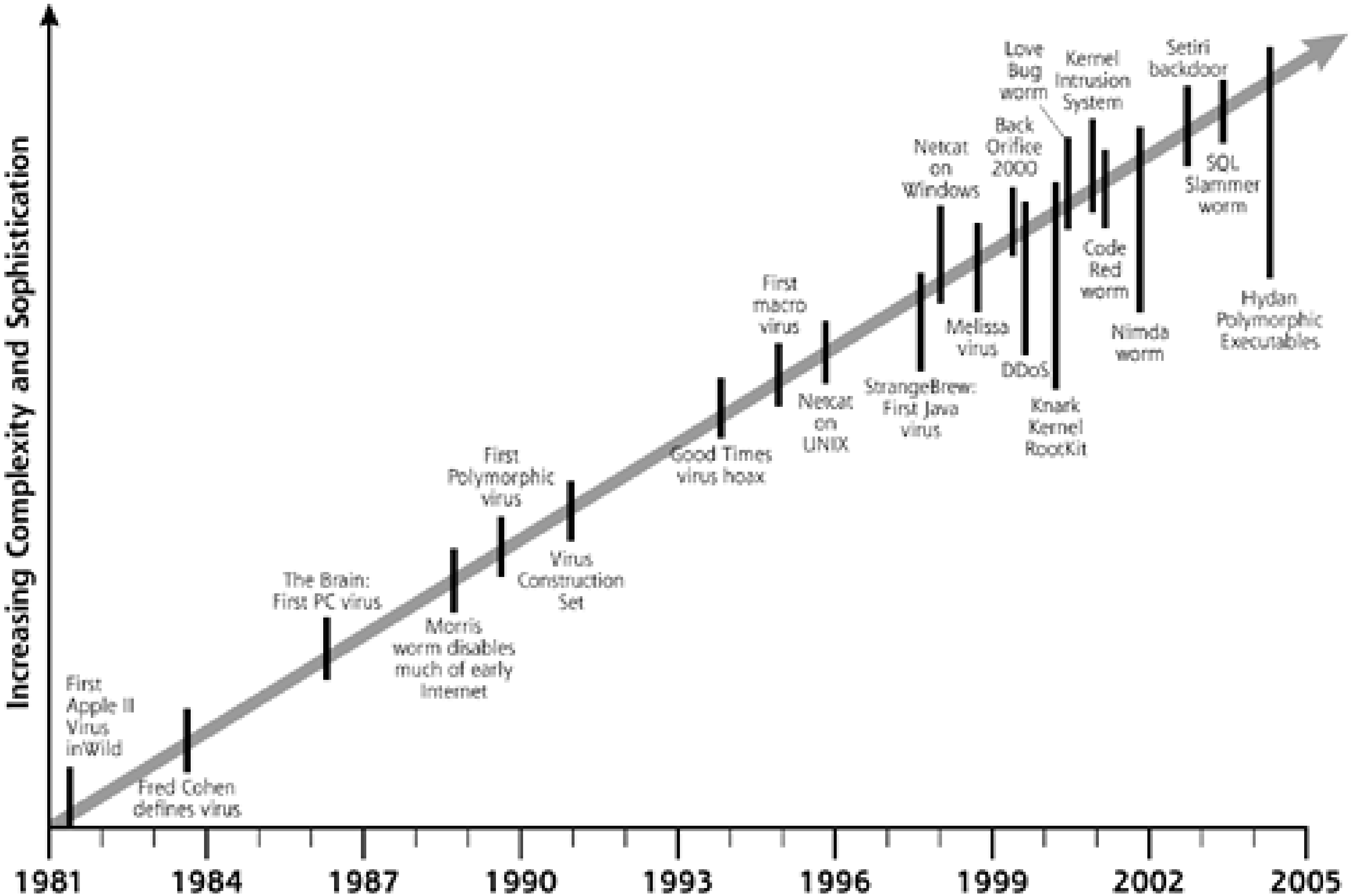
1. Introduction

This paper defines a major computer security problem called a virus. The virus is interesting because of its ability to attach itself to other programs and cause them to become viruses as well. Given the widespread use of sharing in current computer systems, the threat of a virus carrying a Trojan horse [1,20] is significant. Although a considerable amount of work has been done in implementing policies to protect against the illicit dissemination of information [4,7], and many systems have been implemented to provide protection from this sort of attack [12,19,21,22], little work has been done in the area of keeping information entering an area from causing damage [5,18]. There are many types of information paths possible in systems, some legitimate and authorized, and others that may be covert [18], the most commonly ignored one being through the user. We will ignore covert information paths throughout this paper.

The general facilities exist for providing provably correct protection schemes [9], but they depend on a security policy that is effective against the types of attacks being carried out. Even some quite simple protection systems cannot be proven 'safe' [14]. Protection from denial of services requires the detection of halting programs which is well known to be undecidable [11]. The problem of precisely marking information flow within a system [10] has been shown to be NP-complete. The use of guards for the passing of untrustworthy information [25] between users has been examined, but in general depends on the ability to prove program correctness which is well known to be NP-complete.

The Xerox worm program [23] has demonstrated the ability to propagate through a network, and has even accidentally caused denial of services. In a later variation, the game of 'core wars' [8] was invented to allow two programs to do battle with one another. Other variations on this theme have been reported by many unpublished authors, mostly in the context of nighttime games played between programmers. The term virus has also been used in conjunction with an augmentation to

Malware Timeline



Coevolution: Basic Setup

Virus

- Wait for user to execute an infected file
- Infect other (binary) files
- Spread that way

Antivirus

- Identify a sequence of instructions or data
- Formulate a signature
- Scan all files
- Look for signature found verbatim
- Bottleneck: scanning speed

Coevolution: Entry Point Scanning

Virus

- Place virus at the entry point or make it directly reachable from the entry point
- Make virus small to avoid being easily noticed by user

Antivirus

- Entry point scanning
- Do exploration of reachable instruction starting with the entry point of the program
- Continue until no more instructions are found

Coevolution: Virus Encryption

Virus

- Decryption routine
- Virus body
- Decrypt into memory, not do disk
- Set PC to the beginning of the decryption buffer
- Encrypt with a different key before adding virus to new executable

Antivirus

- Decryption (and encryption) routines (packers) used by viruses are easy to fingerprint
- Develop signatures to match these routines
- Attempt to decrypt the virus body to perform a secondary verification (x-raying)

Coevolution: Polymorphic

Virus

- Use a mutation engine to generate a (decryption routine, encryption routine) pair
- Functionally similar or the same, but syntactically very different
- Use the encryption routine to encode the body of the virus
- No fixed part of the virus preserved (decryption, encryption, body)

Antivirus

- Custom detection program designed to recognize specific detection engines
- Generic decryption (GD)
  - Emulator
  - Signature matching engine
  - Scan memory/disk at regular intervals in hopes of finding decoded virus body

GD Challenges

- How long to emulate the execution? Viruses use padding instructions to delay execution. Can also use sleep for a while to slow down the scanner.
- What is the quality of the emulator? How many CPUs to support?
- What if decryption starts upon user interactions? How do we trigger it? What about anti-emulation tricks?

False Positives in Virus Detection

- A "false positive" is when antivirus software identifies a non-malicious file as a virus. When this happens, it can cause serious problems.
- For example, if an antivirus program is configured to immediately delete or quarantine infected files, a false positive in an essential file can render the operating system or some applications unusable.

