



**EMAGINED SECURITY**



## **The New Intrusion Detection**

**Dr. Eugene Schultz, CISSP, CISM  
Chief Technology Officer  
Emagined Security  
EugeneSchultz@emagined.com**

**SoCal Security Symposium  
Long Beach, California  
October 29, 2009**

# Goals of this presentation



- Goals of this presentation include helping attendees:
  - Understand some of the limitations of current intrusion detection systems (IDSs)
  - Learn why and how attack methods have changed substantially over the past few years and the impact on intrusion detection
  - Be aware of the kinds of intrusion detection methods that are currently more likely to be successful than others



## Introduction

Limitations in Current Intrusion Detection Technology

Changes in Attack Methods in Recent Years

Solutions

Conclusion

# The quandry

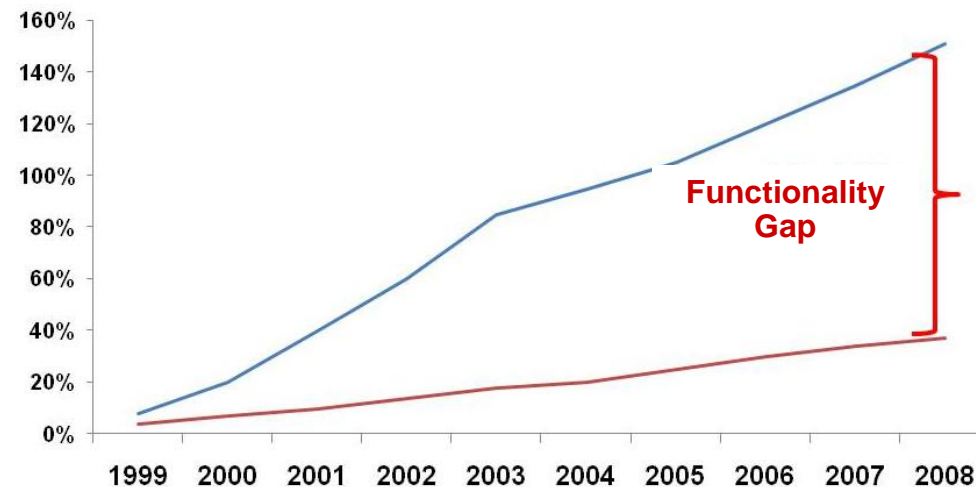


EMAGINED SECURITY

- Intrusion detection has come a long way since its inception

BUT

- The gap between the current state-of-the-art in intrusion detection and what is needed to accurately and reliably discover current attacks has grown to the point that we now must use new methods and approaches





Introduction

## Limitations in Intrusion Detection Technology

Changes in Attack Methods in Recent Years

Solutions

Conclusion

# Limitations of network-based intrusion detection



EMAGINED SECURITY

- Limitations related to network throughput
  - Any rate much greater than 1 – 2 Gb/sec starts to become a problem for network-based intrusion detection
  - Getting top-performance hardware on systems that house network-based IDSs helps very little
  - Network detection rate and processing speed will probably always lag behind the actual rate of traffic flow
- Packet loss can have huge consequences
- Packet fragmentation causes complications
- Susceptible to “slow and gradual” attacks
- Cannot deal with encrypted network traffic
- More...

# Limitations of host-based intrusion detection



EMAGINED SECURITY

- In and of itself, host-based intrusion detection is based only on whatever any single host can record
- Constitutes the latest possible warning and intervention
- Audit logs and other data used and produced by host-based IDSs are often deleted or tampered with
- Can get expensive very quickly



- Different tools and attack methods usually have a “fingerprint” that enables us to identify them
- Fingerprints = signatures
- Well-known attack techniques have their own signatures
  - Exploiting bugs and poor configuration in the Network File System (NFS) in \*nix systems
  - Exploiting bugs in Windows Remote Procedure Call (RPC) and the Server service
  - Much more...
- Today’s generation of IDSs are still very signature-reliant

# Limitations of signatures



EMAGINED SECURITY

- Signatures are probably still the most intuitive way to detect attacks, BUT
- Signatures have significant limitations, including:
  - They are post hoc in nature and thus cannot be used to discover new attacks
  - Some attacks do not have distinguishing signatures
  - Signature analysis tends to be slow and cumbersome
  - They are useless in network-based IDSs when network traffic is encrypted
  - Many signatures in current IDSs are badly outdated
  - Powerful tools that can defeat signature-based IDSs are available
  - They miss today's "subtle" attacks

