Interested in learning more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Evaluating Untrusted Software In a Controlled Environment

Tools and techniques of reverse engineering allow the professional analyst to identify and describe in detail the behavior of malicious software in a test lab environment. However, many users and organizations lack both the resources and time to subject untrusted software to such stringent tests. To address the key business concern of "is this software safe to download and use?", a lightweight filtering methodology is proposed that will yield a reasonably reliable answer with a very modest resource and time investment.

Copyright SANS Institute
Author Retains Full Rights

AD

Evaluating Untrusted Software In a Controlled Environment

Jeff Reava

June 20, 2002

GSEC 1.4

Abstract

Tools and techniques of reverse engineering allow the professional analyst to identify and describe in detail the behavior of malicious software in a test lab environment. However, many users and organizations lack both the resources and time to subject untrusted software to such stringent tests. To address the key business concern of "is this software safe to download and use?", a lightweight filtering methodology is proposed that will yield a reasonably reliable answer with a very modest resource and time investment.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Introduction

Among the utilities and applications readily available for download from the Internet is an abundance of malware. Among other things, malicious software contains "features" that can include initiating unauthorized connections to other systems, downgrading security settings, or even installing listener software designed to wait for remote commands.

The development of techniques for examining the specific behavior of suspected malicious software has been the subject of a number of research projects [1][2][3].

These approaches are an excellent starting point for professionals and researchers seeking to understand the functioning of hostile software as fully as possible. However, many system engineers and security professionals often approach the evaluation of untrusted software with a different set of requirements. The question posed by non-technical users seeking to evaluate a piece of software is much more basic, namely "is this program 'safe' to run on my PC?"

The goal of this paper is to propose a rudimentary method for detecting malicious software behavior with reasonable accuracy. "Safe" software is not defined as software free from bugs, or any level of resistance to abuse. The emphasis is on screening rather than analysis, optimized for technical users who on occasion need to conduct a cursory examination of software and provide a quick response. It is also assumed that the examiner will have limited time and resources to dedicate to the effort.

Assumptions

First and foremost, the filtering process assumes that malicious software behavior is overt, and evidenced by changes to local system settings, network interfaces, or network traffic. The screening techniques are not exhaustive, and assume that the underlying operating system behavior can still be trusted.

Also assumed is that the screener performing the test is able to clearly identify a pattern of "normal" system behavior based on observation and experience, and is able to separate between expected application operation and unwanted "functionality." The last case may be an artificial distinction if the organization's policy disallows specific modes of operation.

Finally, since the screening techniques represent a "black box" approach to testing software during a short period of time, the intent is to catch the obvious offenders. Software that does not fail in the analysis is not necessarily trustworthy. Organizations need to move quickly, and the intent is to provide information that enables a vastly

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

improved mode of decision making over “eyeballing” untrusted software, while at the same time requiring very little additional effort.

Overview

The key to successfully screening for malware lies in isolating and identifying suspicious system changes. Typical changes in system behavior can include attempts to connect to websites or open file shares, send email, or open other communication channels with remote systems, launching new services and/or opening listening ports on a system that wait for remote commands, or modifying startup settings to ensure that it will always run each time the system reboots.

With the exception of network traffic analysis, the primary technique involves taking snapshots of key system attributes before and after launching the untrusted software, and watching for unexpected differences.

In general, the remainder of this paper will lay out these stages in a step-by-step process that can be followed much like a cookbook recipe. Like all good recipes, though, the best results come from interpretation and improvisation. The basic steps listed will work, but are somewhat arbitrary approaches toward achieving the original goal: filtering out overtly malicious software. Regardless of the tools selected, the most reliable results will come from selecting a system state that is composed of many different attributes.