- In May 2007, a faulty virus signature issued by Symantec mistakenly removed essential operating system files, leaving thousands of PCs unable to boot
- Also in May 2007, the executable file required by Pegasus Mail was falsely detected by Norton AntiVirus as being a Trojan and it was automatically removed, preventing Pegasus Mail from running. Norton anti-virus had falsely identified three releases of Pegasus Mail as malware, and would delete the Pegasus Mail installer file when that happened n response to this Pegasus Mail stated:
- On the basis that Norton/Symantec has done this for every one of the last three releases of Pegasus Mail, we can only condemn this product as too flawed to use, and recommend in the strongest terms that our users cease using it in favor of alternative, less buggy anti-virus packages
- In April 2010, McAfee VirusScan detected svchost.exe, a normal Windows binary, as a virus on machines running Windows XP with Service Pack 3, causing a reboot loop and loss of all network access
- In December 2010, a faulty update on the AVG anti-virus suite damaged 64-bit versions of Windows 7, rendering it unable to boot, due to an endless boot loop created
- In October 2011, Microsoft Security Essentials removed the Google Chrome browser, rival to Microsoft's own Internet Explorer. MSE flagged Chrome as a Zbot banking trojan

Top 20 Malware on Internet/user Computer

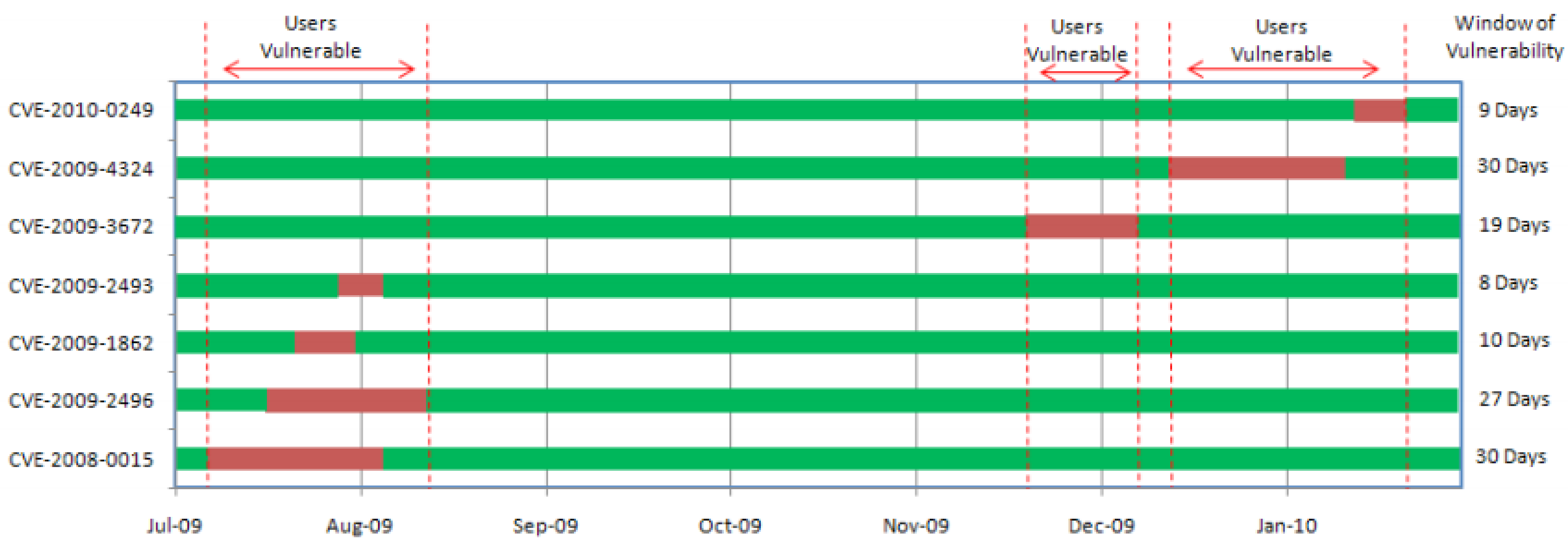
Current rank	Delta	Verdict
1	4	AdWare.Win32.FunWeb.gq
2	New	Hoax.Win32.ArchSMS.pxm
3	3	AdWare.Win32.HotBar.dh
4	8	Trojan.HTML.Iframe.dl
5	New	Hoax.HTML.OdKlas.a
6	New	Trojan.JS.Popupper.aw
7	1	Exploit.JS.Pdfka.ddt
8	-8	Trojan.JS.Agent.btv
9	-9	Trojan-Downloader.JS.Agent.fun
10	-10	Trojan-Downloader.Java.OpenStream.bi
11	-7	Exploit.HTML.CVE-2010-1885.ad
12	New	Trojan.JS.Agent.uo
13	New	Trojan-Downloader.JS.Iframe.cdh
14	New	Packed.Win32.Katusha.o
15	New	Exploit.Java.CVE-2010-0840.d
16	1	Trojan.JS.Agent.bhr
17	New	Trojan-Clicker.JS.Agent.om
18	New	Trojan.JS.Fraud.bl
19	New	Exploit.Java.CVE-2010-0840.c
20	New	Trojan-Clicker.HTML.Iframe.aky

Current rank	Delta	Verdict
1	0	Net-Worm.Win32.Kido.ir
2	0	Virus.Win32.Sality.aa
3	1	Net-Worm.Win32.Kido.ih
4	New	Hoax.Win32.ArchSMS.pxm
5	0	Virus.Win32.Sality.bh
6	-3	HackTool.Win32.Kiser.zv
7	-1	Hoax.Win32.Screensaver.b
8	-1	AdWare.Win32.HotBar.dh
9	8	Trojan.Win32.Starter.yy
10	1	Packed.Win32.Katusha.o
11	1	Worm.Win32.FlyStudio.cu
12	-2	HackTool.Win32.Kiser.il
13	-4	Trojan.JS.Agent.bhr
14	2	Trojan-Downloader.Win32.Geral.cnh
15	New	Porn-Tool.Win32.StripDance.d
16	New	Exploit.JS.Agent.bbk
17	New	Trojan.Win32.AutoRun.azq
18	-5	Trojan-Downloader.Win32.VB.eql
19	-5	Worm.Win32.Mabezat.b
20	-5	Packed.Win32.Klone.bq

http://www.securelist.com/en/analysis/204792170/Monthly_Malware_Statistics_March_2011

Vulnerability Gap

- As long as user has the right virus signatures *and* computer has recently been scanner, detection will likely work
- But the virus landscape changes fast
- This calls for monitoring techniques for unknown viruses



http://www.m86security.com/documents/pdfs/security_labs/m86_security_labs_vulnerability_report.pdf

CVE-2009-4324: December 2009

Adobe Reader/Acrobat "Doc.media.newPlayer()" Memory Corruption

Secunia Stay Secure

Secunia Advisory:	SA37690
Release Date:	2009-12-15
Last Update:	2009-12-16
Popularity:	6,490 views
Critical:	Extremely critical
Impact:	System access
Where:	From remote
Solution Status:	Vendor Workaround
Software:	Adobe Acrobat 3D 8.x Adobe Acrobat 8 Professional Adobe Acrobat 8.x Adobe Acrobat 9.x Adobe Reader 8.x Adobe Reader 9.x