# Behavioral-based intrusion detection



EMAGINED SECURITY

- Based on known, bad and unusual actions by users
- Circumvents many of the limitations associated with signature-based detection

BUT

- Generally yields unacceptably high false alarm rates



Introduction

Limitations in Intrusion Detection Technology

Changes in Attack Methods in Recent  
Years

Solutions

Conclusion

# Radical changes on the attack front



EMAGINED SECURITY

- Commonly occurring attack methods have changed substantially over the last five to ten years
  - “Frontal assault hacks” used to be common
  - With motivation for cyberattacks having changed so substantially, attack methods have changed accordingly
- Today’s attacks
  - Are much more subtle
  - Target applications (Web, Microsoft Office, Adobe Acrobat and Flash Player, and more)
  - Often involve sending many small pieces of content that must be reassembled by the destination host

# Radical changes on the attack front



EMAGINED SECURITY

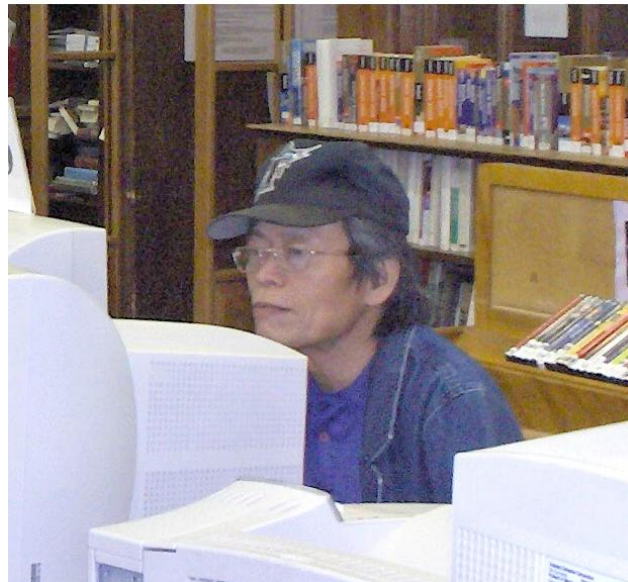
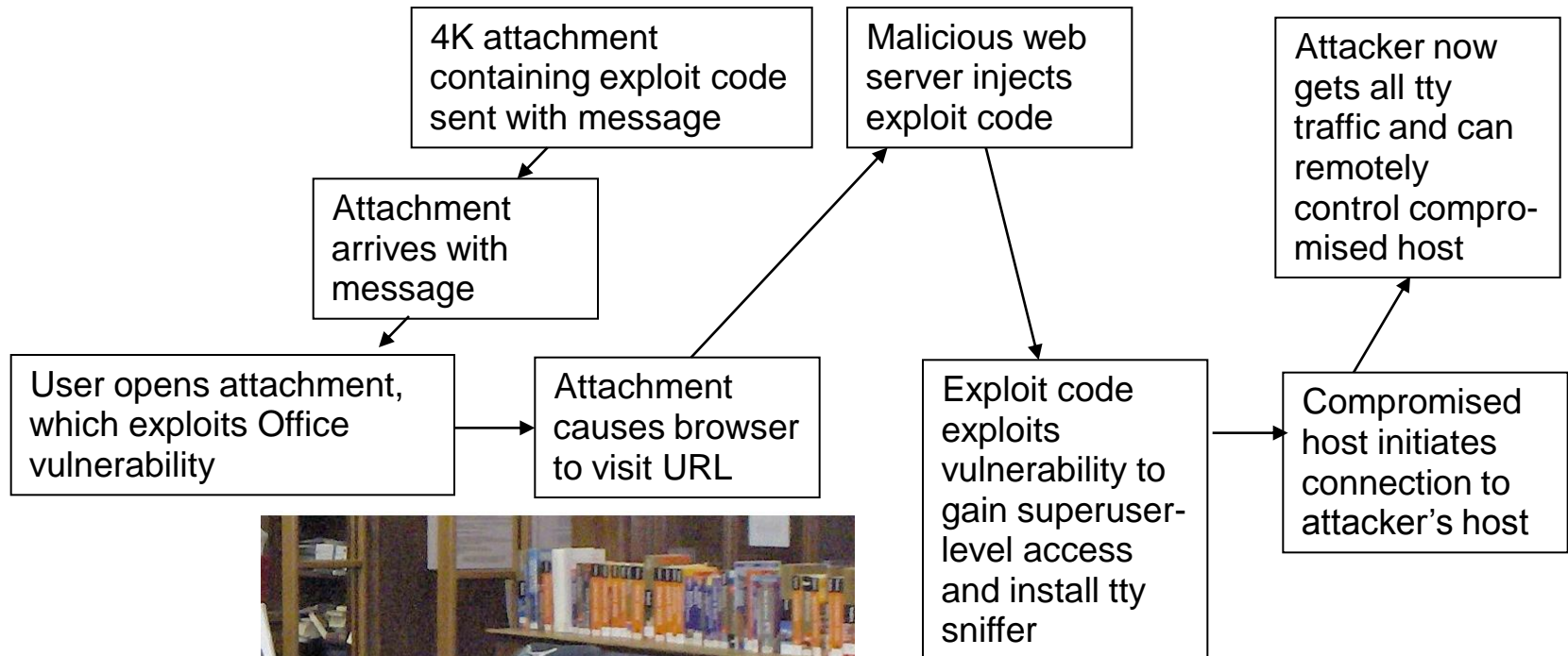
- Today's more subtle attacks have substantially changed the nature of intrusion detection
  - Less reliance on conventional IDSs (and IPSs) per se
  - More reliance on very small indicators of attacks
  - More reliance on discovery methods that are more labor-intensive
- *Common denominator in today's attacks: presence of malware!*