In order to analyze a system, the following components are necessary:

- Test laptop or desktop system running Windows (9X/NT/2000).
- A “Router” system: a Windows desktop/laptop with two network adapters and Internet Connection Sharing enabled.
- A crossover network cable used to connect the test and router systems, and a LAN line to connect the router to the network.
- Windump: for capturing network traffic.[4]
- Nmapnt: to scan for open ports on the test system.[5]
- GHOST: for system backup and restoration[6]
- InCtrl5: for system snapshot[7]
- FPORT for identifying processes on the test system that are bound to listening tcp and/or udp ports.[8]
- PSLIST for listing all processes on a system, whether bound to ports or not.[9]
- Boot disk for network image transfer.[10]

Instructions for installing and configuring all of the components listed above will be provided in the stages that follow. At a high level, the process of analyzing untrusted software is encapsulated in seven basic steps:

1. Configure the test system.
2. Configure a filtering router to pass traffic transparently between the test system and the network.
3. Install network analysis tools on the filtering router.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

4. Snapshot the test system to develop a detailed picture of known, trusted system behavior
5. Transparently isolate the test system from the production environment.
6. Install the untrusted software to be tested
7. Analyze the change in system behavior based on the software install.

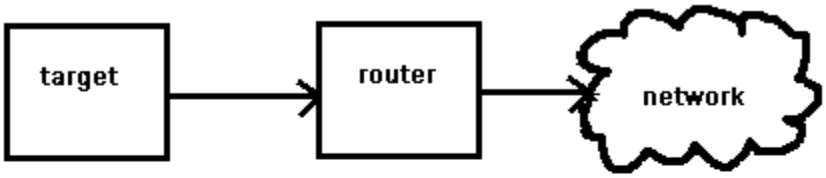

FIGURE 1

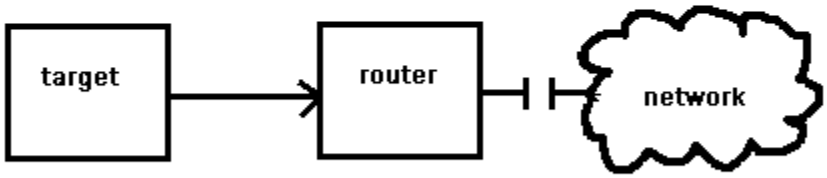

FIGURE 2

Graphically, **figure 1** illustrates steps 1-3 where the router system is placed between the test system and the network environment to which that test system is normally connected.

The logical isolation required by step 5 is illustrated in figure 2. Note that the isolation is logical – not physical. The target system will have no initial indication that the link to the network has been broken. Ideally, this will allow the untrusted software to behave normally without risking any impact in the production environment.

Since there are numerous sub-processes in the overall testing flow, they will be described in detail in the step-by-step discussion that follows.

1. CONFIGURE TEST ENVIRONMENT

Key requirements of the test environment include establishing a very close approximation to the production environment without requiring substantial time and resource investment by the tester. Ideally, an actual production machine will be "sandboxed" for the duration of the test. That is, the production system is logically isolated from its networked environment in a way that does not identify the established connections as broken. To restore the system to its original state following a test, drive imaging software is used.

The overhead associated with archiving a functioning production image is accepted on the premise that the tester does not need to apply patches or configuration changes to the image in use in order to keep test client configurations synchronized with production systems. Using imaging rather than VMWARE[11] virtualization means that repeated testing is slower in this setting. However, the test environment is not virtualized and there

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

is no client administration unique to the testing environment between testing sessions. If virtualization poses no drawbacks and frequent repeated testing is a requirement, VMWARE is the better choice.

1.1 Configure Test Machine

Configuration changes to the test machine should be as minor as possible, so that the untrusted software behaves the same on the test machine as would be expected on a production system. Only one utility will be installed on the test machine: InCtrl5. The other two utilities, Foundstone's FPORT.EXE and Sysinternal's PSLIST.EXE are not installed in the sense that they change the underlying configuration of the system, but are launched from a working directory on that system.

After building the test machine, it should be configured as follows:

1. Verify that the software installed on the test system matches the production environment, especially with respect to service packs, hotfixes, desktop applications, virus scan, and Email client software.
2. Download and install InControl5. Select the default options.
3. Download and unzip FPORT into a working directory such as c:\snapshot.
4. Download, unzip and install PSLIST into the snapshot directory.
5. Test FPORT and PSLIST by launching them from a command line and verify that are functioning correctly.