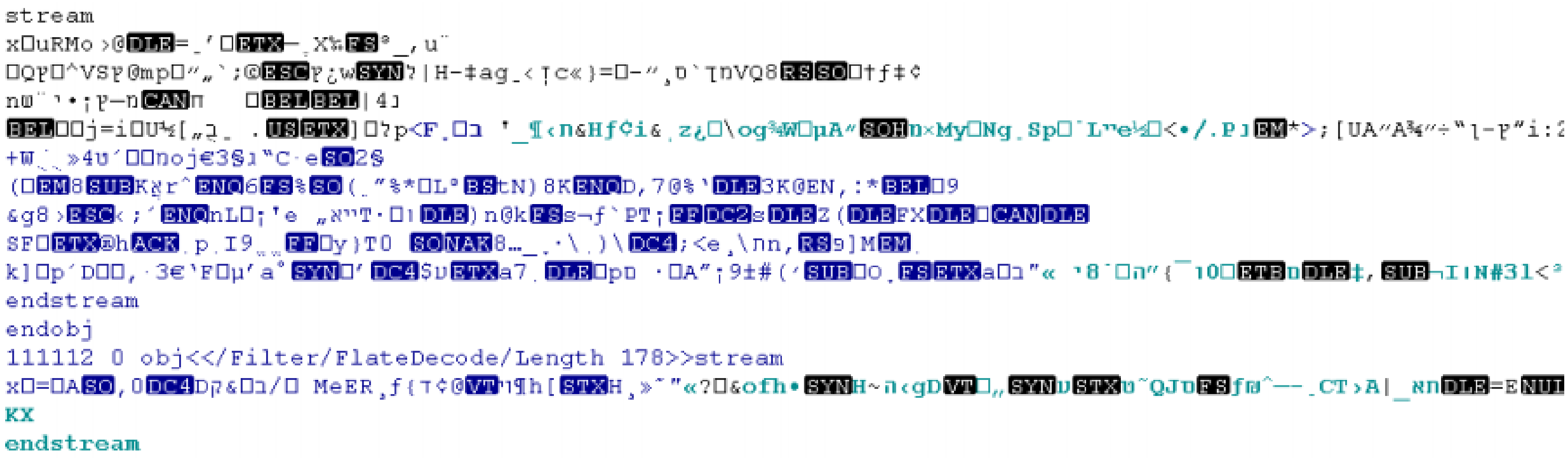
Description:
A vulnerability has been reported in Adobe Reader and Acrobat, which can be exploited by malicious people to compromise a user's system.

The vulnerability is caused due to an unspecified error in the implementation of the "Doc.media.newPlayer()" JavaScript method. This can be exploited to corrupt memory and execute arbitrary code via a specially crafted PDF file.

NOTE: This vulnerability is currently being actively exploited.

http://www.m86security.com/documents/pdfs/security_labs/m86_security_labs_vulnerability_report.pdf

Exploit in the PDF Unfolding…

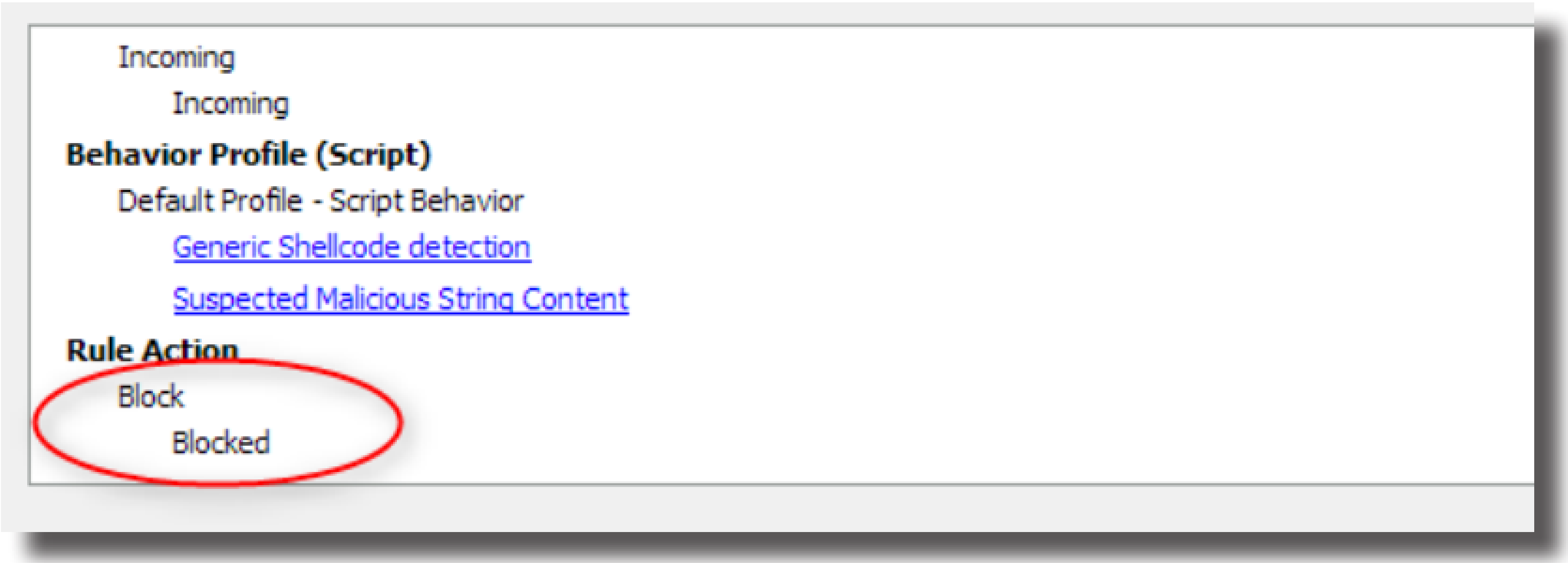



```
ylerati2=new Array();
var fzfpa8 = 'ARG9090ARG9090'.replace(/ARG/g,'%u');
var imkujn2 = 'Z54EBZ758BZ8B3CZ3574ZX378Z56F5Z768BZX32XZ33F5Z49C9ZAD41ZDB33ZXF36Z14BEZ3828Z74E
fzfpa8=unescape(fzfpa8);
imkujn2=unescape(imkujn2);endstream
endobj
111112 0 obj<</Filter/FlateDecode/Length 178>>stream
while(fzfpa8.length <= 0x8000){fzfpa8+=fzfpa8;}
fzfpa8=fzfpa8.substr(0,0x8000 - imkujn2.length);
for(gofmeq=0;gofmeq<xsbrgm;gofmeq++) {ylerati2[gofmeq]=fzfpa8 + imkujn2;}
if(xsbrgm){dwdsf1();dwdsf1();try {this.media.newPlayer(null);} catch(e) {}dwdsf1();}endstream
endobj
trailer<</Root 1 0 R /Size 11>>
```

http://www.m86security.com/documents/pdfs/security_labs/m86_security_labs_vulnerability_report.pdf