***Continued***

# Today's "hacks"



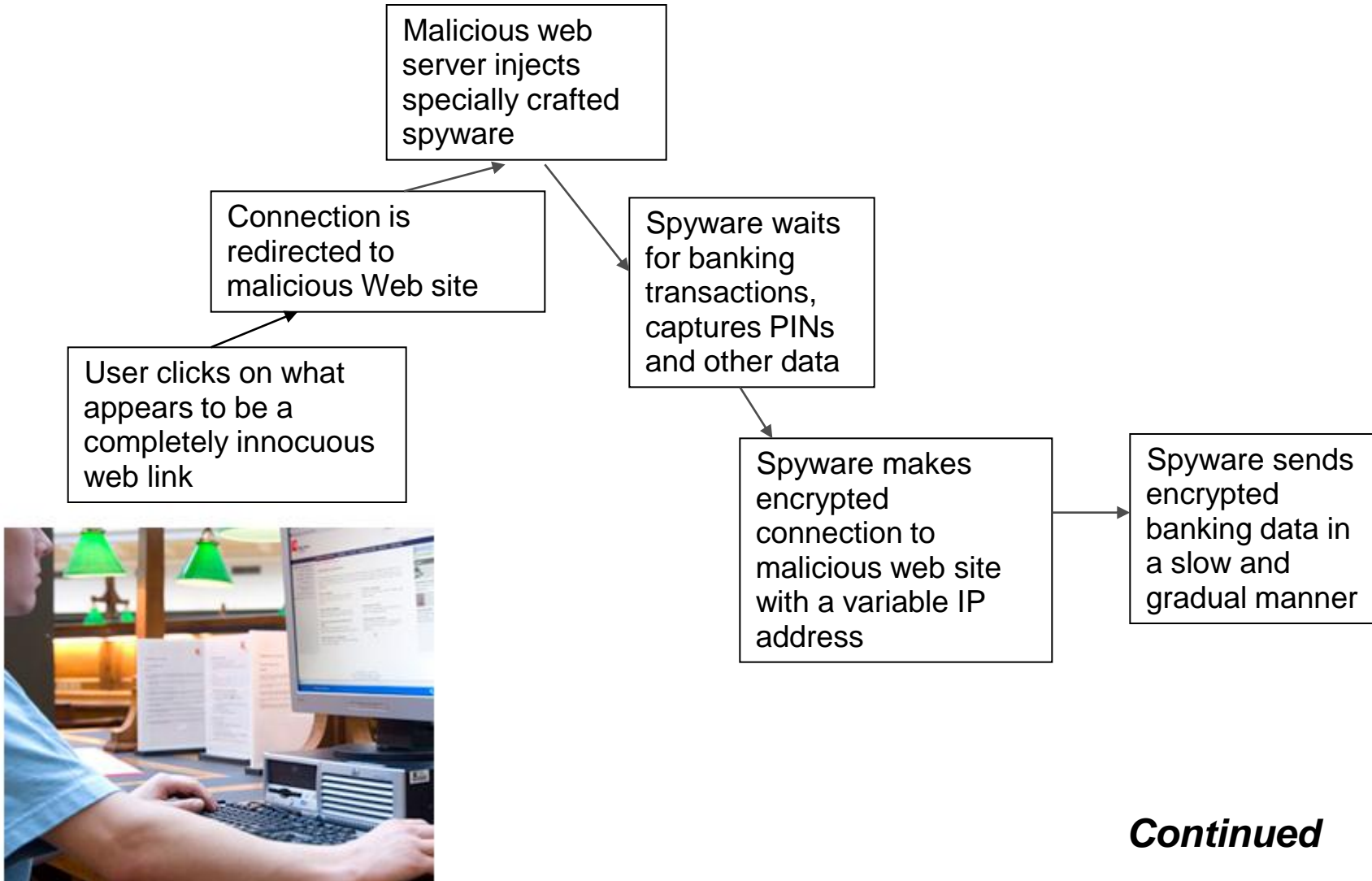
EMAGINED SECURITY



# Today's "hacks"



EMAGINED SECURITY



**Continued**



Introduction

Limitations in Intrusion Detection Technology

Changes in Attack Methods in Recent Years

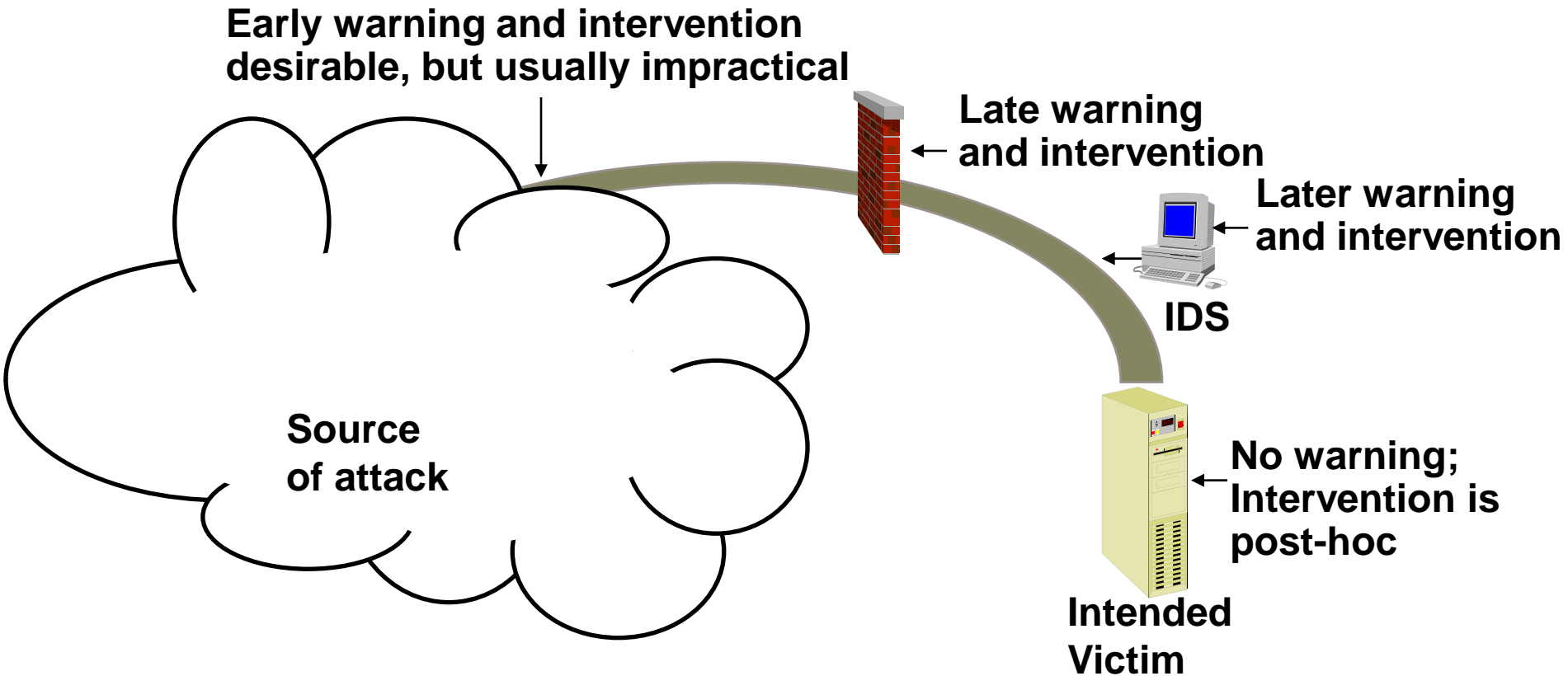
**Solutions**

Conclusion

# How intrusion detection is changing



EMAGINED SECURITY



# Today's intrusion detection: What works and what doesn't



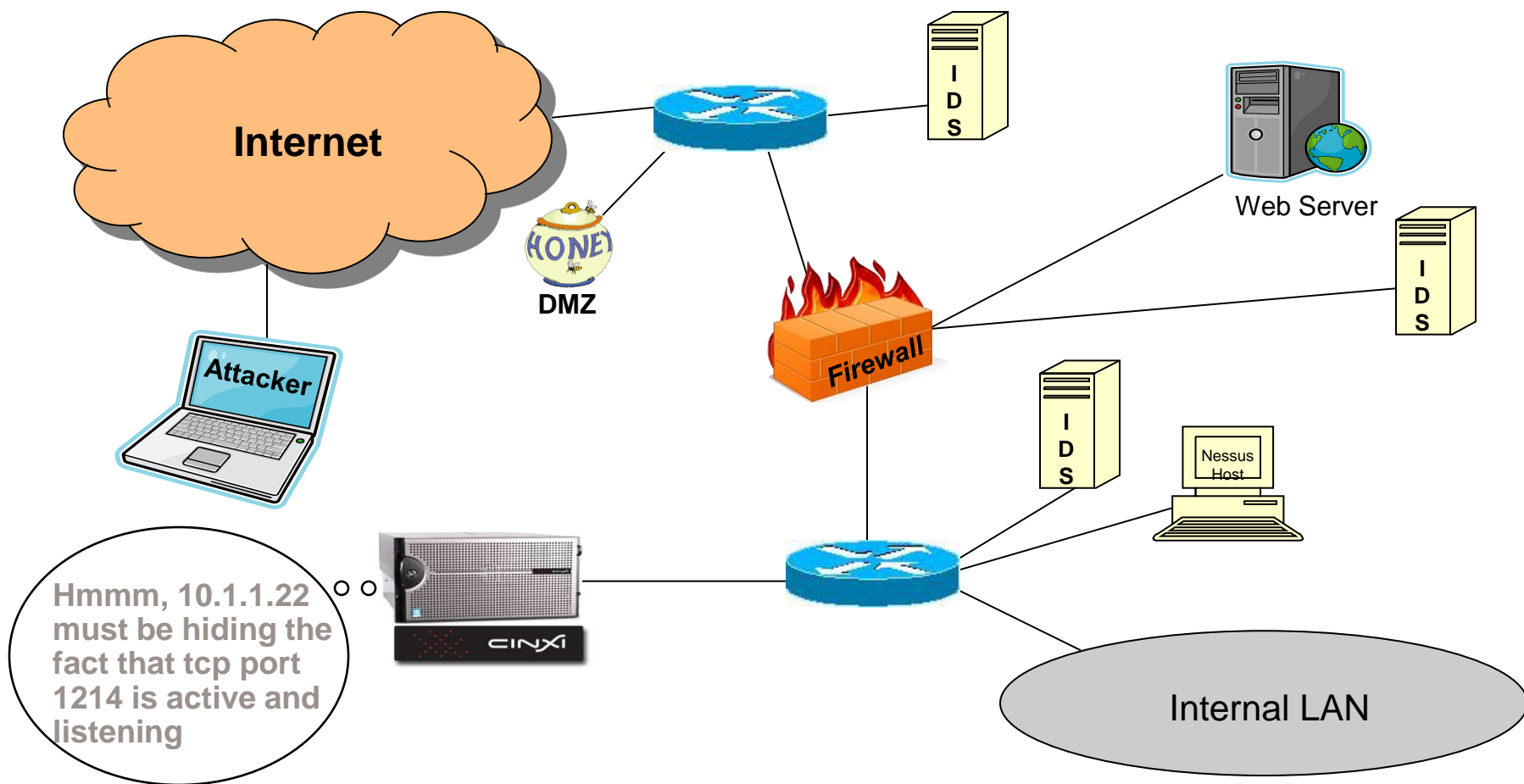
EMAGINED SECURITY

<b>LESS SUITABLE</b>	<b>MORE SUITABLE</b>
<p>Analysis of source-IP addresses</p> <p>Deep packet inspection</p> <p>Stateful inspection</p> <p>Signature-based detection</p> <p>Analysis of connections</p>	<p>Content analysis</p> <p>Context analysis</p> <p>Event correlation (real time and historical)</p> <p>Looking for hard-to-find indications of compromise such as malware in individual hosts</p>

# How event correlation can lead to the discovery of rootkits



EMAGINED SECURITY



# Conventional methods for discovering malware



EMAGINED SECURITY

- Running AV and other detection software (including Tripwire)
- Network-based intrusion detection—can spot traffic to and from suspicious ports
- History file data—if not erased, can be used to identify time malware was installed and commands it has executed
- Commands and tools can list processes and ports
  - › `lsOf` (for \*nix)
  - › `fport` (for Windows)
  - › `ps` (for Linux and Unix)
  - › `pview` and `ptree` (for Windows)
- *Important caveat—in general, do not count on conventional detection methods for discovering malware!*



- Signature-based detection
  - Same method as in virus detection
  - Limited in usefulness because rootkits can control memory reads of detection application
  - Example of tools: vendor rootkit detection tools, chkrootkit
- Behavior detection
  - Logic is to look for deviations from normal system behavior (e.g., function pointers that do not work in connection with ntoskrnl.exe, increases in instruction count, and much more)
  - Example of tools: VICE and Patchfinder
- Cross view-based detection
  - Logic is to compare enumerated elements such as files, processes and Registry keys with and without calling common APIs
  - Example of tools: Rootkit revealer, Klister, Blacklight and more)

# Rootkit detection methods



EMAGINED SECURITY

- Integrity-based detection
  - Logic is to compare CRC and hash values of files and directories from one point in time to another
  - Example of tools: Tripwire and System Virginty Verifier
- Hardware-based detection
  - A PCI card with its own CPU inspects the physical memory
  - Example: Copilot
  - Major disadvantage—cost (\$)
- Event correlation
  - Logic based on the fact that rootkits may be able to hide themselves on *hosts*, but give indications of their presence on the *network*
  - Example of tools: enVision (RSA), ArcSight, Q1 Labs Radar, OpenService
- Manual inspection of potentially compromised system

**(Continued)**

# Manually finding rootkits in Windows hosts



EMAGINED SECURITY

- Windows rootkits are almost impossible to manually discover because they tend to hook
  - Operating system APIs
  - Event logging APIs (EventLog.GetEventLogs, EventLog.WriteEntry, and others)
- Many Windows rootkits do not delete evidence of their installation, however—inspect logs (especially the System Log) for
  - Suspicious service starts and/or stops
  - Stop errors
  - Device driver errors
  - Large increases in size of virtual memory
  - More

# Using the /proc file system



EMAGINED SECURITY

- Is a pseudo-file system in many flavors of \*nix that interfaces with kernel data structures
- A numerical subdirectory (named by the process ID) exists for each running process
  - Each contains directories and pseudofiles

**cmdline**—complete command line used for the process

**cwd**—link to the current working directory

**environment**—the environment for the process

**exe**—the actual path of the command

**fd**—which files are open

**ksyms**—all variables and functions that are exported via LKMs

**mem**—pages of a process' memory

**maps**—currently mapped memory addresses

**net**—current network configuration and status, including all open sockets

**root**—what the root of the filesystem is

**stat**—status information about the process

**statm**—information concerning the memory status of pages

**tty**—the subdirectory that holds entries for tty drivers

**version**—the version of the kernel

# Using the /proc file system



EMAGINED SECURITY

- Enables you to find the contents (files and subdirectories) of programs
- To use this file system, you need to first find the PID of the process of interest:

```
USER      PID %CPU %MEM    SZ   RSS TT  STAT  START   TIME  COMMAND
root      770 33.0  1.9    20    88 p1  R     Jan 5   11437:40 /usr/x
```

- You can access these structures by cd'ing into `/proc`
- If you cd into `/proc` for process 770:

```
#cd /proc/770
```

- You can list the contents

```
#ls -al
```

***Continued***

# Using /proc: netstat output in a Linux host



EMAGINED SECURITY

## Active Connections

Proto	Local Address	Foreign Address	State	
TCP	0.0.0.0:13	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:17	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:19	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:1805	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:1817	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:1993	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:4075	0.0.0.0:0	LISTENING	
TCP	0.0.0.0:4609	0.0.0.0:0	LISTENING	
TCP	127.0.0.1:1041	0.0.0.0:0	LISTENING	
TCP	127.0.0.1:1045	127.0.0.1:1046	ESTABLISHED	TCP
	127.0.0.1:1046	127.0.0.1:1045	ESTABLISHED	
TCP	127.0.0.1:43958	0.0.0.0:0	LISTENING	
TCP	128.3.9.240:139	0.0.0.0:0	LISTENING	
TCP	128.3.9.240:1267	212.199.157.29:2049	ESTABLISHED	TCP
	128.3.9.240:1065	212.199.157.29:2905	ESTABLISHED	TCP
	128.3.9.240:1077	212.199.157.29:1946	ESTABLISHED	TCP
	128.3.9.240:2106	212.199.157.29:111	ESTABLISHED	

# Using /proc: /proc/net/tcp output



EMAGINED SECURITY

```
sl local_address rem_address st tx_queue rx_queue tr tm->when retrnsmt uid timeout inode
0: 00000000:000D 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 310 1
1: 00000000:0011 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 316 1
2: 00000000:0013 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 322 1
3: 00000000:0087 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 346 1
4: 00000000:02BD 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 368 1
...
16: 0100007F:0408 0100007F:0017 01 00000000:00000000 00:00000000 00000000 1000
```

**sl**—the number of the line in the output

**local\_address**—the local IP address and port number for the socket, with the IP address displayed as a little-endian four-byte hexadecimal number in which the least significant byte appears first. The port number is a two-byte hexadecimal number.

**rem\_address**—the remote IP address and port number for the socket

**st**—the socket status

**tx\_queue:rx\_queue**—the transmit and receive queues size

**tr:tm->when**—indicates whether a timer is active for this socket (0 means not active). The tm->when field indicates the time remaining before a timeout starts.

**retrnsmt**—not used

**uid**—the ID of the user who owns the socket

**time-out**—not used

**inode**—the socket to the Linux virtual file system.

# Finding rogue loadable kernel modules (LKMs) in Solaris



EMAGINED SECURITY

- Create a dedicated dev file on the host in question

```
bash-2.04b# mkfile 512m /data/datadump
```

- Make the newly created dev file the dump device

```
bash-2.04b# dumpadm -d /data/datadump
```

```
Dump content: kernel pages
```

```
Dump device: /data/datadump (dedicated)
```

```
Savecore directory: /var/crash/file
```

```
Savecore enabled: yes
```

- Use the `savecore` command to make a system dump

```
bash-2.04b# savecore -L
```

```
dumping to /data/dumpfile, offset 65536, content: kernel
```

```
100% done: 8018 pages dumped, compression ratio 2.73, dump  
succeeded
```

```
System dump time: Fri May 9 12:37:02 2009
```

```
Constructing namelist /var/crash/hack/unix.0
```

```
Constructing corefile /var/crash/hack/vmcore.0
```

```
100% done: 8018 of 8018 pages saved
```

# Finding rogue loadable kernel modules in Solaris



EMAGINED SECURITY

- Copy the dump files to a known good host
- Use mdb to determine the modules that were loaded when the host dumped

```
bash-2.04b# mdb -k unix.0 vmcore.0
```

***Continued***



Introduction

Limitations in Intrusion Detection Technology

Changes in Attack Methods in Recent Years

Solutions

Conclusion

- Radical changes in attack methods have occurred over the last few years
- These changes have
  - Exposed additional major limitations in today's intrusion detection and intrusion prevention technology
  - Served as an impetus to use other methods\* (many of which are minimally automated or not automated at all) to detect such attacks
  - Forced intrusion detection analysts to pay a disproportionate amount of attention to what is going on in *individual hosts*
- Computer forensics and intrusion detection are becoming more intertwined now than ever before

\* - This is by no means intended to imply that today's IDSs are useless! They still have much value and should be part of your organization's security tool arsenal.