1.2 Configure Router

The "router" for this test is a dual-homed Windows 2000 Professional workstation, using the Internet Connection Sharing feature. It must provide initial connectivity to the test system that permit access to network services such as Email, file shares and web services. Again, the assumption is that malware will attempt to use existing tools and connections when they are present.

The router also must be able to monitor the traffic passing from the test machine to the network, and it must also have the ability to actively scan the test machine for external evidence of new or altered network services. Finally, it must be able to quietly break the connection between the test system and the network so that any malicious software will still activate normally, but not be able to communicate.

1.2.1 Enable Internet connection sharing

Dual-home the router system by ensuring that both network cards are installed and have drivers configured. For the purpose of simply passing traffic, instructions for enabling Internet Connection Sharing provided by Microsoft in Knowledge Base article Q237254 [12] describe the basic steps required, which are:

1. In Control Panel, double-click Network and Dial-Up Connections.
2. Right-click the connection you want to share, and then click Properties.
3. Click the Sharing tab, and then click to select the Enable Internet Connection Sharing for this connection check box.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

4. If the connection you are sharing is a Dial-up connection and you want the connection to dial automatically when another computer on your home network attempts to use external resources, click to select the Enable on-demand dialing check box.

NOTE: To enable ICS in Windows, you must have administrative rights.

1.2.2 Enable network monitoring

In addition to controlling connectivity between the test system and the network, the router system must also be able to "sniff" the traffic passing over the wire. The baseline traffic of a known, trusted configuration will be used as the basis of comparison to any traffic generated when the test software is launched. Windump (TCPdump for Windows) is selected for this task.

To install windump:

1. First download the WinPcap packet capture architecture and execute the auto-installer, following the prompts.
2. Next, download Windump.exe to a directory such as c:\tools.
3. Launch windump from a command line to verify that it is working properly. Begin with commands such as:

`windump –D` (list all adapters)

`windump –i#` (where # is the number of the adapter facing the test system, which was listed from the **`–D`** parameter.)

Traffic sniffing provides passive monitoring of network activity originating from the test system. However, it will not detect a passive listener installed on the host system itself. For that type of screening, the router system must actively scan the test system directly to see how it responds.

1.2.3 Enable remote host scanning

Port scanning the test system after the untrusted software has been activated can confirm the presence of an active listening service on the test system. Probing for listening services is performed by Eeye's port of NMAP to the NT platform. To install it:

1. Download the zip archive from http://www.eeye.com/html/Research/Tools/nmapnt.html into a directory such as c:\tools.
2. Unzip nmapNTsp1 into the tools directory.
3. From a command line, launch:

`nmapnt.exe` (no parameters).

4. When executed without parameters, usage instructions will appear, followed by a list of available network adapters that can be specified when launching a scan. Note the adapter facing the test system.

2. ENVIRONMENT PREP

2.1 Establish Test Machine Connectivity

Connectivity from the test machine to the network should be transparent through the router. With the router system running and connected to both the network and to the test system (either directly using a crossover cable, or indirectly through a hub/switch), start

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

the test system. It should pick up an IP address from the router, along with default gateway information, and the DNS entries, etc. from the network. Connect drive letters to network shares, launch the Email client, and make all of the other typical connections that will be observed as part of the test. Ideally, the test system should not behave any differently passing traffic through the Win2K router than it did connected directly to the network. This is important, because the observed baseline connectivity represents the point of reference for identifying unexpected suspicious behavior.

2.2 Archive System Configuration

At this stage, the test system most likely reflects a working production configuration that may need to be restored. To ensure that this exact configuration can be reproduced either for additional testing, or to ensure the ability to safely 'undo' the changes made by the software being tested, GHOST by Symantec provides an easy way to archive and restore an exact copy of the hard drive's contents. NOTE: This is the only stage of the testing process where an account with write privileges to the router's hard drive should be used. During the actual testing of untrusted software, the account used on the test machine should have NO privileges to read/write data from shares on the router. The test system should have no connections directly to the router at the time of the test.

The archive process requires a boot disk with network connectivity that will enable the test system to map a drive letter to the router, run GHOST, and copy the image over that connection to the router for storage. Symantec provides support and guidance for running GHOST. As for a networking boot disk, there are many options available from www.bootdisk.com.