Automatic Zero-Day Blocking

- Scanning engine recognizes the `newPlayer()` vulnerability (checked in red).
- Because this is a zero-day vulnerability, the `newPlayer()` vulnerability would be considered unknown
- Subsequently, the M86 Secure Web Gateway falls back to its behavioral analysis capability.
- Below, the behavior of the JavaScript is suspicious; therefore it is blocked by this default rule, requiring no updates



http://www.m86security.com/documents/pdfs/security_labs/m86_security_labs_vulnerability_report.pdf

Proactive Detection Techniques

- heuristic analyzer
- policy-based security
- intrusion detection/prevention systems
- etc.

http://www.securelist.com/en/downloads/vlpdfs/wp_nikishin_proactive_en.pdf

Heuristic Analyzers

- A heuristic analyzer looks at
  - code of executable files
  - Macros
  - Scripts
  - memory or boot sectors

  to detect malicious programs that cannot be identified using the usual (signature-based) methods

- Heuristic analyzers search for unknown malicious software

- Detection rates are usually low: 20-30% at most

http://www.m86security.com/documents/pdfs/security_labs/m86_security_labs_vulnerability_report.pdf

Policy-based Security

- Use an overall security policy to restrict certain types of actions on the machine
- For instance
  - Don’t open email attachments
  - Don’t open files from the internet whose reputation is unknown
  - Only allow access to a whitelist of web sites
  - Disallow software installation
- The Cisco-Microsoft approach
  - Scan computers of users connecting to the network
  - Limit network access from machines that are not found to be fully compliant (i.e. virus definitions are out of date)
  - Force access to an update server
  - “Shepherd” the user into compliance

Behavioral Monitoring Techniques

	Cisco	McAfee	Panda	Symantec	Trend Micro	BitDefender	Kaspersky
Heuristic Analyzer		•	•	•	•	•	•
IPS		•	•	•			•
Buffer Overrun		•					
Policy based					•		
Alerting system				•	•		•
Behaviour Blocker	•		•			•	•

IDS: Intrusion Detection Systems

- What it is
  - Security guards and “beware of dog” signs are forms of IDS
  - Serve two purposes:
    - Detect something bad was happening
    - deter the perpetrator
- Components
  - Collect signals
  - Process and create alerts
  - Notify system operators

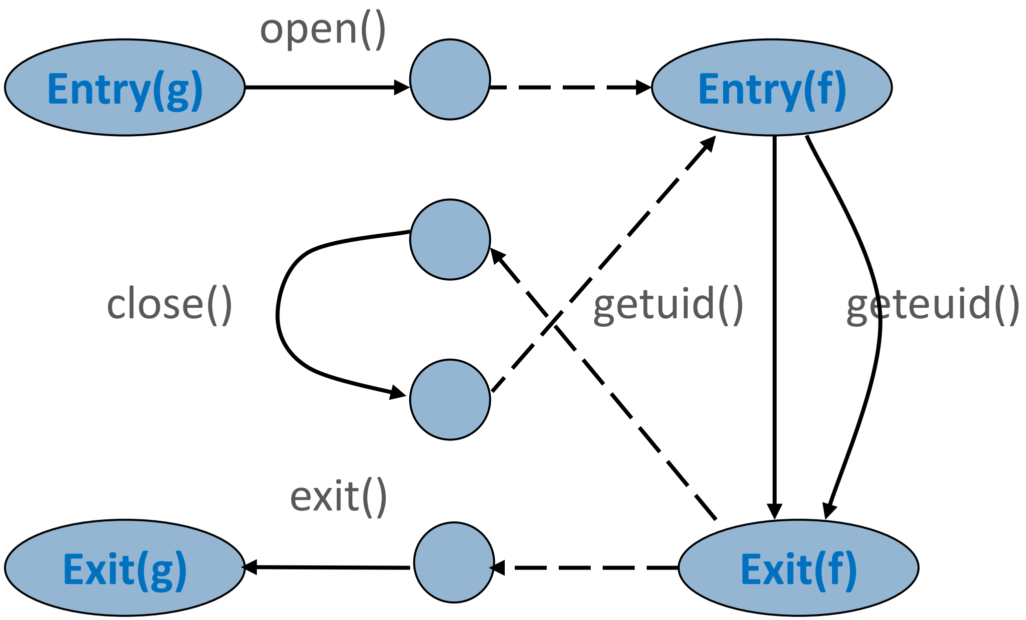
Host-Based vs. Network-Based IDS

- Log analyzers
- Signature-based sensors
- System call analyzers
- Application behavior analyzers
- File integrity checkers

- Scan incoming and outgoing traffic
- Primarily signature-based
- Combined into firewalls
- Can be located on a different machine

Host-Based Intrusion Detection

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

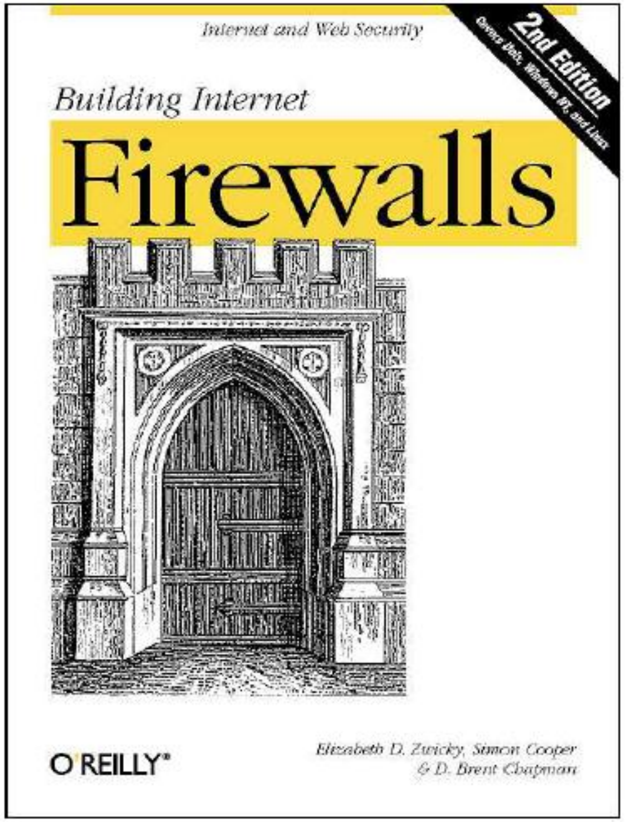


If the observed code behavior is inconsistent with the statically inferred model, something is wrong

Question of the Day

How do you minimize false positives in an intrusion detection system?