# Questions?



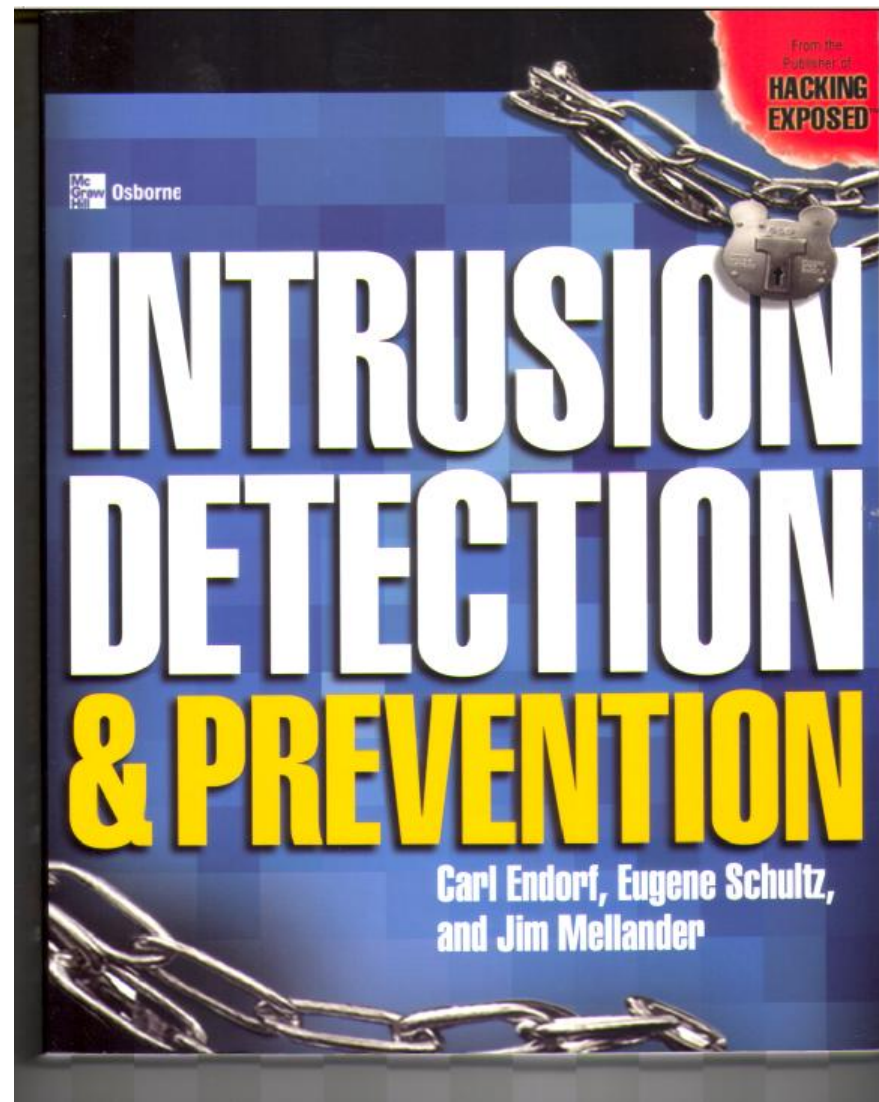
EMAGINED SECURITY

Emagined Security  
2816 San Simeon Way  
San Carlos, CA 94070  
(650) 593-9829  
eugeneschultz@emagined.com  
web: [www.emagined.com](http://www.emagined.com)

See [blog.emagined.com](http://blog.emagined.com)

For a PDF copy of these slides send  
email to:

YvonneVega@emagined.com





*EMAGINED SECURITY*