Assuming that a network boot disk is available to connect to the router, and a copy of GHOST, these steps provide a straightforward approach:

1. From router: Make an image folder on a local drive with at least 2GB of free space. From a command prompt, type:

```
md c:\image
```

2. From router, to share out an imaging folder, type

```
net share image=c:\image
```

3. Since the share in the previous step is made available to everyone with full access, restrict those permissions to <account name> which will be used from the network boot disk (not the testing environment) to copy over an image:

```
cacls c:\image /T /G <account name>:C
```

4. From the router, copy **GHOST.EXE** into the image shared folder.
5. From the test machine, boot from the network boot disk and map a drive letter such as X: to the image share on the router.
6. To copy the entire hard drive from the test machine to the router using compression:

```
ghost.exe -clone,mode=dump,src=2,dst=x:\test.gho
-z3
```

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

7. For safety, the share should be removed from the router, just in case the untrusted test software turns out to be malware of the SMB share scanning variety. Again, from a command line:

```
net share image /delete
```

3. SNAPSHOT THE ENVIRONMENT

Now that a functioning test system has been archived, what are the key elements of a known good state? From the perspective of reverse engineering, that question requires a very detailed view of file, process and memory usage examined step by step. Simple screening for overtly malicious (or at least suspicious) software relies on a much less detailed view, optimizing for the speed of examination instead of exhaustively detailing the results. Again, the assumption is that malware will manifest itself either by installing unwanted components, establish a listening service on the system, or attempt to "phone home", surreptitiously sending data from the test machine to a remote host.

To compensate for the limited detail provided by any single test, this approach relies on observing the test system from multiple unique perspectives in the belief that if one indicator does not detect malware behavior, another will.

3.1 Network Snapshot

A network snapshot in this context reflects elements of baseline, expected traffic by the test system. Given that the test system has drive letter connections mapped, Email open, etc. there is a certain level and type of network traffic that will be passing through the router. For screening purposes, the assumption is that malware will be "noisy" in comparison to usual traffic over established channels, or will attempt to establish new connections.

To track traffic through the router with a minimum of background 'noise', only communications on the test interface that originated from the router will be examined using windump with syntax such as:

```
windump –i1 src testpc > test1.dmp
```

Where **`–i1`** refers to the interface communicating with the test system, **`src`** limits capture to packets originating from the test system (testpc), and **`test1.dmp`** is the target file where console output is redirected to be saved.

A brief sampling of traffic from a relatively quiet test system should give a reasonably distinct reference point to observe changes in activity. After capturing for a short period (30 seconds) terminate the capture using Control+C.

NOTE: A network traffic snapshot will only be an effective screening measure if the test system exhibits an observable increase in activity; if the system "at rest" is still fairly "noisy" in terms of the amount of traffic produced, it will be much more difficult to detect malware if the evidence appears as a change in traffic type instead of volume.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

3.2 Service Snapshot

Another possible indicator of malware behavior is a change in listening services active on the test machine. Using nmapNT.exe, a baseline of listening ports can also provide an indicator of malware if those services change during the untrusted software test. For example:

```
nmapnt –e# –p1-65535 –v > maptcp1.txt
```

provides verbose output from a TCP connect scan against the test machine. The **–e#** parameter explicitly requires the test machine interface (where # is 0, 1, etc.) is to be used for running the scan.

UDP ports should also be scanned:

```
nmapnt –e# –sU –p1-65535 –v > mapudp1.txt
```

By this point, the network and service snapshots of the system will provide enough information to filter against "aggressive" malware that actively attempts to connect to external systems, and/or is openly listening for external connection attempts. To provide a more complete filter and to attribute any suspicious behavior to a specific process, closer examination from the workstation itself is required.

3.3 Host Snapshot

Host snapshots should identify three elements of system state: processes, listening services, and file updates. Taking snapshots on the host system provides a different filter perspective than examining the system externally from a network connection. Starting with a known, trusted configuration, the initial host snapshot should identify all files and configuration settings using InCtrl5, all active processes using SysInternal's PSLIST, and with the help of Foundstone's FPORT, associate running processes with open ports listening for connections.