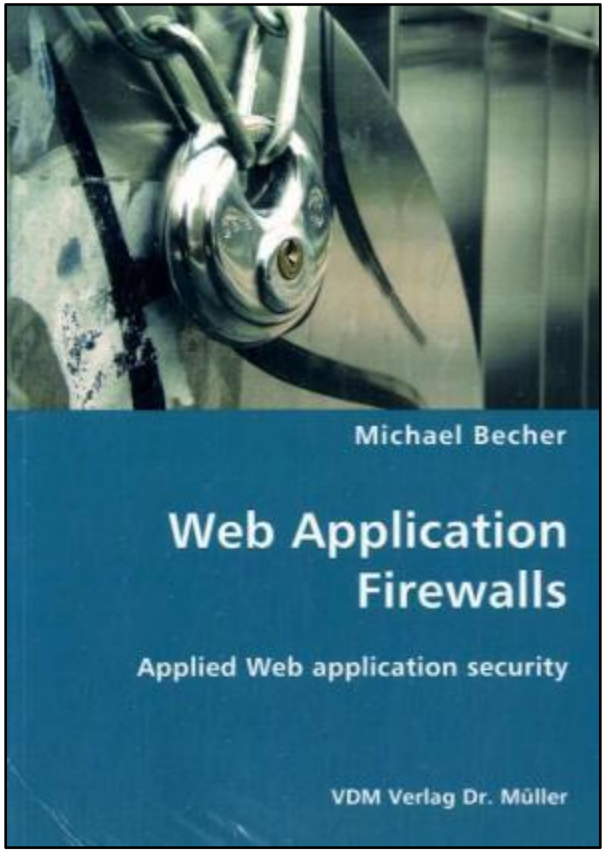
Firewalls: Network and App-level



Elizabeth D. Zwicky

Simon Cooper

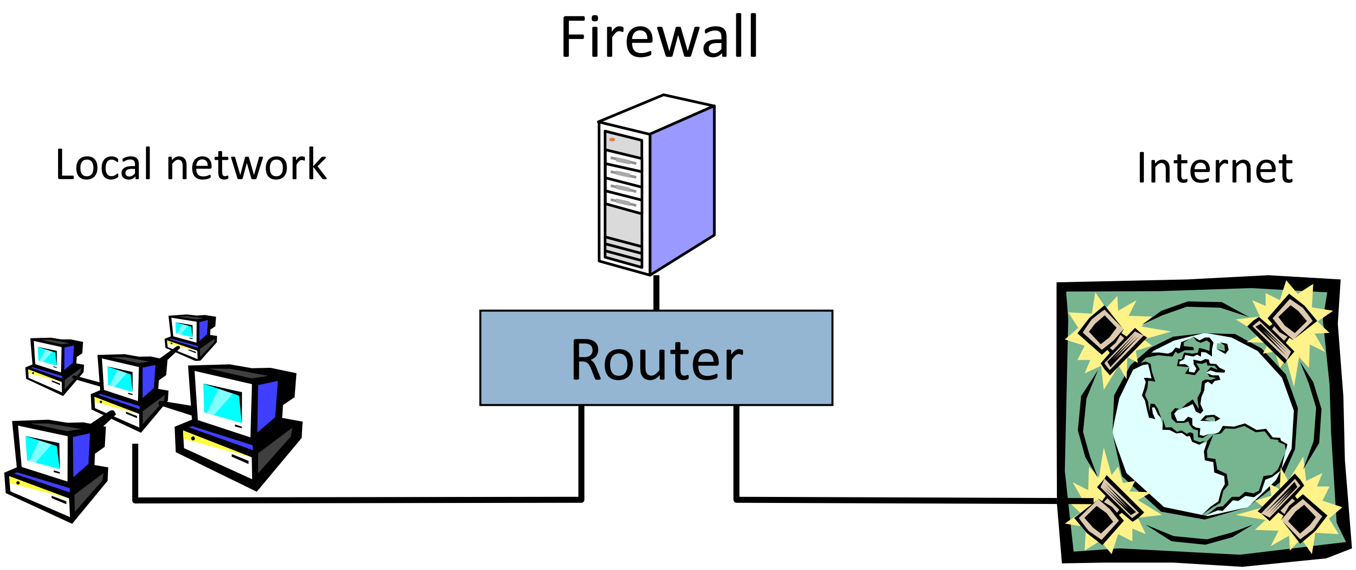
D. Brent Chapman



Michael Becher

Basic Firewall Concept

- Separate local area net from internet

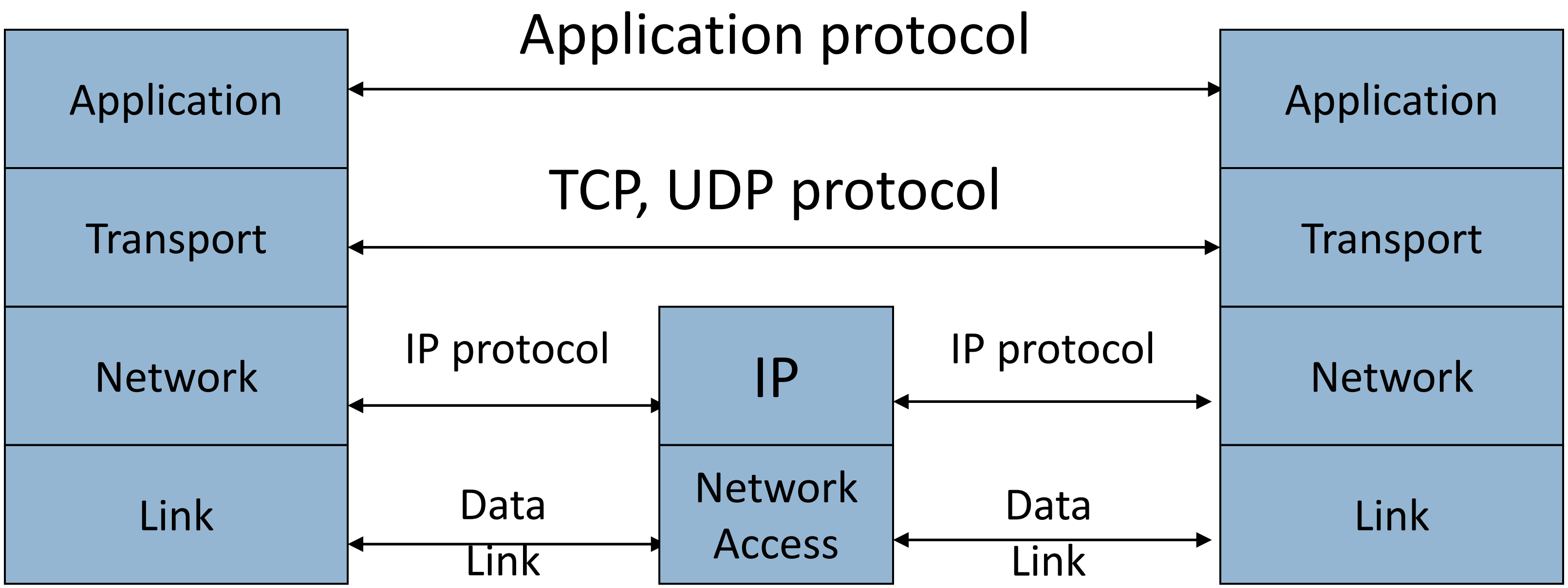


All packets between LAN and internet routed through firewall

Firewall Goals

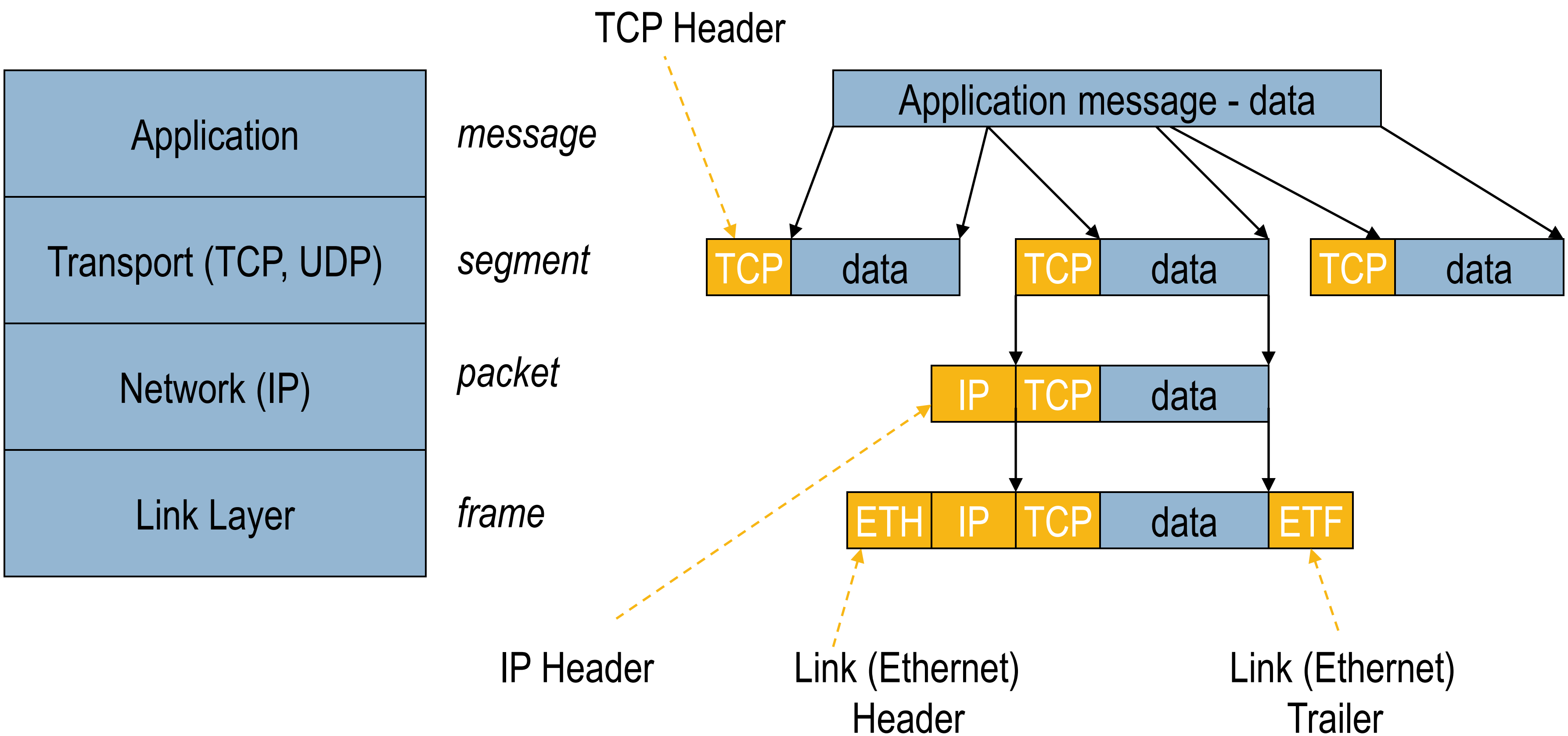
- Prevent malicious attacks on hosts
  - Port sweeps, ICMP echo to broadcast addr, syn flooding, …
  - Worm propagation
- Prevent general disruption of internal network
- Monitor and control quality of service (QoS)
- Provide defense in depth
  - Programs contain bugs and are vulnerable to attack
  - Network protocols may contain;
    - Design weaknesses (SSH CRC)
    - Implementation flaws (SSL, NTP, FTP, SMTP...)
- Control traffic between “zones of trusts”
  - Can control traffic between separate local networks, etc.

Review: TCP Protocol Stack

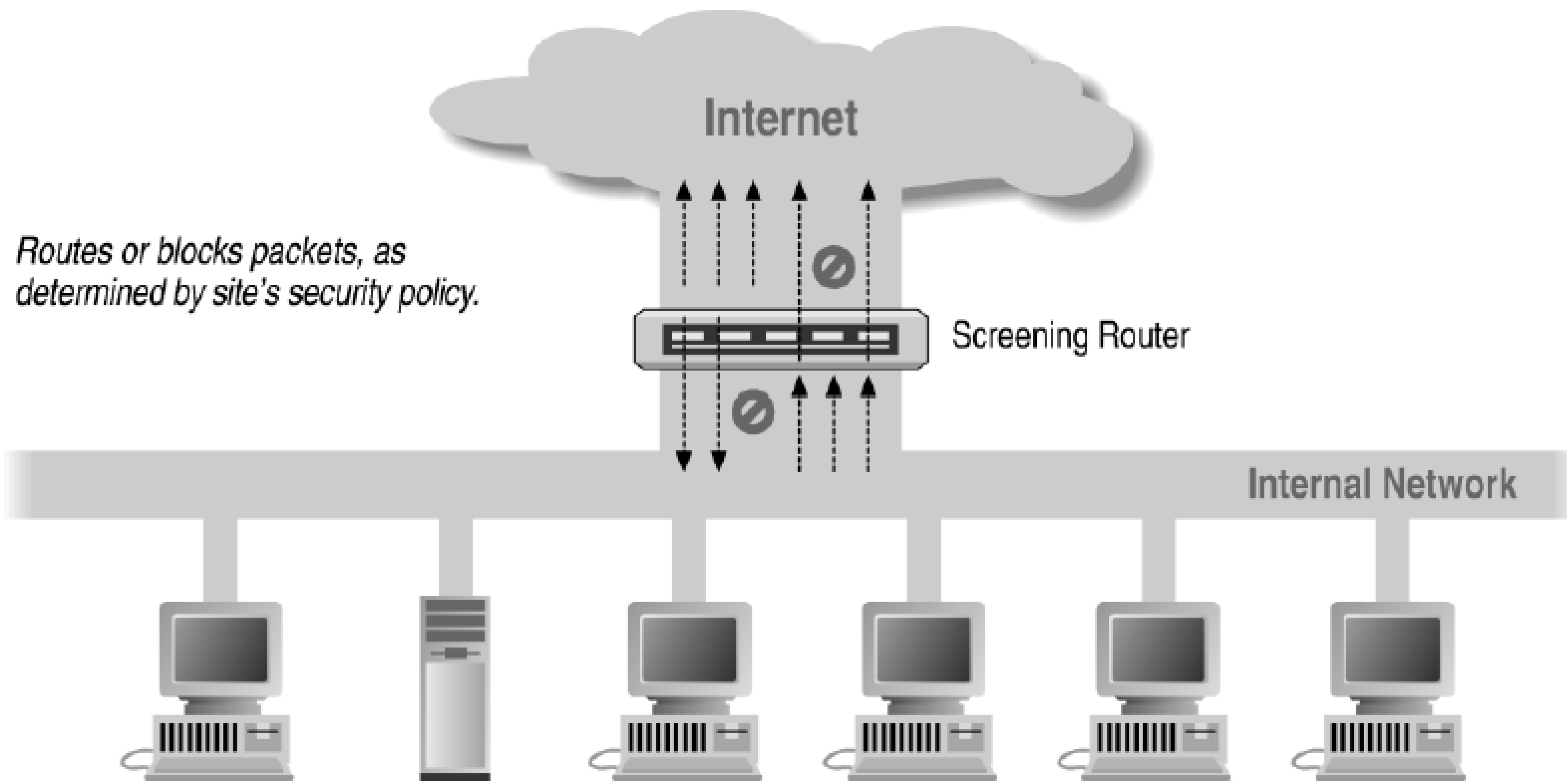


Transport layer provides *ports*, logical channels identified by number

Review: Data Formats



Screening Router for Packet Filtering



Illustrations: Simon Cooper

Packet Filtering

- Uses transport-layer information only
  - IP Source Address, Destination Address
  - Protocol (TCP, UDP, ICMP, etc)
  - TCP or UDP source & destination ports
  - TCP Flags (SYN, ACK, FIN, RST, PSH, etc)
  - ICMP message type
- Examples
  - DNS uses port 53
    - Block incoming port 53 packets except known trusted servers
- Issues
  - Stateful filtering
  - Encapsulation: address translation, other complications
  - Fragmentation

Firewall Configuration (Incoming)

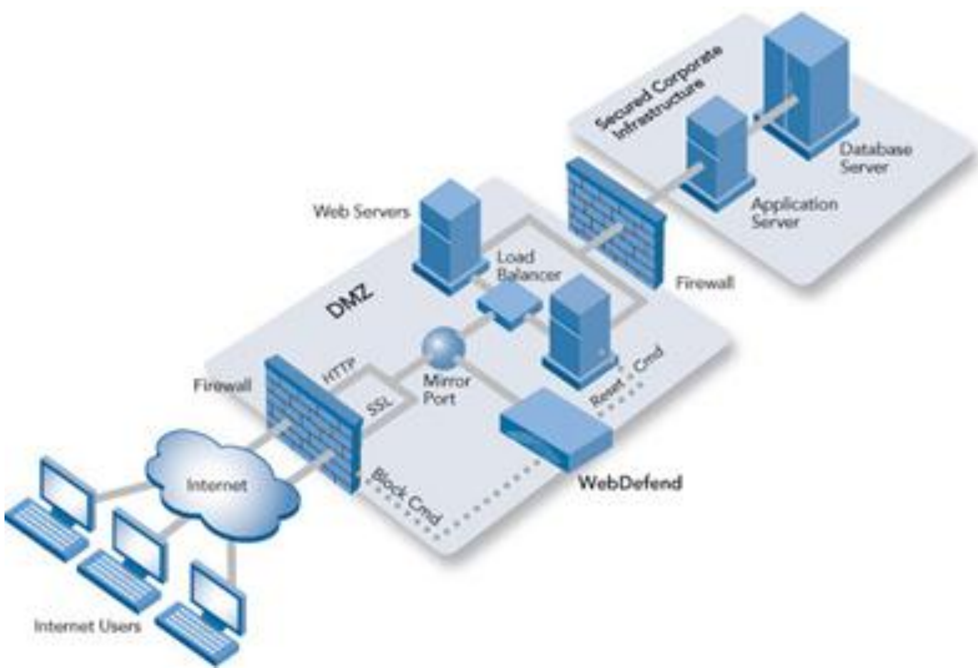
Inbound Rules

Name	Group	Profile	Enabled	Action	Override	Program	Local Address	Remote Address	Protocol	Local Port	Remote Port	Allowed Users	Allowed Computers
Allow authenticated IPsec bypass (Vista a...		All	Yes	Secure ...	Yes	Any	Any	Any	Any	Any	Any	Any	REDMOND\GP-ICF ...
Bonjour Service		Domain	Yes	Allow	No	C:\Progr...	Any	Any	UDP	Any	Any	Any	Any
Bonjour Service		Domain	Yes	Allow	No	C:\Progr...	Any	Any	TCP	Any	Any	Any	Any
Bonjour Service		Domain	Yes	Allow	No	C:\Progr...	Any	Any	UDP	Any	Any	Any	Any
Bonjour Service		Domain	Yes	Allow	No	C:\Progr...	Any	Any	TCP	Any	Any	Any	Any
Client Notification Channel		Private	Yes	Allow	No	Any	Any	Any	UDP	1745	Any	Any	Any
Client Notification Channel		Domain	Yes	Allow	No	Any	Any	Any	UDP	1745	Any	Any	Any
CorpNet: ISATAP - Allow		All	Yes	Allow	No	Any	Any	Any	IPv6	Any	Any	Any	Any
CORPNET: PNRP Allow		Private	Yes	Allow	No	Any	fe80::/10	fe80::/10	UDP	3540	Any	Any	Any
CORPNET: PNRP Secure		All	Yes	Secure	No	Any	Any	Any	UDP	3540	Any	Any	Any
CorpNet: WTT TCP Client - Allow		All	Yes	Allow	No	%WTTBI...	Any	Any	TCP	1778	Any	Any	Any
Daemonu.exe		Private	No	Allow	No	C:\Progr...	Any	Any	TCP	Any	Any	Any	Any
Daemonu.exe		Private	No	Allow	No	C:\Progr...	Any	Any	UDP	Any	Any	Any	Any
Internet Explorer		Domain	Yes	Allow	No	C:\progr...	Any	Any	UDP	Any	Any	Any	Any
Internet Explorer		Domain	Yes	Allow	No	C:\progr...	Any	Any	TCP	Any	Any	Any	Any
iTunes		All	Yes	Allow	No	C:\Progr...	Any	Any	Any	Any	Any	Any	Any
Microsoft Lync 2010		All	Yes	Allow	No	C:\Progr...	Any	Any	Any	Any	Any	Any	Any
Microsoft Office Live Meeting 2007		Domain	Yes	Allow	No	C:\Progr...	Any	Any	UDP	Any	Any	Any	Any
Microsoft Office Live Meeting 2007		Domain	Yes	Allow	No	C:\Progr...	Any	Any	TCP	Any	Any	Any	Any
Microsoft Office Live Meeting 2007		Private	Yes	Allow	No	C:\Progr...	Any	Any	UDP	Any	Any	Any	Any
Microsoft Office Live Meeting 2007		Private	Yes	Allow	No	C:\Progr...	Any	Any	TCP	Any	Any	Any	Any
Microsoft Office Outlook		Private	Yes	Allow	No	C:\Progr...	Any	Any	UDP	6004	Any	Any	Any
Microsoft Office Outlook		All	Yes	Allow	No	%Progra...	Any	Any	UDP	6004	Any	Any	Any
Microsoft OneNote		Private	Yes	Allow	No	C:\Progr...	Any	Any	TCP	Any	Any	Any	Any
Microsoft OneNote		Private	Yes	Allow	No	C:\Progr...	Any	Any	UDP	Any	Any	Any	Any
Microsoft SharePoint Workspace		Private	Yes	Allow	No	C:\Progr...	Any	Any	TCP	Any	Any	Any	Any
Microsoft SharePoint Workspace		Private	Yes	Allow	No	C:\Progr...	Any	Any	UDP	Any	Any	Any	Any
MSIT DA - ICMPv4 Echo Request		All	Yes	Allow	No	Any	Any	Any	ICMPv4	Any	Any	Any	Any
MSIT DA - ICMPv6 Echo Request		All	Yes	Allow	No	Any	Any	Any	ICMPv6	Any	Any	Any	Any
Networking - Address Mask Request (IC...		Domain	Yes	Allow	No	Any	Any	Any	ICMPv4	Any	Any	Any	Any
Networking - Echo Request (ICMPv4-In)		Domain	Yes	Allow	No	Any	Any	Any	ICMPv4	Any	Any	Any	Any
Networking - Echo Request (ICMPv6-In)		Domain	Yes	Allow	No	Any	Any	Any	ICMPv6	Any	Any	Any	Any
Networking - Redirect (ICMPv4-In)		Domain	Yes	Allow	No	Any	Any	Any	ICMPv4	Any	Any	Any	Any

Web Application Firewalls

- When it comes to HTTP traffic, regular firewalls are not very helpful

- Yet we know that most web attacks use regular HTTP channels: XSS, SQL injection



ModSecurity Deployment Modes

modsecurity

Open Source Web Application Firewall

Home | Projects | Documentation | Download | Contact | Blog

ModSecurity || Overview New Features Roadmap

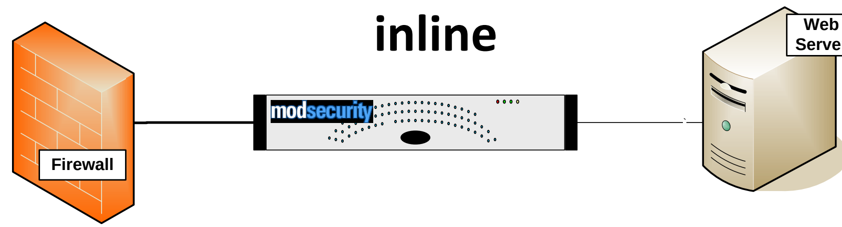
ModSecurity: Overview

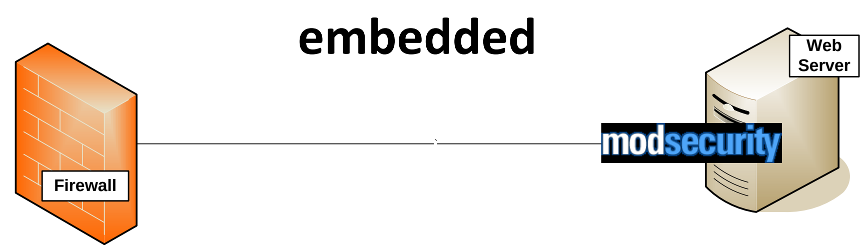
With over 70% of all attacks now carried out over the web application level, organisations need every help they can get in making their systems secure. Web application firewalls are deployed to establish an external security layer that increases security, detects, and prevents attacks before they reach web applications.

HTTP Traffic Logging

Web servers are typically well-equipped to log traffic in a form useful for marketing analyses, but fall short when it comes to logging of traffic to web applications. In particular, most are not capable of logging the request bodies. Your adversaries know this, and that is why most attacks are now carried out via POST requests, rendering your systems blind.

ModSecurity makes full HTTP transaction logging possible, allowing complete requests and responses to be logged. Its logging facilities also allow fine-grained decisions to be made about exactly what is logged and when, ensure only the relevant data is recorded.





Case Study: 1=1

- Classic example of an SQL injection attack
- Often used as a signature.
- But, can be avoided easily using:
  - Encoding: 1%3D1
  - White Space: 1   =%091
  - Comments 1 /* This is a comment */ = 1
- Actually not required at all by attacker.
  - Any true expression would work: 2 > 1
  - In some cases, a constant would also work. In MS-Access all the following are true: 1, “1”, “a89”, 4-4.
- No simple generic detection

Generic Application Layer Signatures

- Detect attack indicators and not attack vectors:
  - `xp_cmdshell`
  - “<“, single quote - Single quote is very much needed to type *O'Brien*
  - `select`, `union` – *which are English words*
- Aggregate indicators to determine an attack:
  - Very strong indicators: xp_cmdshell, varchar,
  - Sequence: *union* …. *select*, *select* … *top* … *1*
  - Amount: *script*, *cookie* and *document* appear in the same input field.
  - Sequence over multiple requests from the same source.

Snort Sig for Bugtraq Vuln #21799

```
/cacti/cmd.php?1+1111)/**/UNION/**/SELECT/**/2,0,1,1,127
.0.0.1,null,1,null,null,161,500, proc,null,1,300,0, ls -
la > ./rra/suntzu.log,null,null/**/FROM/**/host/*+11111
```

Snort Signature:

```
alert tcp $EX… T any -> $H… $HTTP_PORTS
(
  msg:"BLEEDI… EB Cacti cmd.php Remote Arbitrary
  SQL Command Execution Attempt";
  flow:to_server,established;
  uricontent:"/cmd.php?"; nocase;
  uricontent:"UNION"; nocase;
  uricontent:"SELECT"; nocase;
  reference:cve,CVE-2006-6799; r… raq,21799;
  … type: web-application-att… 334; rev:1;
```

Does the application accepts POST requests?

Signature built for specific exploit

An SQL injection does not have to use SELECT or UNION

UNION and SELECT are common English words. So is SELECTION

Back to Bugtraq vulnerability #21799

The Core Rule Set Generic Detection

Supports any type of parameters, POST , GET or any other

```
Se… _FILENAME|ARGS|ARGS_NAMES|
RE… RS|!REQUEST_HEADERS:Referer \

        "(?:\b(?:(?:s(?:elect\b(?:.{1,100}?\b(?:(?:length|count|top)\b.{1,100
}?\bfrom|from\b.{1,100}?\bwhere)|.*?\b(?:d(?:ump\b.*\bfrom|ata_type)|
(?:to_(?:numbe|cha)|inst)r))|p_(?:(?:addextendedpro|sqlexe)c|(?:oacreat|p
repar)e|execute(?:sql)?|makewebtask)|ql_(?:… … … \

        "capture,log,deny,t:replaceComments, t:urlDecodeUni,
t:htmlEntityDecode, t:lowercase,msg:'SQL Injection Attack. Matched
signature <%{TX.0}>',id:'950001',severity:'2'"
```

Every SQL injection related keyword is checked

Common evasion techniques are mitigated

SQL comments are compensated for

Back to Bugtraq Vuln #21799 Virtual Patching

```
<LocationMatch :"/cmd.php$">
      SecRule QUERY_STRING "^[\d\s]*$” “deny,log"
</LocationMatch>
```

Parameters Must Be Numeric

Or

```
SecRule REQUEST_FILENAME :"/cmd.php$" “deny,log"
```

Actually script should not be run remotely

Simpler, isn’t it?

ModSecurity Core Rules

- HTTP Protection: detecting violations of the HTTP protocol and a locally defined usage policy.
- Real-time Blacklist Lookups: utilizes 3rd Party IP Reputation
- Web-based Malware Detection: identifies malicious web content by check against the Google Safe Browsing API.
- HTTP Denial of Service Protections: defense against HTTP Flooding and Slow HTTP DoS Attacks.
- Common Web Attacks Protection - detecting common web application security attack.
- Automation Detection - Detecting bots, crawlers, scanners and other surface malicious activity.
- Integration with AV Scanning for File Uploads - detects malicious files uploaded through the web application.
- Tracking Sensitive Data - Tracks Credit Card usage and blocks leakages.
- Trojan Protection - Detecting access to Trojans horses.
- Identification of Application Defects - alerts on application misconfigurations
- Error Detection and Hiding - Disguising error messages sent by the server

Conclusions

- Viruses
- Virus/antivirus coevolution paper discussed
- Intrusion detection
  - Behavioral detection
  - Firewalls
  - Application firewalls
- Advanced attack techniques
  - Heap spraying
  - Heap feng shui
  - JIT spraying