3.3.1 File and System Configuration Snapshot

InCtrl5 notes changes that occur to files and registry settings as a result of configuration changes or application installs. Using InControl5 in the default "2 phase mode":

1. Start|Programs|InCtrl5|InCtrl5.exe
2. Click GO. Wait for the snapshot to complete.
3. Click OK and InCtrl5 will exit. At this point, InCtrl5 writes snapshot files to c:\program files\InCtrl5 named xxx$$$.$$1. These files will be deleted at the end of the snapshot.
4. Run the software to be tested, and then launch InCtrl5 again.
5. Press the Install Complete button to finish the analysis. InCtrl5 will identify the changes that have occurred and present them in an html report format.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

3.3.2 Open Process Snapshot

PSLIST dumps a summary of open processes running on the system. To capture the open process list to a file, run pslist.exe from a command line with this syntax:

```
pslist > pslist1.txt
```

After taking a second snapshot, running the file compare command line utility will identify differences between the two:

```
fc pslist1.txt pslist2.txt
```

3.3.3 Process Listener Snapshot

While PSLIST identifies all open processes, Foundstone's FPORT shows the subset of those processes that are network aware and will correlate those new files with open listeners. From a command line:

```
fport  > fport1.txt
```

Save the initial snapshot for comparison. After taking a second snapshot, the file compare utility can quickly identify differences that appear as a result of testing the untrusted software.

4. ISOLATE TEST ENVIRONMENT

Unlike virtual machine approaches to malware testing, this filter environment is built on direct, live network connections. This poses a risk to production networks unless the connectivity can be disabled at the "external" router interface (figure 2). At the same time, the test system must still operate on the assumption that network links to the production environment are still "alive". Internet connection sharing provides a graceful way to reliably disable the connectivity without alerting the test host.

To disable routing, execute these commands on the router system:

1. Start|Settings|Network and Dial-Up Connections|<shared connection>
2. Click the Sharing tab.
3. Uncheck the box marked "Enable Internet Connection Sharing for this connection."
4. Click OK.

5. ACTIVATE UNTRUSTED SOFTWARE

In the brief window of time between breaking the connection and when the network connections begin to time out, activate the untrusted software from the test system. Even though the router is not passing that traffic through to the rest of the network, if the application makes an attempt to ftp or post data to a remote website, those packets will appear during the network capture.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

5.1 Snapshot of Test Software Network Activity

The quality of the initial network snapshot will greatly affect the reliability of these results as an indicator of malware activity. If the initial system snapshot was relatively "quiet" and suddenly posts to http://www.foo.com, a positive outcome is easy to conclude. However, "noisy" test systems may generate a baseline level of traffic that make it more difficult to isolate and characterize the presence of additional suspicious packets. Additionally, as the broken link begins to trigger timeouts, isolation becomes problematic.

Once a reasonable amount of post-launch snapshot packet data has been captured, it should be saved to a file. As in 3.1:

```
windump –i1 src testpc > test2.dmp
```

Windump begins capturing to a file, and can be exited using Control+C once a reasonable sample has been collected.

5.2 Test Software Service Activity

Again, using nmapnt with the same parameters as in 3.2, enumerate the list of open ports visible from the network for comparison and analysis against the trusted configuration set.

5.3 Identify Host System Changes

5.3.1 Open Process Snapshot

Follow the same steps in 3.3.2 to generate an open process snapshot after the untrusted software is executed. The untrusted executable should appear in the list, but the list should be checked for any additional unexpected processes.

5.3.2 Process Listener Snapshot

As in 3.3.3, a second snapshot taken by launching FPORT from the command line can identify changes to running processes that are actively listening to open ports on the test system. Using the syntax:

`fport > fport2.txt` captures the current processes.

Again from the command line, usage of the file compare utility makes comparison simple:

```
fc fport1.txt fport2.txt
```

The value of this test assumes that the untrusted software being tested is not designed for use as a listening server process. If it is, the open port(s) will of course appear and this is expected behavior for the application. Filtering using FPORT identifies unexpected features, but does not discern at this level of detail. For untrusted software with server functionality, a reverse engineering approach is much more likely to yield a useful result.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

5.3.3 File and System Configuration Snapshot Analysis

Completing the snapshot process involves running InCtrl5 a second time, following the prompts and sifting through the results. Because even a "quiet" system generates a significant volume of changes within the registry, it will be important to "tune" InCtrl5 to trim the results to items of interest. Malware at times makes configuration changes to ensure that it is loaded at every startup. For this reason, the following set of registry changes may indicate the need for further examination:
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services

Also potentially suspicious are entries in the Current User or system startup groups (folders):
C:\Documents and Settings\<USERID>\Start Menu\Programs\Startup
C:\Documents and Settings\All Users\Start Menu\Programs\Startup

Or, less commonly, legacy startup files:
C:\autoexec.bat
C:\winnt\win.ini
C:\winnt\system.ini
C:\config.sys

6. SAVE OUTPUTS FOR CURSORY SUMMARY REPORT

As essential as the actual testing itself is the preservation of findings in a meaningful output format. From the router system, connect to the test system and save the comparison results from the local fport, pslist and InCtrl5 snapshots on the router system along with the nmapnt snapshot that was preserved there.

7. RESTORE SYSTEM

Restore the system by booting from a network disk, connecting to the router where the system image is stored, and bring down the system image to overwrite the test system. More specifically:

1. Boot the test system from a network boot disk.
2. Map a drive letter such as X: to the image share on the router.

```
net use x: \\router\image
```

3. Restore the entire hard drive, overwriting the tested configuration:

```
ghost.exe -clone,mode=load,src=x:\test.gho,dst=1
```

4. Remove the floppy disk and reboot the system.

8. CONCLUSION

Given before/after snapshots of system processes, file system changes, network activity and open services, do the filter results represent a comprehensive view of actual system state? Based on the initial assumptions used to define the filter processes, that's not a reasonable conclusion to reach. At the same time, if none of the recorded changes to

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

system state are unauthorized or unexpected, the rigor involved makes a conclusion that "software X is safe" a reasonable one.

Reverse engineering, and unstructured informal testing of untrusted software represent two extremes in the continuum of testing options available when examining software operation in a lab environment. As extremes, they optimize different requirements, namely speed of testing and reliability of results.

Most users and organizations have a different set of requirements, somewhere in the middle between the two. Regardless of that position, all analysis involves comparing changes in state on a variety of attributes and as such are easily operationalized in a testing process that reflects those requirements in each defined phase of environmental setup and analysis.

References

[1] The Honeynet Project, "The Reverse Challenge: Your challenge is to analyze a binary captured in the wild." http://www.honeynet.org/reverse/
[2] Zeltser, Lenny, "Reverse Engineering Malware", May 2001
http://www.zeltser.com/sans/gcih-practical/revmalw.html
[3] Arnold, et al, "An Environment for Controlled Worm Replication and Analysis or: Internet-inna-Box"
http://www.research.ibm.com/antivirus/SciPapers/VB2000INW.htm
[4] Windump http://windump.polito.it/
[5] Nmapnt http://www.insecure.org/
[6] GHOST http://www.symantec.com/ghost/
[7] InCtrl5 http://downloads-zdnet.com.com/3000-2096-10059071.html
[8] FPORT http://www.foundstone.com/knowledge/free_tools.html
[9] PSLIST http://www.sysinternals.com/
[10] Bootdisks http://www.bootdisk.com/
[11] VMWARE http://www.vmware.com/
[12] Microsoft Product Support Services, "How to Enable Internet Connection Sharing on a Network Connection in Windows 2000 (Q237254)"
http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q237254&ID=KB;EN-US;Q237254&
Anderson, Ross, "Security Engineering: A Guide to Building Dependable Distributed Systems", Wiley & Sons, 2001
Mandia & Prosise, "Incident Response: Investigating Computer Crime", Osborne, 2001
Kruse & Heiser, "Computer Forensics: Incident Response Essentials", Addison Wesley, 2002

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: May 24th, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced