"NIST Special Publication 800-42

Guideline on Network Security Testing

Recommendations of the National Institute of Standards and Technology

John Wack, Miles Tracy, Murugiah Souppaya

C O M P U T E R   S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

October 2003
"
"
"

U.S. Department of Commerce
Donald L. Evans, Secretary

Technology Administration
Phillip J. Bond, Under Secretary for Technology

National Institute of Standards and Technology
Arden L. Bement, Jr., Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Special Publication 800-series reports on ITL's research, guidance, and outreach efforts in computer security, and its collaborative activities with industry, government, and academic organizations.

"

"

National Institute of Standards and Technology Special Publication 800-42
Natl. Inst. Stand. Technol. Spec. Publ. 800-42, XX pages (October, 2003)
CODEN: XXXXX

"

"
"
"

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2001

"

For sale by the Superintendent of Documents, U.S. Government Printing Office
Internet: bookstore.gpo.gov — Phone: (202) 512-1800 — Fax: (202) 512-2250
Mail: Stop SSOP, Washington, DC 20402-0001

Authority

The National Institute of Standards and Technology (NIST) have developed this document in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided A-130, Appendix III.

This guideline has been prepared for use by federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright though attribution is desired by NIST.

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Acknowledgements

The authors, John Wack and Murugiah Souppaya of NIST and Miles Tracy of Booz Allen Hamilton (BAH), wish to acknowledge staff at NIST and BAH who reviewed drafts of this publication and made substantial improvements to its quality, including Timothy Grance, Wayne Jansen, Tom Karygiannis, Peter Mell, Robert Sorensen, and Marianne Swanson.

Table Of Contents

1. Introduction 1-1
1.1 Purpose and Scope 1-1
1.2 Definitions 1-2
1.3 Audience 1-3
1.4 Document Organization 1-3

2. Security Testing and the System Development Life Cycle 2-1
2.1 System Development Life Cycle 2-1
2.1.1 Implementation Stage 2-2
2.1.2 Operational Stage 2-3
2.2 Documenting Security Testing Results 2-3
2.3 Roles and Responsibilities 2-4
2.3.1 Senior IT Management/Chief Information Officer (CIO) 2-4
2.3.2 Information Systems Security Program Managers (ISSM) 2-4
2.3.3 Information Systems Security Officers (ISSO) 2-5
2.3.4 System and Network Administrators 2-5
2.3.5 Managers and Owners 2-5

3. Security Testing Techniques 3-1
3.1 Roles and Responsibilities for Testing 3-1
3.2 Network Scanning 3-2
3.3 Vulnerability Scanning 3-3
3.4 Password Cracking 3-6
3.5 Log Reviews 3-7
3.6 File Integrity Checkers 3-8
3.7 Virus Detectors 3-9
3.8 War Dialing 3-10
3.9 Wireless LAN Testing (“War Driving”) 3-10
3.10 Penetration Testing 3-11
3.11 Post-Testing Actions 3-16
3.12 General Information Security Principles 3-17
3.13 Summary Comparisons of Network testing Techniques 3-19

4. Deployment Strategies for Security Testing 4-1

4.1 Determine the Security Category of the Information System 4-1
4.2 Determine Cost of Performing Each Test Type per System 4-2
4.3 Identify Benefits of Each Test Type per System 4-2
4.4 Prioritize Systems for Testing 4-2

Appendix A. Terminology A-1

Appendix B. References B-1

Appendix C. Common Testing Tools C-1

C.1 File Integrity Checkers C-1
C.2 Network Sniffers C-2
C.3 Password Crackers C-3
C.4 Scanning and Enumeration Tools C-4
C.5 Vulnerability Assessment Tools C-6
C.6 War Dialing Tools C-7
C.7 Wireless Networking Tools C-8
C.8 Host Based Firewalls C-9

Appendix D. Example Usage Of Common Testing Tools D-1

D.1 Nmap D-1
D.2 L0pht Crack D-8
D.3 LANguard D-9
D.4 Tripwire D-11
D.5 Snort D-16
D.6 Nessus D-21

Appendix E. Index E-1

List Of Tables

Table 3.1: Comparison of Testing Procedures 3-20
Table 3.2: Summarized Evaluation and Frequency Factors 3-21
Table C.1: File Integrity Checker Tools.. C-1
Table C.2: Network Sniffer Tools C-2
Table C.3: Password Cracking Tools... C-3
Table C.4: Scanning and Enumeration Tools C-5
Table C.5: Vulnerability Assessment Tools C-6
Table C.6: War Dialing Tools C-7
Table C.7: Wireless Networking Testing Tools C-8
Table C.8: Host-Based Firewall Tools C-9

List Of Figures

Figure 3.1: Four-Stage Penetration Testing Methodology 3-13
Figure 3.2: Attack Phase Steps with Loopback to Discovery Phase 3-14

Executive Summary

Securing and operating today's complex systems is challenging and demanding. Mission and operational requirements to deliver services and applications swiftly and securely have never been greater. Organizations, having invested precious resources and scarce skills in various necessary security efforts such as risk analysis, certification, accreditation, security architectures, policy development, and other security efforts, can be tempted to neglect or insufficiently develop a cohesive, well-though out operational security testing program.

This guide stresses the need for an effective security testing program within federal agencies. Testing serves several purposes. One, no matter how well a given system may have been developed, the nature of today's complex systems with large volumes of code, complex internal interactions, interoperability with uncertain external components, unknown interdependencies coupled with vendor cost and schedule pressures, means that exploitable flaws will always be present or surface over time. Accordingly, security testing must fill the gap between the state of the art in system development and actual operation of these systems. Two, security testing is important for understanding, calibrating, and documenting the operational security posture of an organization. Aside from development of these systems, the operational and security demands must be met in a fast changing threat and vulnerability environment. Attempting to learn and repair the state of your security during a major attack is very expensive in cost and reputation, and is largely ineffective. Three, security testing is an essential component of improving the security posture of your organization. Organizations that have an organized, systematic, comprehensive, on-going, and priority driven security testing regimen are in a much better position to make prudent investments to enhance the security posture of their systems.

NIST recommends the following:

Make network security testing a routine and integral part of the system and network operations and administration. Organizations should conduct routine tests of systems and verify that systems have been configured correctly with the appropriate security mechanisms and policy. Routine testing prevents many types of incidents from occurring in the first place. The additional costs for performing this testing will be offset by the reduced costs in incident response.

Test the most important systems first. In general, systems that should be tested first include those systems that are publicly accessible, that is, routers, firewalls, web servers, e-mail servers, and certain other systems that are open to the public, are not protected behind firewalls, or are mission critical systems. Organizations can then use various metrics to determine the importance or criticality of other systems in the organization and proceed to test those systems as well.

Use caution when testing. Certain types of testing, including network scanning, vulnerability testing, and penetration testing, can mimic the signs of attack. It is imperative that testing be done in a coordinated manner, with the knowledge and consent of appropriate officials.

Ensure that security policy accurately reflects the organization's needs. The policy must be used as a baseline for comparison with testing results. Without appropriate policy, the usefulness of testing is drastically limited. For example, discovering that a firewall permits the flow of certain types of traffic may be irrelevant if there is no policy that states what type of traffic or what type of network activity is permitted. When there is a policy, testing results can be used to improve the policy.

Integrate security testing into the risk management process. Testing can uncover unknown vulnerabilities and misconfigurations. As a result, testing frequencies may need to be adjusted to meet the

prevailing circumstances, for example, as new controls are added to vulnerable systems or other configuration changes are made because of a new threat environment. Security testing reveals crucial information about an organizations security posture and their ability to surmount attack externally or to avoid significant financial or reputational cost from internal malfeasance. In some cases, the results of the testing may indicate that policy and the security architecture should be updated. Hence, this insight into the security posture of an organization is highly relevant to a well-functioning risk management program.

Ensure that system and network administrators are trained and capable. Security testing must be performed by capable and trained staff. Often, individuals recruited for this task are already involved in system administration. While system administration is an increasingly complex task, the numbers of trained system administrators generally has not kept pace with the increase in computing systems. Competent system administration may be the most important security measure an organization can employ, and organizations should ensure they possess a sufficient number with the required skill level to perform system administration and security testing correctly.

Ensure that systems are kept up-to-date with patches. As a result of security testing, it may become necessary to patch many systems. Applying patches in a timely manner can sharply reduce the vulnerability exposure of an organization. Organizations should centralize their patching efforts so as to ensure that more systems are patched as quickly as possible and immediately tested.

Look at the big picture. The results of routine testing may indicate that an organization should readdress its systems security architecture. Some organizations may need to step back and undergo a formal process of identifying the security requirements for many of its systems, and then begin a process of reworking its security architecture accordingly. This process will result in increased security inefficiency of operations with fewer costs incurred from incident response operations.

Understand the capabilities and limitations of vulnerability testing. Vulnerability testing may result in many false positive scores, or it may not detect certain types of problems that are beyond the detection capabilities of the tools. Penetration testing is an effective complement to vulnerability testing, aimed at uncovering hidden vulnerabilities. However, it is resource intensive, requires much expertise, and can be expensive. Organizations should still assume they are vulnerable to attack regardless of how well their testing scores indicate.

1. Introduction

The Internet has brought about many changes in the way organizations and individuals conduct business, and it would be difficult to operate effectively without the added efficiency and communications brought about by the Internet. At the same time, the Internet has brought about problems as the result of intruder attacks, both manual and automated, which can cost many organizations excessive amounts of money in damages and lost efficiency. Thus, organizations need to find methods for achieving their mission goals in using the Internet and at the same time keeping their Internet sites secure from attack.

Computer systems today are more powerful and more reliable than in the past; however they are also more difficult to manage. System administration is a complex task, and increasingly it requires that system administration personnel receive specialized training. In addition, the number of trained system administrators has not kept pace with the increased numbers of networked systems. One result of this is that organizations need to take extra steps to ensure that their systems are configured correctly and securely. And, they must do so in a cost-effective manner.

This document deals with the subject of testing Internet connected systems and networks when they are in operation. Security testing is perhaps the most conclusive determinant of whether a system is configured and continues to be configured to the correct security controls and policy. The types of testing described in this document are meant to assist network and system administrators and related security staff in keeping their systems operationally secure and resistant as much as possible to attack. These testing activities, if made part of standard system and network administration, can be highly cost-effective in preventing incidents and uncovering unknown vulnerabilities.

1.1 Purpose and Scope

The purpose of this document is to provide guidance on network security testing. This document identifies network testing requirements and how to prioritize testing activities with limited resources. It describes network security testing techniques and tools.[1] This document provides guidance to assist organizations in avoiding duplication of effort by providing a consistent approach to network security testing throughout the organization's networks. Furthermore, this document provides a feasible approach for organizations by offering varying levels of network security testing as appropriate to the organization's mission and security objectives.

The main focus of this document is the basic information about techniques and tools for individuals to begin a network security testing program. This document is by no means all-inclusive. Individuals and organizations should consult the references provided in this document as well as vendor product descriptions and other sources of information.

While this document describes generalized network security testing that is applicable to all networked systems, it is aimed more towards the following types of systems:

- Firewalls, both internal and external

[1] There are many excellent freeware (no fee required for license) and shareware (requires nominal fee for license) security tools. However, great care should be used in selecting freely available tools. Generally, freeware/shareware tools should not be used unless an expert has reviewed the source code or they are widely used and are downloaded from a known safe repository. Appendix C provides a list of well-known tools for downloading. The costs of supporting "freeware" applications can be significant, as in-house experts may have to be developed to support any widely used application. The cost of this support should be compared to the cost of a commercial product to determine which is the most cost effective.

- Routers and switches
- Related network-perimeter security systems such as intrusion detection systems
- Web servers, email servers, and other application servers
- Other servers such as for Domain Name Service (DNS) or directory servers or file servers (CIFS/SMB, NFS, FTP, etc.)

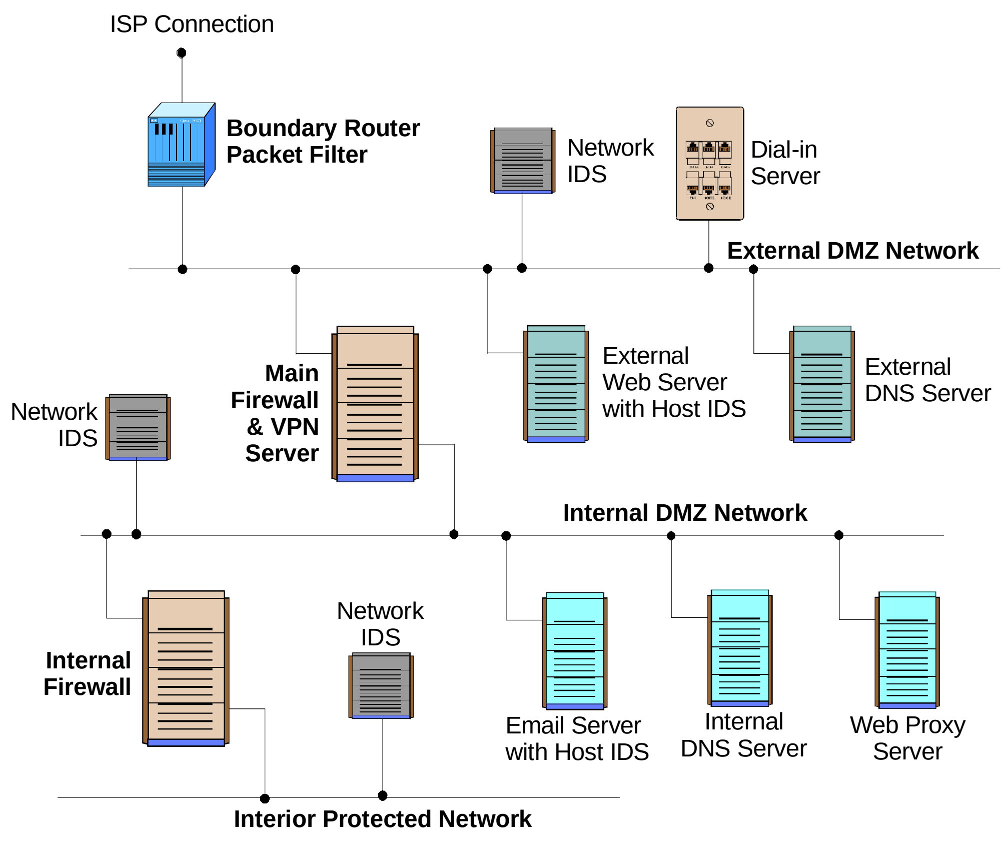


Figure 1.1: Examples of Mission Critical Systems for Initial Testing

These systems generally should be tested first before proceeding onto testing general staff and related systems, i.e., desktop, standalone, and mobile client systems.

The tests described in this document are applicable to various stages of the system development lifecycle, and are most useful as part of a routine network security test program to be conducted while systems are running in their operational environments.

1.2 Definitions

This document uses the terms **system**, **network security testing**, **operational testing**, and **vulnerability** extensively. For the purposes of this document, their definitions will be as follows:

System – A system is any of the following:

- Computer system (e.g., mainframe, minicomputer)
- Network system (e.g., local area network [LAN])
- Network domain
- Host (e.g., a computer system)
- Network nodes, routers, switches and firewalls
- Network and/or computer application on each computer system.

Network Security Testing – Activities that provide information about the integrity of an organization's networks and associated systems through testing and verification of network-related security controls on a regular basis. "Security Testing" or "Testing" is used throughout this document to refer to Network Security Testing. The testing activities can include any of the types of tests described in Chapter 3, including network mapping, vulnerability scanning, password cracking, penentration testing, war dialing, war driving, file integrity checking, and virus scanning.

Operational Security Testing – Network security testing conducted during the operational stage of a system's life, that is, while the system is operating in its operational environment.

Vulnerability – A bug or misconfigurations or special sets of circumstances that could result in an exploitation of that vulnerability. For the purposes of this document, a vulnerability could be exploited directly by an attacker, or indirectly through automated attacks such as Distributed Denial of Service (DDOS) attacks or by computer viruses.

1.3 Audience

This document should be useful for security program managers, technical and functional managers, network and system administrators, and other information technology (IT) staff members. It provides them with a structured approach to network security testing. Management personnel who are responsible for systems can apply the testing procedures and tools discussed in this document to become informed about the status of the assets under their stewardship. This document can also assist in evaluating compliance with their organization's security standards and requirements. Managers can also use this information to evaluate the technical basis and support for the decision-making processes. This document can be used to formulate a test plan to verify and assess the implemented security controls.

1.4 Document Organization

This document is organized as follows:

- Chapter 1 provides an introduction and overview
- Chapter 2 describes the rationale for testing and the overall relationship of security testing to the system's life cycle

- Chapter 3 defines network security testing goals and objectives, identifies critical areas of testing, prioritizes testing requirements, and describes active and passive types of testing
- Chapter 4 describes how to prioritize security testing, with possibly limited resources
- Appendix A lists acronyms used in this document
- Appendix B lists the references used in this document
- Appendix C provides a list of testing tools
- Appendix D provides examples of tool usage.

Several sections of the document assume some advanced knowledge of Linux/Unix, Windows NT/2000/XP, and TCP/IP networking.

2. Security Testing and the System Development Life Cycle

The primary reason for testing the security of an operational system is to identify potential vulnerabilities and subsequently repair them. The number of reported vulnerabilities is growing daily; for example, the number of new information system vulnerabilities reported to the Bugtraq[2] database has more that quintupled since the start of 1998, from an average of 20 to over 100 per month. The number of computers per person in many organizations continues to rise, increasing the demands on competent and experienced system administrators. Consequently, it is imperative that organizations routinely test systems for vulnerabilities and misconfigurations to reduce the likelihood of system compromise.

Typically, vulnerabilities are exploited repeatedly by attackers to attack weaknesses that organizations have not patched or corrected. A report in a SANS Security Alert, dated May 2000, provides a discussion of this issue: "A small number of flaws in software programs are responsible for the vast majority of successful Internet attacks…. A few software vulnerabilities account for the majority of successful attacks because attackers don't like to do extra work. They exploit the best-known flaws with the most effective and widely available attack tools. And they count on organizations not fixing the problems."[3]

In a study involving federal agencies, security software vendors, security consulting firms, and incident response teams, a consensus was reached on a top 20 list of critical Internet security vulnerabilities.[4] SANS Security Alert lists these vulnerabilities and outlines recommendations and suggestions for overcoming these weaknesses. In this environment, security testing becomes critical to all organizations interested in protecting their networks.

2.1 System Development Life Cycle

Evaluation of system security can and should be conducted at different stages of system development. Security evaluation activities include, but are not limited to, risk assessment, certification and accreditation (C&A), system audits, and security testing at appropriate periods during a system's life cycle. These activities are geared toward ensuring that the system is being developed and operated in accordance with an organization's security policy. This section discusses how network security testing, as a security evaluation activity, fits into the system development life cycle.

A typical systems lifecycle[5] would include the following activities:

1. **Initiation** – the system is described in terms of its purpose, mission, and configuration.
2. **Development and Acquisition** – the system is possibly contracted and constructed according to documented procedures and requirements.
3. **Implementation and Installation** – the system is installed and integrated with other applications, usually on a network.
4. **Operational and Maintenance** – the system is operated and maintained according to its mission requirements.
5. **Disposal** – the system's lifecycle is complete and it is deactivated and removed from the network and active use.

[2] See http://www.securityfocus.com/.
[3] SANS Institute. SANS Security Alert, May 2000. P 1., http://www.sans.org/newsletters/sac/
[4] Ibid, P 1.
[5] There are numerous systems lifecycle models; this model is simplified so as to illustrate those general stages in which security testing can be conducted.

Typically, network security testing is conducted after the system has been developed, installed, and integrated during the **Implementation** and **Operational** stages. Figure 2.1 shows a flow diagram of the system development lifecycle.

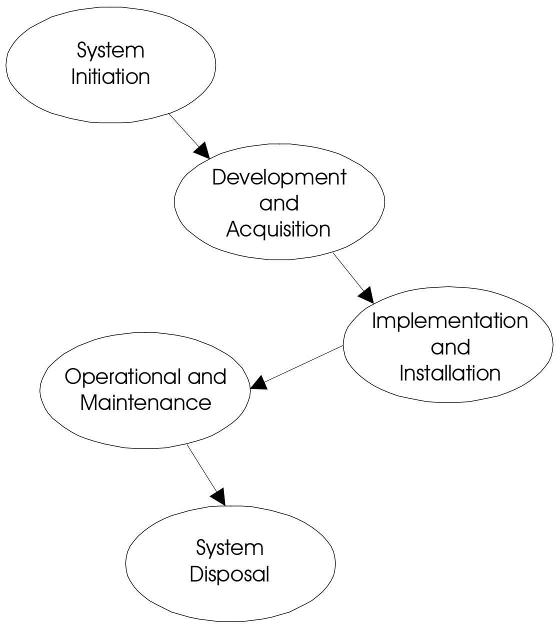

System Initiation
Development and Acquisition
Implementation and Installation
Operational and Maintenance
System Disposal

Figure 2.1 System Development Life Cycle

2.1.1 Implementation Stage

During the Implementation Stage, Security Testing and Evaluation should be conducted on particular parts of the system and on the entire system as a whole. Security Test and Evaluation (ST&E) is an examination or analysis of the protective measures that are placed on an information system once it is fully integrated and operational. The objectives of the ST&E are to:

- Uncover design, implementation and operational flaws that could allow the violation of security policy
- Determine the adequacy of security mechanisms, assurances and other properties to enforce the security policy
- Assess the degree of consistency between the system documentation and its implementation.

The scope of an ST&E plan typically addresses computer security, communications security, emanations security, physical security, personnel security, administrative security, and operations security.

All operational security tests described in Chapter 3 should also be used at this stage to ensure that the existing system configuration is as secure as possible prior to full implementation on an active, live network. During the Operational Stage, the operational security tests should be repeated periodically.

NIST Special Publication 800-26, *Security Self-Assessment Guide for IT Systems*,[6] goes into more detail on conducting ST&E testing; readers are encouraged to read this document.

2.1.2 Operational Stage

Once a system is operational, it is important to ascertain its operational status, that is, "…whether a system is operated according to its current security requirements. This includes both the actions of people who operate or use the system and the functioning of technical controls."[7] The tests described in Chapter 3 can be conducted to assess the operational status of the system. The types of tests selected and the frequency in which they are conducted depend on the importance of the system and the resources available for testing. These tests, however, should be repeated periodically and whenever a major change is made to the system. For systems that are exposed to constant threat (e.g., web servers) or that protect critical information (e.g., firewalls), testing should be conducted more frequently.

As shown in Figure 2.2, the Operational Stage is subdivided into two stages to include a Maintenance Stage in which the system may be temporarily off-line due to a system upgrade, configuration change, or an attack.

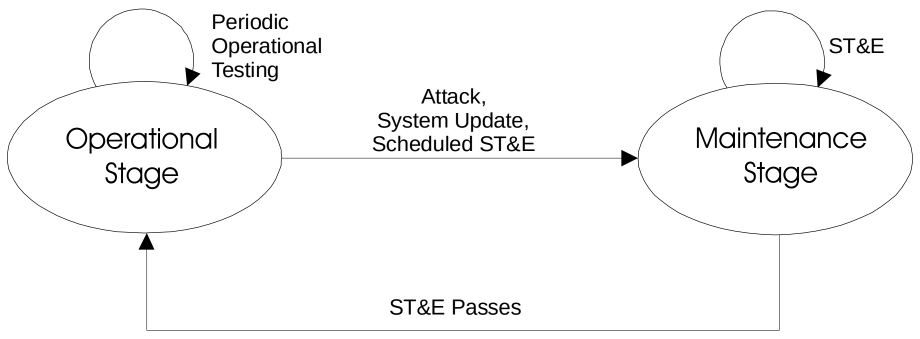

Periodic Operational Testing
ST&E
Attack, System Update, Scheduled ST&E
Operational Stage
Maintenance Stage
ST&E Passes

Figure 2.2 Testing Activities at the Operations and Maintenance Stages

During the Operational Stage, periodic operational testing is conducted (the testing schedules in Table 3.2 can be used). During the Maintenance Stage, ST&E testing may need to be conducted just as it was during the Implementation Stage. This level of testing may also be required before the system can be returned to its operational state, depending upon the criticality of the system and its applications. For example, an important server or firewall may require full testing, whereas a desktop system may not.

2.2 Documenting Security Testing Results

Security testing provides insight into the other system development life cycle activities such as risk analysis and contingency planning. Security testing results should be documented and made available for staff involved in other IT and security related areas. Specifically, security testing results can be used in the following ways:

[6] See http://csrc.nist.gov/publications/nistpubs/800-26/sp800-26.pdf.

[7] National Institute of Standards and Technology, Generally Accepted Principles and Practices for Securing Information Technology systems. September 1996. P 24, see http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf

- As a reference point for corrective action,
- In defining mitigation activities to address identified vulnerabilities,
- As a benchmark for tracing an organization's progress in meeting security requirements,
- To assess the implementation status of system security requirements,
- To conduct cost/benefit analysis for improvements to system security, and
- To enhance other life-cycle activities, such as risk assessments, Certification and Authorization (C&A), and performance improvement efforts.

2.3 Roles and Responsibilities

Because security testing provides input into and can be a part of multiple system development life cycle phases, a number of IT and system security staff may be interested in its execution and result. This section provides a list of those roles and identifies their responsibilities related to security testing. These roles may vary with the organization, however, and not all organizations will have the identical roles described here.

2.3.1 Senior IT Management/Chief Information Officer (CIO)

The Senior IT Management/CIO ensures that the organization's security posture is adequate. The Senior IT Management provides direction and advisory services for the protection of information systems for the entire organization. The Senior IT Management/CIO is responsible for the following activities that are associated with security testing:

- Coordinating the development and maintenance of the organization's information security policies, standards, and procedures,
- Ensuring the establishment of, and compliance with, consistent security evaluation processes throughout the organization, and
- Participating in developing processes for decision-making and prioritization of systems for security testing.

2.3.2 Information Systems Security Program Managers (ISSM)

The Information Systems Security Program Managers (ISSMs) oversee the implementation of, and compliance with the standards, rules, and regulations specified in the organization's security policy. The ISSMs are responsible for the following activities associated with security testing:

- Developing and implementing standard operating procedures (security policy),
- Complying with security policies, standards and requirements, and

- Ensuring that critical systems are identified and scheduled for periodic testing according to the security policy requirements of each respective system.

2.3.3 Information Systems Security Officers (ISSO)

Information Systems Security Officers (ISSOs) are responsible for overseeing all aspects of information security within a specific organizational entity. They ensure that the organization's information security practices comply with organizational and departmental policies, standards, and procedures. ISSOs are responsible for the following activities associated with security testing:

- Developing security standards and procedures for their area of responsibility,
- Cooperating in the development and implementation of security tools and mechanisms,
- Maintaining configuration profiles of all systems controlled by the organization, including but not limited to, mainframes, distributed systems, microcomputers, and dial access ports, and
- Maintaining operational integrity of systems by conducting tests and ensuring that designated IT professionals are conducting scheduled testing on critical systems.

2.3.4 System and Network Administrators

System and network administrators must address the security requirements of the specific system(s) for which they are responsible on a daily basis. Security issues and solutions can originate from either outside (e.g., security patches and fixes from the vendor or computer security incident response teams) or within the organization (e.g., the Security Office). The administrators are responsible for the following activities associated with security testing:

- Monitoring system integrity, protection levels, and security related events,
- Resolving detected security anomalies associated with their information system resources,
- Conducting security tests as required, and
- Assessing and verifying the implemented security measures.

2.3.5 Managers and Owners

Managers and owners of a system oversee the overall compliance of their assets with their defined/identified security requirements. They are also responsible for ensuring that test results and recommendations are adopted as appropriate.

3. Security Testing Techniques

There are several different types of security testing. The following section describes each testing technique, and provides additional information on the strengths and weakness of each. This information is also summarized in Table 3.1 and Table 3.2. Some testing techniques are predominantly manual, requiring an individual to initiate and conduct the test. Other tests are highly automated and require less human involvement. Regardless of the type of testing, staff that setup and conduct security testing should have significant security and networking knowledge, including significant expertise in the following areas: network security, firewalls, intrusion detection systems, operating systems, programming and networking protocols (such as TCP/IP).

The following types of testing are described in this section:

- Network Scanning
- Vulnerability Scanning
- Password Cracking
- Log Review
- Integrity Checkers
- Virus Detection
- War Dialing
- War Driving (802.11 or wireless LAN testing)
- Penetration Testing

Often, several of these testing techniques are used together to gain more comprehensive assessment of the overall network security posture. For example, penetration testing usually includes network scanning and vulnerability scanning to identify vulnerable hosts and services that may be targeted for later penetration. Some vulnerability scanners incorporate password cracking. None of these tests by themselves will provide a complete picture of the network or its security posture. Table 3.1 at the end of this section summarizes the strengths and weaknesses of each test.

After running any tests, certain procedures should be followed, including documenting the test results, informing system owners of the results, and ensuring that vulnerabilities are patched or mitigated. Section 3.11 discusses post-testing actions that should be followed as a matter of course.

3.1 Roles and Responsibilities for Testing

Only designated individuals, including network administrators or individuals contracted to perform the network scanning as part of a larger series of tests, should conduct the tests described in this section. The approval for the tests may need to come from as high as the CIO depending on the extent of the testing. It would be customary for the testing organization to alert other security officers, management, and users that network mapping is taking place. Since a number of these test mimic some of the signs of attack, the appropriate manages must be notified to avoid confusion and unnecessary expense. In some cases, it may be wise to alert local law enforcement officials if, for example, the security policy included notifying law enforcement.

3.2 Network Scanning

Network scanning involves using a port scanner to identify all hosts potentially connected to an organization's network, the network services operating on those hosts, such as the file transfer protocol (FTP) and hypertext transfer protocol (HTTP), and the specific application running the identified service, such as WU-FTPD, Internet Information Server (IIS) and Apache for the HTTP service. The result of the scan is a comprehensive list of all active hosts and services, printers, switches, and routers operating in the address space scanned by the port-scanning tool, i.e., any device that has a network address or is accessible to any other device.

Port scanners, such as nmap[8] (see Appendix B for more information), first identify active hosts in the address range specified by the user using Transport Control Protocol/Internet Protocol (TCP/IP) Internet Control Message Protocol (ICMP) ECHO and ICMP ECHO_REPLY packets. Once active hosts have been identified, they are scanned for open TCP and User Datagram Protocol (UDP) ports[9] that will then identify the network services operating on that host. A number of scanners support different scanning methods that have different strengths and weaknesses that are usually explained in the scanner documentation (see Appendix D for more information). For example, certain scans are better suited for scans through firewalls and others are better suited for scans that are internal to the firewall. Individuals not familiar with the details of TCP/IP protocols should review the references listed in Appendix B.

All basic scanners will identify active hosts and open ports, but some scanners provide additional information on the scanned hosts. The information gathered during this open port scan will often identify the target operating system. This process is called operating system *fingerprinting*. For example, if a host has TCP port 135 and 139 open, it is most likely a Windows NT or 2000 host. Other items such as the TCP packet sequence number generation and responses to ICMP packets, e.g., the TTL (Time To Live) field, also provide a clue to identifying the operating system. Operating system fingerprinting is not foolproof. Firewalls filter (block) certain ports and types of traffic, and system administrators can configure their systems to respond in nonstandard ways to camouflage the true operating system.

In addition, some scanners will assist in identifying the application running on a particular port. For example, if a scanner identifies that TCP port 80 is open on a host, it often means that the host is running a web server. However, identifying which web server product is installed can be critical for identifying vulnerabilities. For example, the vulnerabilities for Microsoft's IIS server are very different from those associated with Apache web server. The application can be identified by "listening" on the remote port to capture the "banner" information transmitted by the remote host when a client (web browser in this example) connects. Banner information is generally not visible to the end-user (for web servers/browsers); however when it is transmitted, it can provide a wealth of information, including the application type, application version and even operating system type and version. Again this is not foolproof since a security conscious administrator can alter the transmitted banners. The process of capturing banner information is sometimes called *banner grabbing*.

While port scanners identify active hosts, services, applications and operating systems, they do NOT identify vulnerabilities (beyond some common Trojan ports). Vulnerabilities can only be identified by a

[8] See http://www.insecure.org for more information and free download.

[9] In TCP/IP terminology, a port is where an application receives information from the transport (TCP/UDP) layers. For example, all data received on TCP port 80 is forwarded to the Web server application. If an IP address identifies a particular host, the port is used to identify a particular service (HTTP, FTP, SMTP, etc.) running on that host.

human who interprets the mapping and scanning results. From these results, a qualified individual can ascertain what services are vulnerable and the presence of Trojans. Although the scanning process itself is highly automated, the interpretation of scanned data is not.

Organizations should conduct network scanning to:

- Check for unauthorized hosts connected to the organization's network,
- Identify vulnerable services,
- Identify deviations from the allowed services defined in the organization's security policy,
- Prepare for penetration testing,
- Assist in the configuration of the intrusion detection system (IDS), and
- Collect forensics evidence.

A relatively high level of human expertise is required to interpret the results. The scanning can also disrupt network operations by consuming bandwidth and slowing network response times. However, network scanning does enable an organization to maintain control of its IP address space and ensure that its hosts are configured to run only approved network services. To minimize disruptions to operations, scanning software should be carefully selected (see Appendix C). Network scanning can also be conducted after hours to ensure minimal impact to operations, with the caveat that some systems may not be turned on.

Network scanning results should be documented and identified deficiencies corrected. The following corrective actions may be necessary as a result of network scanning:

- Investigate and disconnect unauthorized hosts,
- Disable or remove unnecessary and vulnerable services,
- Modify vulnerable hosts to restrict access to vulnerable services to a limited number of required hosts (e.g., host level firewall or TCP wrappers), and
- Modify enterprise firewalls to restrict outside access to known vulnerable services.

3.3 Vulnerability Scanning

Vulnerability scanners take the concept of a port scanner to the next level. Like a port scanner, a vulnerability scanner identifies hosts and open ports, but it also provides information on the associated vulnerabilities (as opposed to relying on human interpretation of the results). Most vulnerability scanners also attempt to provide information on mitigating discovered vulnerabilities.

Vulnerability scanners provide system and network administrators with proactive tools that can be used to identify vulnerabilities before an adversary can find them. A vulnerability scanner is a relatively fast and easy way to quantify an organization's exposure to surface vulnerabilities.[10]

[10] A surface vulnerability is a weakness, as it exists in isolation, independent from other vulnerabilities. The difficultly in identifying the risk level of vulnerabilities is that they rarely exist in isolation. For example there could be several "low risk" vulnerabilities that exist on a particular network that, when combined, present a high risk. A vulnerability scanner

Vulnerability scanners attempt to identify vulnerabilities in the hosts scanned. Vulnerability scanners can also help identify out-of-date software versions, applicable patches or system upgrades, and validate compliance with, or deviations from, the organization's security policy. To accomplish this, vulnerability scanners identify operating systems and major software applications running on hosts and match them with known exposures. Scanners employ large databases of vulnerabilities to identify flaws associated with commonly used operating systems and applications. [11]

The scanner will often provide significant information and guidance on mitigating discovered vulnerabilities. In addition vulnerability scanners can automatically make corrections and fix certain discovered vulnerabilities. This assumes that the operator of the vulnerability scanners has "root" or administrator access to the vulnerable host.

However, vulnerability scanners have some significant weaknesses. Generally, they only identify surface vulnerabilities and are unable to address the overall risk level of a scanned network. Although the scan process itself is highly automated, vulnerability scanners can have a high false positive error rate (reporting vulnerabilities when none exist). This means an individual with expertise in networking and operating system security and in administration must interpret the results.

Since vulnerability scanners require more information than port scanners to reliably identify the vulnerabilities on a host, vulnerability scanners tend to generate significantly more network traffic than port scanners. This may have a negative impact on the hosts or network being scanned or network segments through which scanning traffic is traversing. Many vulnerability scanners also include tests for denial of service (DoS) attacks that, in the hands of an inexperienced tester, can have a considerable negative impact on scanned hosts.

Another significant limitation of vulnerability scanners is that they rely on constant updating of the vulnerability database in order to recognize the latest vulnerabilities. Before running any scanner, organizations should install the latest updates to its vulnerability database. Some vulnerability scanner databases are updated more regularly than others. The frequency of updates should be a major consideration when choosing a vulnerability scanner.

Vulnerability scanners are better at detecting well-known vulnerabilities than the more esoteric ones, primarily because it is difficult to incorporate all known vulnerabilities in a timely manner. Also, manufacturers of these products keep the speed of their scanners high (more vulnerabilities detected requires more tests which slows the overall scanning process).

Vulnerability scanners provide the following capabilities:

- Identifying active hosts on network
- Identifying active and vulnerable services (ports) on hosts.
- Identifying applications and banner grabbing.
- Identifying operating systems.

---

would generally not recognize the danger of the combined vulnerabilities and thus would assign a low risk to them leaving the network administrator with a false sense of confidence in his or her security measures. The reliable way to identify the risk of vulnerabilities in aggregate is through penetration testing.

[11] NIST maintains a database of vulnerability and related patch information at http://icat.nist.gov. This database uses the Common Vulnerabilities and Exposures (CVE) vulnerability identification scheme in use by other databases and vendors.

- Identifying vulnerabilities associated with discovered operating systems and applications.
- Identifying misconfigured settings.
- Testing compliance with host application usage/security policies.
- Establishing a foundation for penetration testing.

Vulnerability scanners can be of two types: network-based scanners and host-based scanners. Network-based scanners are used primarily for mapping an organization's network and identifying open ports and related vulnerabilities. In most cases, these scanners are not limited by the operating system of targeted systems. The scanners can be installed on a single system on the network and can quickly locate and test numerous hosts. Host-based scanners have to be installed on each host to be tested and are used primarily to identify specific host operating system and application misconfigurations and vulnerabilities. Because host-based scanners are able to detect vulnerabilities at a higher degree of detail than network-based scanners, they usually require not only host (local) access but also a "root" or administrative account. Some host-based scanners offer the capability of repairing misconfigurations.

Organizations should conduct vulnerability scanning to validate that operating systems and major applications are up to date on security patches and software version. Vulnerability scanning is a somewhat labor-intensive activity that requires a high degree of human involvement in interpreting the results. It may also disrupt network operations by taking up bandwidth and slowing response times. However, vulnerability scanning is extremely important for ensuring that vulnerabilities are mitigated before they are discovered and exploited by adversaries. Vulnerability scanning should be conducted at least quarterly to semi-annually. Highly critical systems such as firewalls, public web servers, and other perimeter points of entry should be scanned nearly continuously. It is also recommended that since no vulnerability scanner can detect all vulnerabilities, more than one should be used. A common practice is to use a commercially available scanner and a freeware scanner[12].

Vulnerability scanning results should be documented and discovered deficiencies corrected. The following corrective actions may be necessary as a result of vulnerability scanning:

- Upgrade or patch vulnerable systems to mitigate identified vulnerabilities as appropriate.
- Deploy mitigating measures (technical or procedural) if the system cannot be immediately patched (e.g., operating system upgrade will make the application running on top of the operating system inoperable), in order to minimize the probability of this system being compromised.
- Improve configuration management program and procedures to ensure that systems are upgraded routinely.
- Assign a staff member to monitor vulnerability alerts and mailing lists, examine their applicability to the organization's environment and initiate appropriate system changes.
- Modify the organization's security policies, architecture, or other documentation to ensure that security practices include timely system updates and upgrades.

Network and host-based vulnerability scanners are available for free or for a fee. Appendix C contains a list of readily available vulnerability scanning tools.

[12] This mirrors common anti-virus practices, which are to use different products on the desktop versus the email server so that the deficiencies of one may be compensated for by the other.

3.4 Password Cracking

Password cracking programs can be used to identify weak passwords. Password cracking verifies that users are employing sufficiently strong passwords. Passwords are generally stored and transmitted in an encrypted form called a hash. When a user logs on to a computer/system and enters a password, a hash is generated and compared to a stored hash. If the entered and the stored hashes match, the user is authenticated.

During a penetration test or a real attack, password cracking employs captured password hashes. Passwords hashes can be intercepted when they are transmitted across the network (using a network sniffer) or they can be retrieved from the targeted system. The latter generally requires administrative or "root" access on the target system.

Once the hashes are obtained, an automated password cracker rapidly generates hashes until a match is found. The fastest method for generating hashes is a *dictionary attack* that uses all words in a dictionary or text file. There are many dictionaries available on the Internet that cover most major and minor languages, names, popular television shows, etc. So any "dictionary" word no matter how obscure is weak.

Another method of cracking is called a *hybrid attack*, which builds on the dictionary method by adding numeric and symbolic characters to dictionary words. Depending on the password cracker being used, this type of attack will try a number of variations. The attack tries common substitutes of characters and numbers for letters (e.g., p@ssword and h4ckme). Some will also try adding characters and numbers to the beginning and end of dictionary words (e.g., password99, password$%, etc.).

The most powerful password-cracking method is called the *brute force* method. Although brute force can take a long time, it usually takes far less time than most password policies specify for password changing. Consequently, passwords found during brute force attacks are still too weak. Brute force randomly generates passwords and their associated hashes. However since there are so many possibilities it can take months to crack a password. Theoretically all passwords are "crackable" from a brute force attack given enough time and processing power. Penetration testers and attackers often have multiple machines to which they can spread the task of cracking password. Multiple processors greatly shorten the length of time required to crack strong passwords.

A strong Linux/Unix password is one that is long (greater than 10 characters at least) and complex (contains both upper and lower case letters, special characters and numbers). Creating a strong Windows password is somewhat more complicated. Versions of Windows prior to Windows 2000 use LanMan password hashes, which have several associated weaknesses. First, LanMan is not case sensitive, all alphabetic characters are converted to uppercase. This effectively reduces the number of different combinations a password cracker has to try. Second, all LanMan passwords are stored as two 7 character hashes. Passwords that are exactly 14 characters long will be split into two 7 character hashes. Password less than 14 characters will be padded up to 14 characters. The splitting of the hash into two causes LanMan passwords to be less resistant to password cracking.[13] See Appendix D for an example of how to use the Windows password cracker, L0pht Crack.

[13] Due to the mathematics of password cracking, two 7 character hashes are significantly easier to crack than one 14 character hash. To ensure the appropriate strength, passwords for versions prior to Windows 2000 should be either 7 or 14 characters long, include upper and lowercase, and include numbers and characters. For particularly sensitive accounts, use extended characters ASCII characters (these are NOT represented on a standard keyboard). These characters are entered by hitting the Alt key and a sequence of numbers on the numeric keypad (e.g. Alt + 0174 = ®). These characters and their associated keyboard sequences can be identified using the Windows Character Map application.

Password crackers should be run on the system on a monthly basis or even continuously to ensure correct password composition throughout an organization. The following actions can be taken if an unacceptably high number of passwords can be cracked:[14]

- If the cracked passwords were selected according to policy, the policy should be modified to reduce the percentage of crackable passwords. If such policy modification would lead to users writing down their passwords because they are difficult to memorize, an organization should consider replacing password authentication with another form of authentication.
- If cracked passwords were not selected according to policy, the users should be educated on possible impacts of weak password selections. If such violations by the same users are persistent, management should consider additional steps (additional training, password management software to enforce better choices, deny access, etc.) to gain user compliance. Many server platforms also allow the system administrator to set minimum password length and complexity.

On systems that support password filters, the filters should be set so as to force the use of strong passwords, and this may reduce or even the need to perform password cracking. Passwords, no matter how strong, often are sent in the clear over networks; thus organizations should be moving towards the use of stronger forms of authentication.

3.5 Log Reviews

Various system logs can be used to identify deviations from the organization's security policy, including firewall logs, IDS logs, server logs, and any other logs that are collecting audit data on systems and networks. While not traditionally considered a testing activity, log review and analysis can provide a dynamic picture of ongoing system activities that can be compared with the intent and content of the security policy. Essentially, audit logs can be used to validate that the system is operating according to policies.

For example, if an IDS sensor is placed behind the firewall (within the enclave), its logs can be used to examine the service requests and communications that are allowed into the network by the firewall. If this sensor registers unauthorized activities beyond the firewall, it indicates that the firewall is no longer configured securely and a backdoor exists on the network.

Snort is a free IDS sensor with ample support. It is a network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. Snort can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI (Common Gateway Interface) attacks, SMB (System Message Block) probes, and OS fingerprinting attempts. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that uses a modular plugin architecture. Snort has a real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a Unix socket, or WinPopup messages to Windows clients using Samba's smbclient. Snort has three primary uses. It can be used as a straight packet sniffer like tcpdump, a packet logger (useful for network traffic debugging, etc), or as a complete network intrusion detection system.

Manual audit log review is extremely cumbersome and time consuming. Automated audit tools provide a

[14] On many systems, especially those exposed to the Internet, even one cracked password should be considered unacceptable. Attackers are extremely proficient at escalating privilege once they have an access to a system, therefore in many instances a single cracked password for a (or, even worse, an unpassworded) guest account can be enough to compromise the entire system. In addition it should be considered unacceptable if any administrator or root level password is compromised.

means to significantly reduce the required review time and to generate reports (predefined and customized) that summarize the log contents to a set of specific activities. It is critical that any filters applied to the logs filter out what is unwanted and pass everything else.

Log reviews should be conducted very frequently, if not daily, on major servers and firewalls. Again, using log-reduction tools will assist system administrators greatly in identifying problems and suspicious activity. For the specific purpose of testing implementation of required security configurations, once a month may be sufficient with the exception of on demand reviews resulting from major system upgrades that require validation. The following actions can be taken if a system is not configured according to policies:

- Remove vulnerable services if they are not needed.
- Reconfigure the system as required to reduce the chance of compromise.
- Change firewall policy to limit access to the vulnerable system or service.
- Change firewall policy to limit accesses from the IP subnet that is the source of compromise.

3.6 File Integrity Checkers

A file integrity checker computes and stores a checksum for every guarded file and establishes a database of file checksums. It provides a tool for the system administrator to recognize changes to files, particularly unauthorized changes. Stored checksums should be recomputed regularly to test the current value against the stored value to identify any file modifications. A file integrity checker capability is usually included with any commercial host-based intrusion detection system.

An integrity checker is a useful tool that does not require a high degree of human interaction, but it needs to be used carefully to ensure that it is effective. A file integrity checker requires a system that is known to be secure to create the initial reference database. Otherwise, cryptographic hashes of a compromised system may be created, an event that can create a false sense of security for the tester. The reference database should be stored off-line so that attacks cannot compromise the system and hide their tracks by modifying the database. A file integrity checker can also generate false positive alarms. Each file update and system patch implementation changes the file and will, therefore, require an update of the checksum database. As a result, keeping the database up-to-date may be difficult. However, even if the integrity checker is run only once (when the system is first installed), it can still be a useful activity for determining which files have been modified in case of a suspected compromise. Finally, attackers have demonstrated the ability to modify a file in ways the commonly used 32-bit Cyclic Redundancy Check (CRC) checksum could not detect. Therefore, stronger checksums such as the SHA-1 are recommended to ensure the integrity of data that is stored in the checksum database.

Integrity checkers should be run daily on a selection of system files that would be affected by a compromise. Integrity checkers should also be used when a compromise is suspected for determining the extent of possible damage. If an integrity checker detects unauthorized system file modifications, the possibility of a security incident should be considered and investigated according to the organization's incident response and reporting policy, and its procedures. See Appendix C for an example in using the LANguard freeware file integrity checkers.

3.7 Virus Detectors

All organizations are at risk of "contracting" computer viruses, Trojans and worms[15] if they are connected to the Internet, or use removable media (e.g., floppy disks and CD-ROMs), or use shareware/freeware software. The impact of a virus, Trojan, or worm can be as harmless as a pop-up message on a computer screen, or as destructive as deleting all the files on a hard drive. With any malicious code, there is also the risk of exposing or destroying sensitive or confidential information.

There are two primary types of anti-virus programs available: those that are installed on the network infrastructure and those that are installed on end-user machines. Each has advantages and disadvantages, but the use of both types of programs is generally required for the highest level of security.

The virus detector installed on the network infrastructure is usually installed on mail servers or in conjunction with firewalls at the network border of an organization. Server based virus detection programs can detect viruses before they enter the network or before users download their e-mail. Another advantage of server based virus detection is that all virus detectors require frequent updating to remain effective. This is much easier to accomplish on the server-based programs due to their limited number relative to client hosts.

The other type of virus detection software is installed on end-user machines. This software detects malicious code in e-mails, floppies, hard disks, documents and the like but only for the local host. The software also sometimes detects malicious code from web sites. This type of virus detection program has less impact on network performance but generally relies on end-users to update their signatures, a practice that is not always reliable. Most anti-virus software is now able to automatically update the list of virus signatures.

No matter what type of virus detection program is being used, it cannot offer its full protection unless it has an up-to-date virus identification database (sometimes called virus signatures) that allow it to recognize all viruses. If the virus signature database is not up-to-date, it usually will not detect a new virus. To detect viruses, anti-virus software compares file contents with the known computer virus signatures, identifies infected files, quarantines and repairs them if possible, or deletes them if not. More sophisticated programs also look for virus-like activity in an attempt to identify new or mutated viruses that would not be recognized by the current virus detection database. While not perfect, this system can provide an additional layer of protection with the cost of some false positives.

Viruses and other malicious code, such as worms and Trojans, can be enormously destructive to a computer system. The most important aspect of virus detection software is frequent regular updates of virus definition files and on-demand updates when a major virus is known to be spreading throughout the Internet. When the database is updated frequently, more viruses will be detected by the anti-virus software. If these preliminary steps are taken, the chances of a major virus infection are minimized. The following steps are recommended:

- Virus definition files should be updated at least weekly and whenever a major outbreak of a new virus occurs.
- The anti-virus software should be configured to run continuously in the background and use heuristics, if available to look for viruses.
- After the virus definition files are updated, a full system scan should be performed.

[15] These are all examples of malicious computer code that are sometimes collectively referred to as viruses even though they infect and propagate quite differently.

3.8 War Dialing

In a well-configured network, unauthorized modems are often an overlooked vulnerability. These unauthorized modems provide a means to bypass most or all of the security measures in place. There are several software packages available (see Appendix C) that allow attackers and network administrators to dial large blocks of phone numbers in search of available modems. This process is called war dialing. A computer with four modems can dial 10,000 numbers in a matter of days. Certain war dialers will even attempt some limited automatic hacking when a modem is discovered. All will provide a report on the "discovered" numbers with modems.

War dialing should be conducted at least annually and performed after-hours to limit potential disruption to employees and the organization's phone system (this has to be balanced with the possibility that modems may be turned off after hours and, therefore, will not be detected). The check should include all numbers that belong to an organization, except those that could be impacted negatively by receiving a large number of calls (e.g., 24-hour operation centers, emergency numbers, etc.). Most war dialing software allows the tester to exempt particular numbers from the calling list.

If any unauthorized modems are identified, they should be investigated and removed, if appropriate. Generally the Private Branch Exchange (PBX) administrator should be able to identify the user to whom the number was assigned. If removal is not possible, the PBX should be configured to block inbound calls to the modem. If inbound calls are required, ensure that a strong authentication method is in-place.

Although attacks via the Internet get much publicity, many successful attacks are launched through unauthorized modems. The increase in laptops has exacerbated this problem since most laptops have a modem. A single compromise via an authorized modem could allow an attacker direct and undetected access to a network, avoiding perimeter security.

3.9 Wireless LAN Testing ("War Driving")

Wireless technology is a rapidly growing area of networking. The most popular Wireless LAN protocol is 802.11b, which has serious flaws in its current implementation of WEP, the Wireless Equivalent Privacy protocol. There is further risk because that most 802.11b equipment is configured insecurely in its default configuration (i.e., out of the box). Wireless LANs, which may provide attackers the means to bypass Firewalls and IDS, are rapidly replacing unauthorized modems as the most popular back door into networks[16] (if not placed outside the firewall).

Attackers and other malicious parties now regularly drive around office parks and neighborhoods with laptops equipped with wireless network cards attempting to connect to open access points (this practice is called *war driving*). There are now web sites that publish the locations of discovered wireless networks (e.g., http://www.netstumbler.com). The range for many wireless devices is currently 300-600 feet but this range is increasing as manufacturers introduce new products. Attackers often add larger antennas to their wireless network cards to increase the reception range of their cards.

Unfortunately, a number of security vulnerabilities are associated with the 802.11b networking protocol, making it vulnerable to:

- Insertion attacks,

[16] NIST Special Publication 800-48, Wireless Network Security: 802.11, Bluetooth, and Handheld Devices provides additional information about wireless security. See http://csrc.nist.gov/publications/nistpubs/800-48/NIST_SP_800-48.pdf

- Interception and monitoring of wireless traffic,
- Denial of service, and
- Client to client attacks.

Additional security risks in wireless networks result when access points are configured in the least secure mode out of the box. This makes installation easier, but puts the responsibility for security on the network administrator or user installing the wireless network.

For many organizations, the benefits of wireless networks may outweigh the risks. To operate a relatively secure wireless network, an organization needs to create a wireless policy and broadly disseminate that policy to all employees and contractors. Given the low cost and ease of use, an organization will need to periodically test its networks for unauthorized and/or misconfigured wireless LANs, as well as periodically scan their sites for incoming signals from neighboring wireless LANs. Creating one or more portable computers with wireless network cards and testing tools (see Appendix C) for detecting wireless LANs will assist in this effort. The frequency for testing wireless networks will depend on several factors:

- Physical factors of the location to be tested (e.g., a building located on a secure installation thousands of feet from any public access area will need testing less often than an office located in a busy downtown office district).
- The threat level faced by the organization.
- Organizational control over network resources (e.g., an organization with tight central control over the network may need to test less often than with a very decentralized network support structure).
- The use of more robust security techniques in the network, such as the WPA (Wi-Fi Protected Access) or RSN (Robust Security Network).
- Sensitivity of the data on the organizations network.

As a general guideline, organizations with high risks and threats should test for unauthorized and/or misconfigured wireless LANs on a monthly level or more often. Random audits are also recommended.

3.10 Penetration Testing

Penetration testing is security testing in which evaluators attempt to circumvent the security features of a system based on their understanding of the system design and implementation. The purpose of penetration testing is to identify methods of gaining access to a system by using common tools and techniques used by attackers. Penetration testing should be performed after careful consideration, notification, and planning.

Penetration testing can be an invaluable technique to any organization's information security program. However, it is a very labor-intensive activity and requires great expertise to minimize the risk to targeted systems. At a minimum, it may slow the organization's networks response time due to network scanning and vulnerability scanning. Furthermore, the possibility exists that systems may be damaged in the course of penetration testing and may be rendered inoperable, even though the organization benefits in knowing that the system could have been rendered inoperable by an intruder. Although this risk is mitigated by the use of experienced penetration testers, it can never be fully eliminated.

Since penetration testing is designed to simulate an attack and use tools and techniques that may be restricted by law, federal regulations, and organizational policy, it is imperative to get formal permission for conducting penetration testing prior to starting. This permission, often called the rules of engagement, should include:

- Specific IP addresses/ranges to be tested
- Any restricted hosts (i.e., hosts, systems, subnets, not to be tested)
- A list of acceptable testing techniques (e.g. social engineering, DoS, etc.) and tools (password crackers, network sniffers, etc.)
- Times when testing is to be conducted (e.g., during business hours, after business hours, etc.)
- Identification of a finite period for testing
- IP addresses of the machines from which penetration testing will be conducted so that administrators can differentiate the legitimate penetration testing attacks from actual malicious attacks
- Points of contact for the penetration testing team, the targeted systems, and the networks
- Measures to prevent law enforcement being called with false alarms (created by the testing)
- Handling of information collected by penetration testing team.

Penetration testing can be overt or covert. These two types of penetration testing are commonly referred to as Blue Teaming and Red Teaming. Blue Teaming involves performing a penetration test *with* the knowledge and consent of the organization's IT staff. Red Teaming involves performing a penetration test *without* the knowledge of the organization's IT staff but with full knowledge and permission of the upper management. Some organizations designate a trusted third party for the Red Teaming exercises to ensure that an organization does not take measures associated with the real attack without verifying that an attack is indeed under way (i.e., the activity they are seeing does not originate from an exercise). The trusted third party provides an agent for the testers, the management, and the IT and security staff that mediates the activities and facilitates communications. This type of test is useful for testing not only network security, but also the IT staff's response to perceived security incidents and their knowledge and implementation of the organization's security policy. The Red Teaming may be conducted with or without warning.

Of the two types of penetration tests, Blue Teaming is the least expensive and most frequently used. Red Teaming, because of the stealth requirements, requires more time and expense. To operate in a stealth environment, a Red Team will have to slow its scans and other actions to move below the ability of the target organization’s Intrusion Detection System) and firewall to detect their actions. However, Red Teaming provides a better indication of everyday security of the target organization since system administrators will not be on heightened awareness.

A penetration test can be designed to simulate an inside and/or an outside attack. If both internal and external testing are to be performed, the external testing usually occurs first. With external penetration testing, firewalls usually limit the amount and types of traffic that are allowed into the internal network from external sources. Depending on what protocols are allowed through, initial attacks are generally focused on commonly used and allowed application protocols such as FTP, HTTP, or SMTP and POP.

To simulate an actual external attack, the testers are not provided with any real information about the target environment other than targeted IP address/ranges and they must covertly collect information before the attack. They collect information on the target from public web pages, newsgroups and similar sites. They then use port scanners and vulnerability scanners to identify target hosts. Since they are, most likely, going through a firewall, the amount of information is far less than they would get if operating internally. After identifying hosts on the network that can be reached from the outside, they attempt to compromise one of the hosts. If successful, they then leverage this access to compromise others hosts not generally accessible from outside. This is why penetration testing is an iterative process that leverages minimal access to gain greater access.

An internal penetration test is similar to an external except that the testers are now on the internal network (i.e., behind the firewall) and are granted some level of access to the network (generally as a user but sometimes at a higher level). The penetration testers will then try to gain a greater level of access to the network through privilege escalation. The testers are provided with the information about a network that somebody with their provided privileges would normally have. This is generally as a standard employee although it can also be anything up to and including a system or network administrator depending on the goals of the test.

Penetration testing consists of four phases (see Figure 3.1):

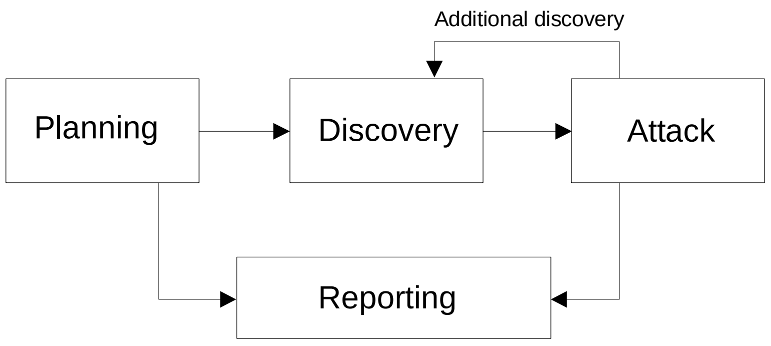


Figure 3.1: Four-Stage Penetration Testing Methodology

In the planning phase, rules are identified, management approval is finalized, and the testing goals are set. The planning phase sets the groundwork for a successful penetration test. No actual testing occurs in the planning phase.

The discovery phase starts the actual testing. Network scanning (port scanning), as described in Section 3.2 is used to identify potential targets. In addition to port scanning, other techniques are commonly used to gather information on the targeted network:

- Domain Name System (DNS) interrogation
- InterNIC (whois) queries
- Search of the target organization’s web server(s) for information

- Search of the organization's Lightweight Directory Access Protocol server(s) (LDAP) for information
- Packet capture (generally only during internal tests)
- NetBIOS enumeration (generally only during internal tests)
- Network Information System ([NIS] generally only during internal tests)
- Banner grabbing

The second part of the discovery phase is vulnerability analysis. During this phase, services, applications, and operating systems of scanned hosts are compared against vulnerability databases (for vulnerability scanners this process is automatic). Generally human testers use their own database or public databases to identify vulnerabilities manually.[17] This manual process is better for identifying new or obscure vulnerabilities, but is much slower than an automated scanner.

Executing an attack is at the heart of any penetration test. This is where previously identified potential vulnerabilities are verified by attempting to exploit them. If an attack is successful, the vulnerability is verified and safeguards are identified to mitigate the associated security exposure. Frequently, exploits[18] that are executed during attack execution do not grant the maximum level of access that can be gained by an attacker. Instead they may result in the testing team learning more about the targeted network and its potential vulnerabilities, or they may induce a change in the state of the security of the targeted network. In either case, additional analysis and testing is required to determine the true level of risk for the network. This is represented in the feedback loop in Figure 3.2 between the Attack and Discovery phase of a penetration test.

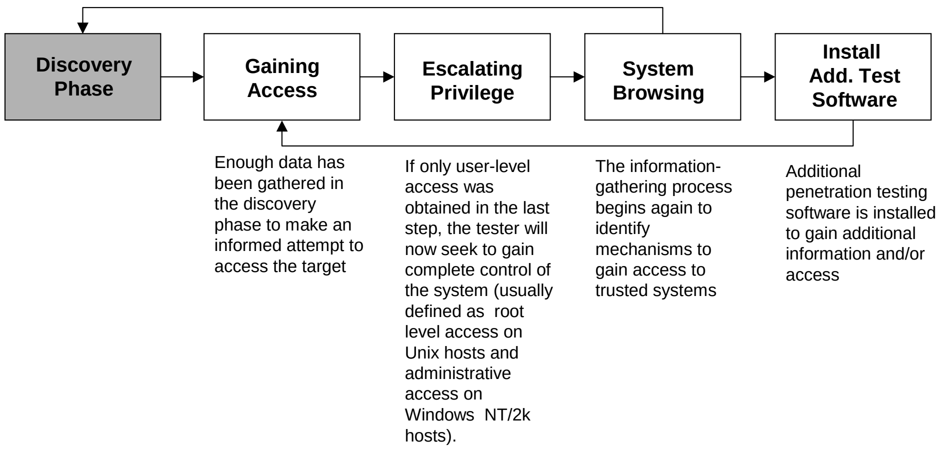

Discovery Phase
Gaining Access
Escalating Privilege
System Browsing
Install Add. Test Software
Enough data has been gathered in the discovery phase to make an informed attempt to access the target
If only user-level access was obtained in the last step, the tester will now seek to gain complete control of the system (usually defined as root level access on Unix hosts and administrative access on Windows NT/2k hosts).
The information-gathering process begins again to identify mechanisms to gain access to trusted systems
Additional penetration testing software is installed to gain additional information and/or access

Figure 3.2: Attack Phase Steps with Loopback to Discovery Phase

[17] Some popular vulnerability databases include: http://icat.nist.gov/icat.cfm, http://cve.mitre.org/ http://www.securityfocus.com/

[18] Exploits are documented methods or programs/scripts that take advantage of vulnerabilities. The same cautions that apply to freeware tools apply to exploit programs/scripts. Many vulnerability databases including www.securityfocus.com provide exploit instructions or code for most identified vulnerabilities. Exploit programs or scripts are actually just specialized tools for exploiting a specific vulnerability.

While vulnerability scanners only check that a vulnerability may exist, the attack phase of a penetration test exploits the vulnerability, confirming its existence. Most vulnerabilities exploited by penetration testing and malicious attackers fall into the following categories:

- **Kernel Flaws**—Kernel code is the core of an operating system. The kernel code enforces the overall security model for the system. Any security flaw that occurs in the kernel puts the entire system in danger.

- **Buffer Overflows**—A buffer overflow occurs when programs do not adequately check input for appropriate length, which is usually a result of poor programming practice. When this occurs, arbitrary code can be introduced into the system and executed with the privileges of the running program. This code often can be run as root on Unix systems and SYSTEM (administrator equivalent) on Windows systems.

- **Symbolic Links**—A symbolic link or symlink is a file that points to another file. Often there are programs that will change the permissions granted to a file. If these programs run with privileged permissions, a user could strategically create symlinks to trick these programs into modifying or listing critical system files.

- **File Descriptor Attacks**—File descriptors are nonnegative integers that the system uses to keep track of files rather than using specific filenames. Certain file descriptors have implied uses. When a privileged program assigns an inappropriate file descriptor, it exposes that file to compromise.

- **Race Conditions**—Race conditions can occur when a program or process has entered into a privileged mode but before the program or process has given up its privileged mode. A user can time an attack to take advantage of this program or process while it is still in the privileged mode. If an attacker successfully manages to compromise the program or process during its privileged state, then the attacker has won the "race." Common race conditions include signal handling and core-file manipulation.

- **File and Directory Permissions**—File and directory permissions control the access users and processes have to files and directories. Appropriate permissions are critical to the security of any system. Poor permissions could allow any number of attacks, including the reading or writing of password files or the addition of hosts to the list of trusted remote hosts

- **Trojans**—Trojan programs can be custom built or could include programs such as BackOrifice, NetBus, and SubSeven. Kernel root kits could also be employed once access is obtained to allow a backdoor into the system at anytime.

- **Social Engineering**—Social engineering is the technique of using persuasion and/or deception to gain access to, or information about, information systems. It is typically implemented through human conversation or other interaction. The usual medium of choice is telephone but can also be e-mail or even face-to-face interaction. Social engineering generally follows two standard approaches. In the first approach the penetration tester poses as a user experiencing difficultly and calls the organization's help desk in order to gain information on the target network or host, obtain a login ID and credentials, or get a password reset. The second approach is to pose as the help desk and call a user in order to get the user to provide his/her user id(s) and password(s). This technique can be extremely effective.

The reporting phase occurs simultaneously with the other three phases of the penetration test (see Figure 3.1). In the planning phase, rules of engagement, test plans and written permission are developed. In the discovery and attack phase, written logs are usually kept and periodic reports are made to system administrators and/or management, as appropriate. Generally, at the end of the test an overall testing report is developed to describe the identified vulnerabilities, provide a risk rating, and to give guidance on the mitigation of the discovered weaknesses.

Penetration testing is important for determining how vulnerable an organization's network is and the level of damage that can occur if the network is compromised. Because of the high cost and potential impact, annual penetration testing may be sufficient. The results of penetration testing should be taken very seriously and discovered vulnerabilities should be mitigated. As soon as they are available, the results should be presented to the organization’s managers.

Corrective measures can include closing discovered and exploited vulnerabilities, modifying an organization's security policies, creating procedures to improve security practices, and conducting security awareness training for personnel to ensure that they understand the implications of poor system configurations and poor security practices. Organizations should consider conducting less labor-intensive testing activities on a regular basis to ensure that they are in compliance with their security policies and are maintaining the required security posture. If an organization performs other tests (e.g., network scanning and vulnerability scanning) regularly between the penetration testing exercises and corrects discovered deficiencies, it will be well prepared for the next penetration testing exercise and for a real attack.

3.11 Post-Testing Actions

For most organizations, testing likely will reveal issues that need to be addressed quickly. How these issues are addressed and mitigated is the most important step in the testing process. The most common root causes and methods for addressing them are provided below.

Lack of (or Poorly Enforced) Organizational Security Policy: Perhaps the single largest contributor to poorly secured systems is the lack of an organizational security policy. A security policy is important as it ensures consistency. Consistency is a critical component of a successful security posture because it leads to predictable behavior. This will make it easier for an organization to maintain secure configurations and will assist in identifying security problems (which often manifest themselves as deviations from predictable, expected behavior). Each organization needs to have a security policy and to communicate that policy to users and administrators. A security policy should generally include:

- Organizational Standards (these usually specify uniform use of specific technologies, parameters and/or procedures)
- Privacy (e.g., whether there is monitoring of email and web use)
- Acceptable use guidelines (e.g., what is acceptable use of organization computing and network resources)
- Roles and responsibilities (for users, administrators, management)
- Accountability (auditing, incident handling)
- Authentication (e.g., passwords, biometrics)
- Availability of resources (redundancy, recovery, backups)
- Compliance (infractions, consequences and penalties).

Misconfiguration: This occurs when a system is not configured in a secure or recommended fashion. There are various actions that can be taken to remedy or minimize the chances of misconfiguration:

- Create a configuration management process (often called a configuration control board) for critical systems and networks. A configuration management process controls the changes made to a system or network and ensures compliance with the organization's policies. The configuration management process should not hinder the timely application of updates and security patches[19].
- Create (or use readily available) configuration checklists. These checklist provide a list of configuration settings that if implemented will provide a more secure system or network. These checklists are available from numerous sources including Federal agencies, vendors and individuals.

Software (Un)Reliability: Many successful attacks exploit errors ("bugs") in the software code used on computers and networks. Organizations can minimize the problems caused by software errors in several ways. For code developed in-house, the proper procedures for code development and testing should implemented to ensure the appropriate level of quality control. The organization will have less control over the quality of the code that is purchased from outside vendors. To mitigate this risk, organizations should regularly check for updates and patches from vendors and apply them in a timely manner. When organizations are considering the purchase of commercially produced software, they should check vulnerability databases (e.g., http://icat.nist.gov) and examine the past performance of the vendor's software (however, past performance may not always be an accurate indicator of future performance).

Failure to Apply Patches: Software has become so complicated that software errors ("bugs") are inevitable. When an error is discovered, the software publisher generally issues a patch to correct or mitigate the error so it cannot be exploited by malicious entities. Unfortunately many administrators do not have the time, resources, or the knowledge to apply patches in a timely manner. This is reflected in estimates that many network intrusions could be avoided by keeping systems up to date with appropriate patches

The results of testing could also show the need to make large-scale changes in the network and security architecture of an organization. For example, the results of penetration testing may show the need for a more layered defense strategy with increased numbers of firewalls. Or, the demands of keeping large numbers of systems up to date with patches may cause an organization to adopt a centralized management scheme, or even a centralized computer purchasing and configuration scheme. Accordingly, an organization can benefit greatly by using the testing results to arrive at a bigger picture of their operating environment, and how that environment could change to make testing easier and to reduce exposures to vulnerabilities.

3.12 General Information Security Principles

When addressing security issues, some general information security principles should be kept in mind, as follows[20],[21]:

[19] See NIST Special Publication 800-40, Procedures for Handling Security Patches, http://csrc.nist.gov/publications/nistpubs/800-40/sp800-40.pdf.

[20] See Curtin, Matt, *Developing Trust: Online Privacy and Security*, November 2001, and Saltzer and Schroeder, *The Protection of Information in Computer Systems*, Volume 63, pages 1278-1308.

[21] For more general information, also see NIST Special Publication 800-14, Generally Accepted Principles and Practices for Securing Information Technology Systems, http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf, and NIST Special

- **Simplicity**—Security mechanisms (and information systems in general) should be as simple as possible. Complexity is at the root of many security issues.
- **Fail-Safe**—If a failure occurs, the system should fail in a secure manner. That is, if a failure occurs, security should still be enforced. It is better to lose functionality than lose security.
- **Complete Mediation—**Rather than providing direct access to information, mediators that enforce access policy should be employed. Common examples include files system permissions, web proxies and mail gateways.
- **Open Design—**System security should not depend on the secrecy of the implementation or it components. "Security through obscurity" does not work.
- **Separation of Privilege—**Functions, to the degree possible, should be separate and provide as much granularity as possible. The concept can apply to both systems and operators/users. In the case of system operators and users, roles should be as separate as possible. For example if resources allow, the role of system administrator should be separate from that of the security administrator.
- **Psychological Acceptability—**Users should understand the necessity of security. This can be provided through training and education. In addition, the security mechanisms in place should present users with sensible options that will give them the usability they require on a daily basis. If users find the security mechanisms too cumbersome, they find ways to work around or compromise them. An example of this is using random passwords that are very strong but difficult to remember; users may write them down or looks for methods to circumvent the policy.
- **Layered Defense—**Organizations should understand that any single security mechanism is generally insufficient. Security mechanisms (defenses) need to be layered so that compromise of a single security mechanism is insufficient to compromise a host or network. There is no "magic bullet" for information system security.
- **Compromise Recording—**When systems and networks are compromised, records or logs of that compromise should be created. This information can assist in securing the network and host after the compromise and assist in identifying the methods and exploits used by the attacker. This information can be used to better secure the host or network in the future. In addition, this can assist organizations in identifying and prosecuting attackers.

A number of NIST documents can be helpful in reconfiguring systems. The following documents may be particularly useful:

- **SP 800-48, Wireless Network Security: 802.11, Bluetooth, and Handheld Devices,** http://csrc.nist.gov/publications/nistpubs/800-48/NIST_SP_800-48.pdf
- **SP 800-46, Security for Telecommuting and Broadband Communications,** http://csrc.nist.gov/publications/nistpubs/800-48/NIST_SP_800-46.pdf
- **SP 800-45, Guidelines on Electronic Mail Security,** http://csrc.nist.gov/publications/nistpubs/800-45/sp800-45.pdf

Publication 800-27, Engineering Principles for Information Technology Security (A Baseline for Achieving Security), http://csrc.nist.gov/publications/nistpubs/800-27/sp800-27.pdf.

- **SP 800-44, Guidelines on Securing Public Web Servers,** http://csrc.nist.gov/publications/nistpubs/800-44/sp800-44.pdf
- **SP 800-43, Systems Administration Guidance for Windows 2000 Professional,** http://csrc.nist.gov/itsec/guidance_W2Kpro.html
- **SP 800-41, Guidelines on Firewalls and Firewall Policy,** http://csrc.nist.gov/publications/nistpubs/800-41/sp800-41.pdf
- **SP 800-40, Procedures for Handling Security Patches,** http://csrc.nist.gov/publications/nistpubs/800-40/sp800-40.pdf.

3.13 Summary Comparisons of Network testing Techniques

Table 3.1 and Table 3.2 provide a comparison of the testing techniques discussed above.

Type of Test	Strengths	Weaknesses
Network Scanning	• Fast (as compared to vulnerability scanners or penetration testing) • Efficiently scans hosts, depending on number of hosts in network • Many excellent freeware tools available • Highly automated (for scanning component) • Low cost	• Does not directly identify known vulnerabilities (although will identify commonly use Trojan ports [e.g., 31337, 12345, etc]) • Generally used as a prelude to penetration testing not as final test • Requires significant expertise to interpret results
Vulnerability Scanning	• Can be fairly fast depending on number of hosts scanned • Some freeware tools available • Highly automated (for scanning) • Identifies known vulnerabilities • Often provides advice on mitigating discovered vulnerabilities • High cost (commercial scanners) to low (freeware scanners) • Easy to run on a regular basis	• Has high false positive rate • Generates large amount of traffic aimed at a specific host (which can cause the host to crash or lead to a temporary denial of service) • Not stealthy (e.g., easily detected by IDS, firewall and even end-users [although this may be useful in testing the response of staff and altering mechanisms]) • Can be dangerous in the hands of a novice (particularly DoS attacks) • Often misses latest vulnerabilities • Identifies only surface vulnerabilities
Penetration Testing	• Tests network using the methodologies and tools that attackers employ • Verifies vulnerabilities • Goes beyond surface vulnerabilities and demonstrates how these vulnerabilities can be exploited iteratively to gain greater access • Demonstrates that vulnerabilities are not purely theoretical • Can provide the realism and evidence needed to address security issues • Social engineering allows for testing of procedures and the human element network security	• Requires great expertise • Very labor intensive • Slow, target hosts may take hours/days to "crack" • Due to time required not all hosts on medium or large sized networks will be tested individually • Dangerous when conducted by inexperienced testers • Certain tools and techniques may be banned or controlled by agency regulations (e.g., network sniffers, password crackers, etc.) • Expensive • Can be organizationally disruptive

Type of Test	Strengths	Weaknesses
Password Cracking	• Quickly identifies weak passwords • Provides clear demonstration of password strength or weakness • Easily implemented • Low cost	• Potential for abuse • Certain organizations restrict use
Log Reviews	• Provides excellent information • Only data source that provides historical information	• Cumbersome to manually review • Automated tools not perfect can filter out important information
File Integrity Checkers	• Reliable method of determining whether a host has been compromised • Highly automated • Low cost	• Does not detect any compromise prior to installation • Checksums need to be updated when system is updated • Checksums need to be protected (e.g., read only CD-Rom) because they provide no protection if they can be modified by an attacker
Virus Detectors	• Excellent at preventing and removing viruses • Low/Medium cost	• Require constant updates to be effective • Some false positive issues • Ability to react to new, fast replicating viruses is often limited
War Dialing	• Effective way to identify unauthorized modems	• Legal and regulatory issues especially if using public switched network • Slow
War Driving	• Effective way to identify unauthorized wireless access points	• Possible legal issues if other organization's signals are intercepted • Requires some expertise in computing, wireless networking and radio engineering

Table 3.1: Comparison of Testing Procedures

Table 3.2 describes a general schedule and list of evaluation factors for testing categories. Category 1 systems are those sensitive systems that provide security for the organization or that provide other critical functions. These systems often include

- Firewalls, routers, and perimeter defense systems such as for intrusion detection,
- Public access systems such as web and email servers,
- DNS and directory servers, and other internal systems that would likely be intruder targets.

Category 2 systems are generally all other systems, i.e., those systems that are protected by firewalls, etc., but that still must be tested periodically.

Test Type	Category 1 Frequency	Category 2 Frequency	Benefit
Network Scanning	Continuously to Quarterly	Semi-Annually	• Enumerates the network structure and determines the set of active hosts, and associated software • Identifies unauthorized hosts connected to a network • Identifies open ports • Identifies unauthorized services
Vulnerability Scanning	Quarterly or bi-monthly (more often for certain high risk systems), when the vulnerability database is updated	Semi-Annually	• Enumerates the network structure and determines the set of active hosts, and associated software • Identifies a target set of computers to focus vulnerability analysis • Identifies potential vulnerabilities on the target set • Validates that operating systems and major applications are up to date with security patches and software versions
Penetration Testing	Annually	Annually	• Determines how vulnerable an organization's network is to penetration and the level of damage that can be incurred • Tests IT staff's response to perceived security incidents and their knowledge of and implementation of the organization's security policy and system's security requirements
Password Cracking	Continuously to same frequency as expiration policy	Same frequency as expiration policy	• Verifies that the policy is effective in producing passwords that are more or less difficult to break • Verifies that users select passwords that are compliant with the organization's security policy
Log Reviews	Daily for critical systems, e.g., firewalls	Weekly	• Validates that the system is operating according to policies
Integrity Checkers	Monthly and in case of suspected incident	Monthly	• Detects unauthorized file modifications
Virus Detectors	Weekly or as required	Weekly or as required	• Detects and deletes viruses before successful installation on the system
War Dialing	Annually	Annually	• Detects unauthorized modems and prevents unauthorized access to a protected network
War Driving	Continuously to weekly	Semi-annually	• Detects unauthorized wireless access points and prevents unauthorized access to a protected network

Table 3.2: Summarized Evaluation and Frequency Factors

4. Deployment Strategies for Security Testing

The goal of security testing is to maximize the benefit to the organization as a whole. Deciding on the types and frequency of testing during the operational and maintenance phase (both for minimum and comprehensive testing) involves a prioritization process based on the security category, cost of conducting tests, and benefit to the overall organization's systems. While deciding what to test for during the implementation phase involves a single system, the same decision during the operational and maintenance phase is more complicated. To maximize the value of testing, the prioritization process should consider the interconnectivity of systems. The CIO or a senior IT manager should be involved in the prioritization process to ensure that the organizational perspective is considered. This section describes the prioritization process that can be used.

4.1 Determine the Security Category of the Information System

FIPS Publication 199, *Standards for Security Categorization of Federal Information and Information Systems (*Pre-publication final), December 2003, provides standards for determining the security category of an organization's information systems which can be helpful in developing a priority ranking of those systems for testing purposes. FIPS Publication 199 security categories are based on the potential impact on an organization should certain events occur which jeopardize the information systems needed by the organization to accomplish its assigned mission, protect its assets, fulfill its legal responsibilities, maintain its day-to-day functions, and protect individuals. Security categories are to be used in conjunction with vulnerability and threat information in assessing the risk to an organization by operating an information system.[22] FIPS Publication 199 defines three levels of *potential impact* on organizations or individuals should there be a breach of security (i.e., a loss of confidentiality, integrity, or availability).

The *potential impact* is **LOW** if—

– The loss of confidentiality, integrity, or availability could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.[23] A limited adverse effect means that, for example, the loss of confidentiality, integrity, or availability might: (i) cause a degradation in mission capability to an extent and duration that the organization is able to perform its primary functions, but the effectiveness of the functions is noticeably reduced; (ii) result in minor damage to organizational assets; (iii) result in minor financial loss; or (iv) result in minor harm to individuals.

The *potential impact* is **MODERATE** if—

– The loss of confidentiality, integrity, or availability could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals. A serious adverse effect means that, for example, the loss of confidentiality, integrity, or availability might: (i) cause a significant degradation in mission capability to an extent and duration that the organization is able to perform its primary functions, but the effectiveness of the functions is significantly reduced; (ii) result in significant damage to organizational assets; (iii) result in significant financial loss; or (iv) result in significant harm to individuals that does not involve loss of life or serious life threatening injuries.

The *potential impact* is **HIGH** if—

[22] NIST Special Publication 800-30, *Risk Management Guide*, provides guidance on conducting a risk assessment See http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[23] Adverse effects on individuals may include, but are not limited to, loss of the privacy to which individuals are entitled under law.

– The loss of confidentiality, integrity, or availability could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals. A severe or catastrophic adverse effect means that, for example, the loss of confidentiality, integrity, or availability might: (i) cause a severe degradation in or loss of mission capability to an extent and duration that the organization is not able to perform one or more of its primary functions; (ii) result in major damage to organizational assets; (iii) result in major financial loss; or (iv) result in severe or catastrophic harm to individuals involving loss of life or serious life threatening injuries.

4.2 Determine Cost of Performing Each Test Type per System

When system security categories are determined, the cost of conducting each test should be ascertained. The cost depends on a number of factors:

- Size of the system to be tested—Local Area Network (LAN), Wide Area Network (WAN), single database, or major application.
- Complexity of the system to be tested—testing a network of a large organization with a heterogeneous operating system environment will be more costly.
- Level of human interaction required for each test.
- The feasibility of selecting a sample for conducting the tests and the size of the sample—while it may not make sense to conduct network scanning on a sample of network hosts, selecting sample hosts for penetration testing is entirely possible.

For each system, the costs of conducting each type of test should be quantified.

4.3 Identify Benefits of Each Test Type per System

To ensure that the cost of testing does not exceed its value to the organization, the benefits of conducting the tests should be identified and qualified or quantified as much as possible. While the overall benefit of testing is in identifying vulnerabilities before an attacker exploits them, but there are other benefits as well. The following are examples of factors that should be considered in identifying the benefits of testing:

- Value of gained knowledge about systems and networks that was absent prior to testing—improved knowledge facilitates better organizational control of its assets
- Significantly decreased probability of successful intrusion or business disruption—by testing and correcting discovered deficiencies, an organization reduces the number of vulnerabilities that can be exploited.

4.4 Prioritize Systems for Testing

The results of Steps 1-3 should be evaluated and ranked to prioritize the identified systems for security testing. This analysis should yield a list of systems ranked by security category, cost of testing, and benefit. The list will include required resources (costs) for conducting each type of test for each system under consideration. The starting point for determining minimum required resources should be minimum testing for those systems with the highest level of impact. The resources available for security testing should then be identified and compared with required resources. If the gap between “required” and “available” resources does not cover minimum testing for highest impact systems, as defined in the list of prioritized systems, additional resources should be sought to conduct minimum security testing. Cost of testing as calculated, will provide quantitative evidence of why more resources are required. After the funding is identified for the most critical systems, the lower priority items may be tested with less frequency and in descending order. The result of this final step is a prioritized list of highest impact systems that will be tested with associated testing techniques and frequency.

Appendix A. Terminology

CGI	Common Gateway Interface
CIO	Chief Information Officer
CRC	Cyclic Redundancy Check
DNS	Domain Name System
DoS	Denial of Service
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IIS	Microsoft's Internet Information (Web) Server
InterNIC	Internet Network Information Center
ISSM	Information Systems Security Manager
ISSO	Information Systems Security Officer
IT	Information Technology
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
NETBIOS	Network Basic Input/Output System
NIS	Network Information System
OS	Operating System
PBX	Private Branch Exchange
POP	Post Office Protocol
POTS	Plain Old Telephone System
RFC	Request For Comments
SANS	System Administration, Networking, and Security
SMB	Simple Message Block
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
ST&E	Security Testing and Evaluation
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network
WU-FTPD	Washington University FTP server

Appendix B. References

D. Brent Chapman and Elizabeth D. Zwicky, *Building Internet Firewalls*, 1995.

Federal CIO Council, *How To Eliminate The Ten Most Critical Internet Security Threats*, November 2000.

Hatch, Brian, et al., *Hacking Exposed: Linux*, 2001.

Information Security Institute (ISI) Swiss Army Knife Reference, *Resource for Security & Audit Professionals*, Michael Ira Sobol (MIS) Training Institute.

MIS Training Institute, *TCP/IP Network Security and Vulnerability Testing.*

NIST, *ITL Bulletin, Advising Users On Information Technology*, May 1999.

NIST, *ITL Bulletin, Computer Attacks: What They Are and How to Defend Against Them*, May 1999.

NIST, FIPS Pub 199 (Draft), *Standards for Security Categorization of Federal Information and Information Systems*, September, 2003.

NIST, SP 800-14, *Generally Accepted Principles and Practices for Securing Information Technology Systems*, September 1996

NIST, SP 800-41, *Guideline on Firewalls and Firewall Policy*, January 2002.

NIST, SP 800-61 (Draft), *Computer Security Incident Handling Guide*, September, 2003.

National Information System Security Glossary, NSTISSI No. 4009, January 1999.

Office of Management and Budget, Circular A-130, February 1996.

Scambray, Joel, et al., *Hacking Exposed: Second Edition*, 2001.

System Administration, Networking, and Security (SANS) Institute, *SANS Security Alert*, May 2000.

SANS Institute, *SANS Snap: Computer and Hacker Exploits – Step by Step.*

SANS Institute, *SANS Snap: Intrusion Detection – The Big Picture.*

MIS Training Institute, *Staying Ahead of the Hackers: Network Vulnerability Testing*.

Stevens, W. Richard, *TCP/IP Illustrated, Volume 1:The Protocols*, 1994.

Appendix C. Common Testing Tools

C.1 File Integrity Checkers

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
Aide	Unix and Linux	http://www.cs.tut.fi/~rammer/aide.html	✓		Free
Description	*AIDE (Advanced Intrusion Detection Environment) is a free replacement for Tripwire. It does file integrity checking and supports a number a large number of Unix and Linux platforms.*				
LANGuard	Windows 2000/NT	http://www.gfi.com/languard/		✓	Free
Description	*LANguard File Integrity Checker is a utility that provides intrusion detection by checking whether files have been changed, added or deleted on a Windows 2000/NT system.*				
Tripwire	Windows, Unix, Linux, and Routers	http://www.tripwiresecurity.com/	✓	✓	Free for Unix
Description	*Tripwire monitors file changes, verifies integrity, and notifies the administrator of any violations of data on network hosts.*				

Table C.1: File Integrity Checker Tools

C.2 Network Sniffers

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
Dsniff	Unix sniffer	http://www.monkey.org/~dugsong/dsniff/	✓		Free
Description	*Dsniff is a collection of tools for network auditing and penetration testing. Dsniff, filesnarf, mailsnarf, msgsnarf, urlsnarf, and webspy passively monitor a network for interesting data (passwords, e-mail, files, etc.). Arpspoof, dnsspoof, and macof facilitate the interception of network traffic normally unavailable to an attacker (e.g, due to layer-2 switching). Sshmitm and webmitm implement active monkey-in-the-middle attacks against redirected SSH and HTTPS sessions by exploiting weak bindings in ad-hoc PKIs.*				
Ethereal	Unix/Windows sniffer with GUI	http://www.ethereal.com/	✓	✓	Free
Description	*Ethereal is a free network protocol analyzer for Unix and Windows. It allows users to examine data from a live network or from a capture file on disk. It can interactively browse the capture data, viewing summary and detail information for each packet. Ethereal has several powerful features, including a rich display filter language and the ability to view the reconstructed stream of a TCP session and parse an 802.11 packet.*				
Sniffit	Unix sniffer	http://reptile.rug.ac.be/~coder/sniffit/sniffit.html http://www.symbolic.it/Prodotti/sniffit.html (Windows)	✓	✓	Free
Description	*A freeware general-purpose sniffer for various versions of Linux, Unix, and Windows.*				
Snort	Unix sniffer/IDS	http://www.snort.org	✓	✓	Free
Description	*A freeware lightweight IDS and general-purpose sniffer for various versions of Linux, Unix and Windows.*				
TCPDump	Unix sniffer	http://www-nrg.ee.lbl.gov/	✓		Free
Description	*A freeware general-purpose sniffer for various versions of Linux and Unix.*				
WinDump	Windows sniffer	http://netgroup-serv.polito.it/windump/		✓	Free
Description	*A freeware Windows general-purpose sniffer based on TCPDump.*				

Table C.2: Network Sniffer Tools

C.3 Password Crackers

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
Crack 5	Unix password cracker	http://www.crypticide.org/users/alecm/	✓		Free
Description	*Crack is a password guessing program that is designed to quickly locate insecurities in Unix (or other) password files by scanning the contents of a password file, looking for users who have misguidedly chosen a weak login password.*				
IMP 2.0	Novell Netware password cracker	http://www.wastelands.gen.nz		✓	Free
Description	*Imp is a NetWare password cracking utility with a GUI (Win95/NT). It loads account information directly from NDS or Bindery files and allows the user to attempt to compromise the account passwords with various attack methods.*				
John the Ripper	Windows and Unix password cracker	http://www.openwall.com/john/	✓	✓	Free
Description	*John the Ripper is a fast password cracker, currently available for many flavors of Unix, DOS, Win32, and BeOS. Its primary purpose is to detect weak Unix passwords, but a number of other hash types are supported as well.*				
L0pht Crack	Windows password cracker	http://www.securityfocus.com/tools/1005		✓	$
Description	*A password cracking utility for Windows NT, 2000 and XP.*				
Nwpcrack	Novell Netware password cracker	http://ftp.cerias.purdue.edu/pub/tools/novell/		✓	Free
Description	*A password cracking utility for Novell Netware.*				

Table C.3: Password Cracking Tools

C.4 Scanning and Enumeration Tools

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
DUMPSec	Windows enumeration tool	http://www.systemtools.com		✓	Free
Description	*DumpSec is a security auditing program for Microsoft Windows. It dumps the permissions (DACLs) and audit settings (SACLs) for the file system, registry, printers and shares in a concise, readable listbox format, so that holes in system security are readily apparent. DumpSec also dumps user, group, and replication information.*				
Firewalk	Firewall filter rule mapper	http://www.packetfactory.net/firewalk/	✓		Free
Description	*Firewalking is a technique that employs traceroute-like techniques to analyze IP packet responses to determine gateway ACL filters and map networks. Firewalk the tool employs the technique to determine the filter rules in place on a packet forwarding device.*				
Fscan	Port scanner	http://www.foundstone.com/		✓	Free
Description	*FScan is a command-line port scanner. It will scan for both TCP and UDP ports.*				
LANguard Network Scanner	Port scanner, OS detection	http://www.gfi.com/languard/lanscan.htm		✓	Free
Description	*LANguard Network Scanner is a freeware security and port scanner to audit your network security. It scans entire networks and provides NetBIOS information for each computer such as hostname, shares, logged on user name. It does OS detection, password strength testing, detects registry issues and more. Reports are outputted in HTML.*				
NDS Snoop	Novell Enumeration Tool	http://www.novell.com/coolsolutions/		✓	Free
Description	*Provides the ability to enumerate a variety of NDS objects and values.*				
Nmap	Port scanner, OS detection	http://www.insecure.org/nmap/	✓	✓	Free
Description	*Nmap ("Network Mapper") is an open source utility for network exploration or security auditing. It was designed to rapidly scan large networks, although it also works against single hosts. Nmap uses raw IP packets to determine what hosts are available on the network, what services (ports) they are offering, what operating system (and version) they are running, what type of packet filters/firewalls are in use, and other characteristics.*				
Solarwinds	Network enumeration	http://www.solarwinds.net/		✓	$
Description	*A collection of network and management and discovery tools.*				
SuperScan	Port scanner, OS detection, Banner enumeration	http://www.foundstone.com/		✓	Free

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
Description	*A GUI port mapper. It will rapidly scan large networks to determine what hosts are available on the network, was services they are offering, the version of these services and the type and version of the operating system. Will also perform reverse DNS lookup.*				

Table C.4: Scanning and Enumeration Tools

C.5 Vulnerability Assessment Tools

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
CyberCop Scanner	Vulnerability scanner	http://www.pgp.com/products/	✓	✓	$
Description	*CyberCop Scanner is a network-based vulnerability-scanning tool that identifies security holes on network hosts.*				
ISS Internet Scanner	Vulnerability scanner	http://www.iss.net/		✓	$
Description	*ISS Internet Scanner is a network-based vulnerability-scanning tool that identifies security holes on network hosts.*				
Nessus	Vulnerability scanner	http://www.nessus.org/	✓	✓ (client only)	Free
Description	*A freeware network-based vulnerability-scanning tool that identifies security holes on network hosts.*				
SecureScanNX	Vulnerability scanner	http://www.vigilante.com/securescan/		✓	$
Description	*SecureScan NX is a corporate network security assessment tool that proactively probes your organization's network and firewalls to assess vulnerabilities and suggest corrective action.*				
SAINT	Vulnerability scanner	http://www.wwdsi.com/saint/	✓		$
Description	*SAINT is an updated and enhanced version of SATAN, is designed to assess the security of computer networks.*				
SARA	Vulnerability scanner	http://www-arc.com/sara/	✓		Free
Description	*Sara is a freeware network-based vulnerability-scanning tool that identifies security holes on network hosts.*				
SATAN	Vulnerability scanner	http://www.fish.com/satan/	✓		Free
Description	*SATAN is a tool to help system administrators. It recognizes several common networking-related security problems, and reports the problems without actually exploiting them.*				

Table C.5: Vulnerability Assessment Tools

C.6 War Dialing Tools

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
PhoneSweep	War Dialer	http://www.sandstorm.net/		✓	$
Description	*A commercial war-dialing program that supports multiple modems and attempts automated penetration.*				
Telesweep	War Dialer	http://www.securelogix.com/telesweepsecure/		✓	$
Description	*A commercial war dialing application that supports multiple modems and attempts to automated penetration.*				
THC	War Dialer	http://www.thc.org/releases.php		✓	Free
Description	*Freeware DOS based war dialing program.*				
ToneLoc	War Dialer	http://www.securityfocusonline.com/tools/category/26		✓	Free
Description	*Freeware DOS based war dialing program.*				

Table C.6: War Dialing Tools

C.7 Wireless Networking Tools

Tool	Capabilities	Website	Linux/ Unix	Win32	Cost
Aerosol	Wireless Sniffer	http://www.sec33.com/sniph/aerosol.php		✓	Free
Description	*Aerosol is a freeware wireless LAN sniffer tool, which can also crack WEP encryption keys. Aerosol operates by passively monitoring transmissions, computing the encryption key when enough packets have been gathered.*				
AirSnort	Wireless Sniffer	http://airsnort.shmoo.com/	✓		Free
Description	*AirSnort is a freeware wireless LAN sniffer tool, which recovers encryption keys. AirSnort operates by passively monitoring transmissions, computing the encryption key when enough packets have been gathered.*				
Kismet	Wireless Sniffer	http://www.kismetwireless.net/	✓		Free
Description	*Kismet is an 802.11b wireless network sniffer. It is capable of sniffing using almost any wireless card supported in Linux.*				
Netstumbler	Wireless Sniffer	http://www.netstumbler.com		✓	Free
Description	*Netstumbler is a 802.11b tool that listens for available networks and records data about that access point. A version is available for the Pocket PC.*				
Sniffer Wireless	Wireless Sniffer	http://www.sniffer.com/		✓	$
Description	*A Sniffer Wireless is a commercial wireless LAN sniffer that provides network monitoring, capturing, decoding, and filtering capabilities.*				
WEPCrack	WEP encryption cracker	http://sourceforge.net/projects/wepcrack/	✓		Free
Description	*WEPCrack is a tool that cracks 802.11 WEP encryption keys using the latest discovered weakness of RC4 key scheduling.*				
WaveStumbler	Wireless Network Mapper	http://www.cqure.net/tools08.html	✓		Free
Description	*WaveStumbler is a freeware console based 802.11 network mapper for Linux. It reports the basic wireless network characteristics including channel, WEP, ESSID, MAC etc.*				

Table C.7: Wireless Networking Testing Tools

C.8 Host Based Firewalls

Tool	Web Site	Linux/ Unix	Win32	MacOS	Cost
BlackIce	http://www.networkice.com/		✓		$
McAfee Personal Firewall	http://www.mcafee.com/		✓		$
NeoWatch Personal Firewall	http://www.neoworx.com/		✓		$
Net Barrier	http://www.intego.com/netbarrier/			✓	$
Norton Personal Firewall	http://www.symantec.com/		✓		$
PC Viper	http://www.pcviper.com/		✓		$
Securepoint	http://www.securepoint.cc/		✓		Free
SINUS	http://www.ifi.unizh.ch/ikm/SINUS/	✓			Free
SmoothWall	http://www.smoothwall.org/	✓			Free
Sygate Personal Firewall	http://www.sygate.com/		✓		Free[24]
T.Rex	http://www.opensourcefirewall.com/	✓			Free
Tiny Firewall	http://www.tinysoftware.com/		✓		Free [25]
Winproxy	http://www.winproxy.com/		✓		$
ZoneAlarm	http://www.zonelabs.com/		✓		$ or Free

Table C.8: Host-Based Firewall Tools

Note: Windows XP contains a host-based firewall that can be enabled to provide port blocking.

[24] Free for personal use only.
[25] Free for personal use only prior to Version 3. Cost for commercial use.

Appendix D. Example Usage Of Common Testing Tools

D.1 Nmap

A commonly used port scanner for identifying active hosts and associated services (i.e., open ports) is nmap (see Appendix C for website). Nmap allows for a variety of different types of port scans to be used in order to determine whether a port is open or closed. Nmap uses raw IP packets to identify the available hosts on a network, the services or ports that are open, type of operating system and version that hosts are running, type of packet filters and firewalls in use, and other characteristics.

The most basic form of port scanning supported by nmap is the TCP connect() scan, using the *-sT* option flag (nmap is case sensitive). The connect() system call provided by the host operating system is used to attempt to open a connection to any or all ports (user selects) on a remote host. If the port is listening or open, then the connect() will succeed, otherwise the port is not listening or is in a closed state. No special privileges are needed in order to employ this kind of scan.

A more common scan that is not as easily detected as the TCP connect() scan is the TCP SYN scan, also known as a SYN Stealth scan or "half-open" scan, since nmap does not open a full TCP connection (see Figure D.1). This scan is implemented using the *-sS* flag. On a Unix/Linux host running nmap, root privileges are needed in order to create the custom SYN packets that are needed for this type of scan.

First a SYN packet is sent as though the machine running nmap is initiating a "genuine" TCP connection. The host running nmap then waits for a response. A SYN|ACK response is indicative of a listening or open port. A response of RST is indicative of a non-listening or closed port. If a SYN|ACK is received, a RST is immediately sent to "cancel" the connection. This final action is required to remove the possibility of causing a SYN flood DoS attack. This can occur because all pending connections are stored in a buffer. If a RST is not sent, the target host's buffer may reach capacity. When this occurs, legitimate requests will not be processed resulting in a DoS until either a RST is received or timeout occurs on the pending requests.

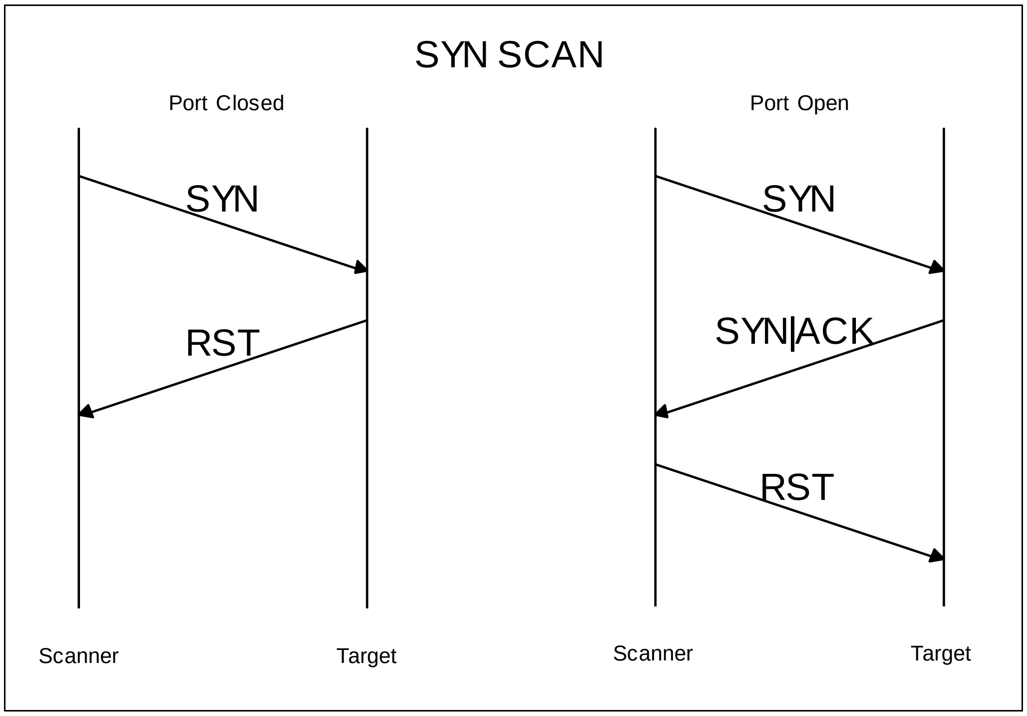


Figure D.1: SYN Stealth Scan

When SYN scanning is not clandestine enough, Stealth FIN, XMAS Tree, or Null scan modes can be used. These scans are implemented using the *-sF*, *-sX*, and *-sN* flags respectively. The FIN scan uses a bare FIN scan as the probe (see Figure D.2). The XMAS Tree scan turns on the FIN, URG, and PUSH flags (see Figure D.3). The Null scan turns off all flags (see Figure D.4). With all of these, a response of RST is indicative of a non-listening or closed port while no response is indicative of a listening or open port. This response is based on the Request for Comments (RFC) 793. Microsoft does not implement these features therefore these scans will not work properly when scanning a Windows host. Root privileges are required to create custom packets needed for these scans.

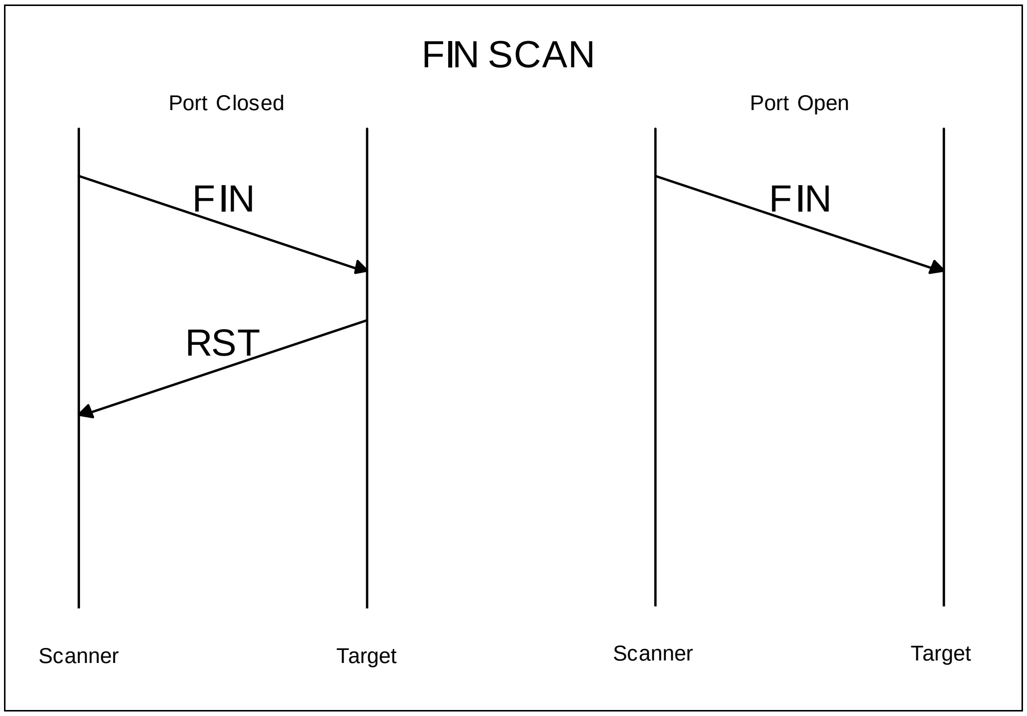


Figure D.2: FIN Scan

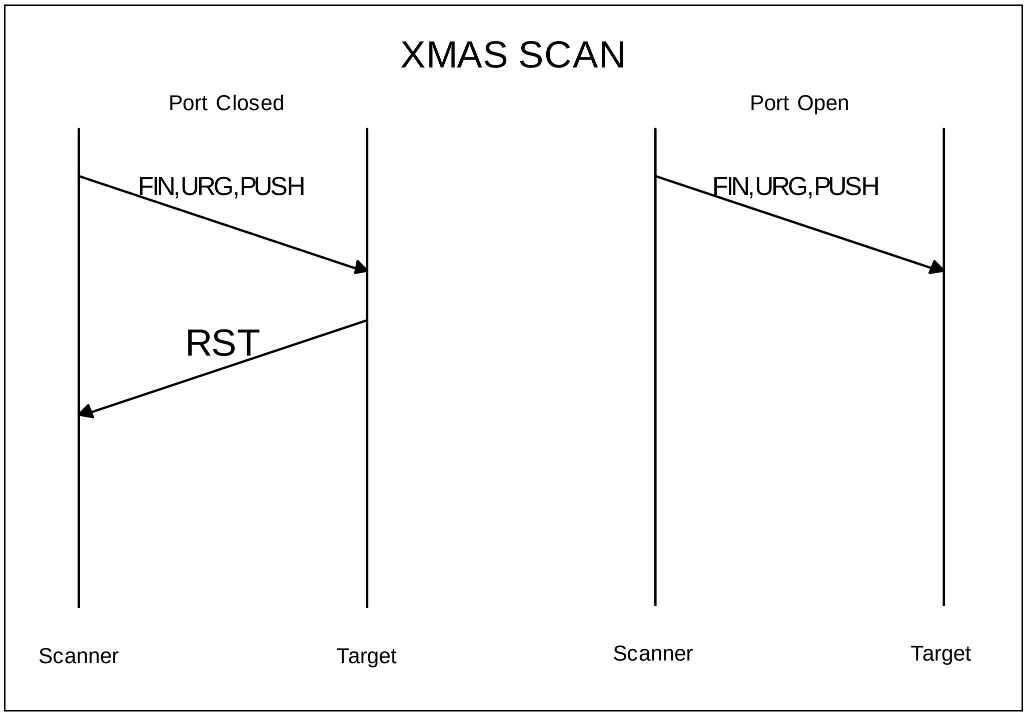


Figure D.3: XMAS Scan

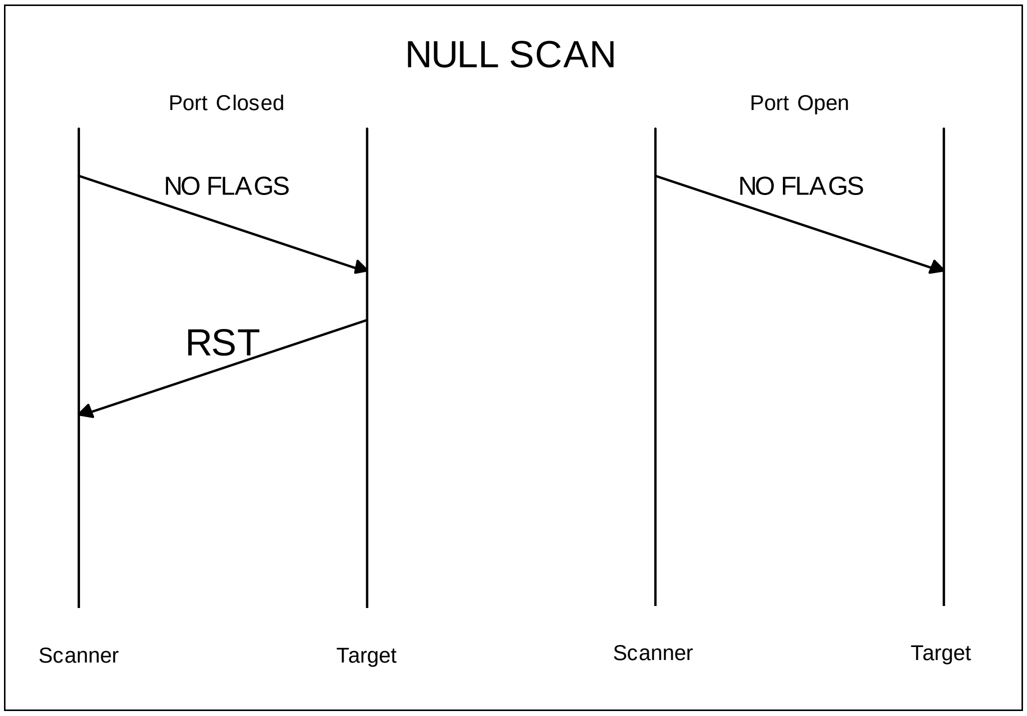


Figure D.4: Null Scan

Normally, a ping is used to determine if a host is up on a network before scanning. In a network environment that does not allow ICMP echo requests or responses, the *-P0* flag can be used to prevent pinging hosts before scanning them. This is generally useful for scanning through firewalls.

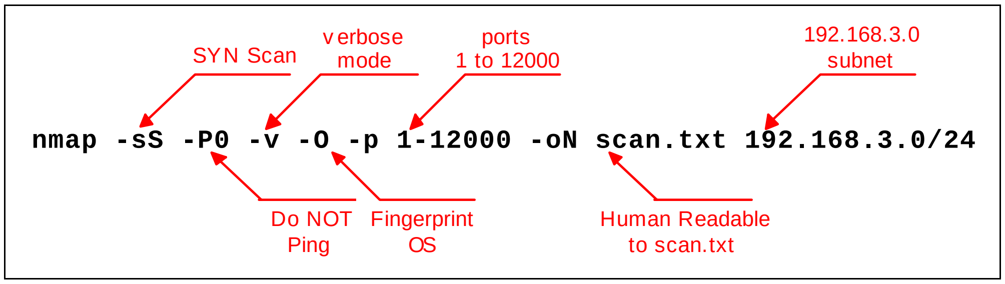
Nmap allows other types of ping requests to be used also. These types include a "TCP" ping, connection request ping, and true ICMP ping or ICMP echo request. A "TCP" ping, flag of *-PT*, sends out TCP ACK packets throughout the target network and waits for responses. Hosts that are up on the network should respond with a RST. A connection request ping, flag of *-PS*, sends out connection request or SYN packets onto the target network. Hosts that are on the network should respond with a RST. A true ICMP ping, flag of *-PI*, sends an ICMP echo request packet on to the network and waits for an ICMP echo response to validate hosts that are on the network. The default ping type, flag of *-PB*, uses a combination of the "TCP" ping and the ICMP ping in parallel. This allows one to find hosts that are operating behind firewalls that filter one but not both types of pings.

Another feature of nmap is the ability to remotely fingerprint the operating system and version that the scanned hosts are running. Nmap uses queries of the host's TCP/IP stack and the knowledge that different operating systems and their respective versions have different responses. This feature can be implemented with the *-O* flag.

The command line format for running nmap is as follows:

```
nmap [Scan Type(s)] [Options] <host or net #1 ... [#N]>
```

An example SYN scan of a class C network is shown in Figure D.5:



```
nmap -sS -P0 -v -O -p 1-12000 -oN scan.txt 192.168.3.0/24
```

Figure D.5: Example Nmap Configuration

This scan would perform a SYN stealth scan without first pinging the hosts on the class C subnet 192.168.3.0. In addition, the scan will check ports 1 through 12,000 inclusive for open services. After mapping the ports, nmap will attempt to fingerprint the operating system and version. All output from this scan will be in verbose mode (which provides more details on the scanned hosts) and this output will also be saved in human readable (plain text as opposed to binary) output to the file scan.txt. The output of nmap on one of the scanned hosts is provided in Figure D.6 (the text file would list similar results for each active host found).

```
Host hp5.doahq.gov (192.168.3.10) appears to be up ... good.
Initiating SYN half-open stealth scan against hp5.doahq.gov
(192.168.3.10)
Interesting ports on hp5.doahq.gov (192.168.3.10):
(The 1516 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
280/tcp    open        http-mgmt
515/tcp    open        printer
631/tcp    open        unknown
9100/tcp   open        unknown
TCP Sequence Prediction: Class=trivial time dependency
                         Difficulty=1 (Trivial joke)
Sequence numbers: 6E1BC7 6E1BC8 6E1BC8 6E1BC9 6E1BCA 6E1BCB
Remote OS guesses: HP Print Server, HP LaserJet Printer
```

Figure D.6: Example Nmap Output

D.1.1 Nmap Command Summary

Usage:

nmap [Scan Type(s)] [Options] <host or net list>

Scan Types:

-sT TCP connect() scan: Most common form of TCP scanning. Because it completes the full TCP handshake it is the most detectable.

-sS TCP SYN scan: Often referred to as "half-open" scanning, because this scan does not open a full TCP connection. Because this scan does not complete the TCP handshake, it is somewhat less detectable and is often referred to as a “stealth” scan. The user will need root privileges to conduct this type of scan.

-sP Ping scanning: Uses ICMP ping to detect active hosts. It does not conduct a port scan.

-sF Stealth FIN scan: Uses a bare FIN packet as the probe.

-sU UDP scan: This scan is used to determine which UDP ports are open on a host. UDP scanning can be slow due to the fact that some hosts limit the ICMP error message rate.

Options:

-P0 With this option enabled, nmap will not attempt to ping the host prior to starting a scan. This is useful for scanning through firewalls that block ICMP echo requests.

-PT This option enables the use of TCP "ping" to determine active hosts. To set the destination port of the probe packets use **–PT <port number>**. This option is similar to the **–sP** option in determining active hosts except that it does not rely on ICMP which makes it useful for scanning through firewalls that block ICMP echo requests.

-F This option enables fast scan mode. With fast scan enabled, nmap will scan only for ports listed in the services file included with nmap.

-O This option activates remote host identification via TCP/IP fingerprinting.

-h This option displays a quick reference screen of nmap usage options.

-n/-R The option tells namp to never (**-n**) or always (**-R**) perform DNS resolution.

-v This option enables verbose mode. This mode provides additional information and can be used twice for greater effect (**-v -v)**

-oN <logfilename> Logs results of the scan in a “normal” (plain text) format to the file specified.

-oM <logfilename> Logs results of scan in a machine parsable form into the file specified.

--resume <logfilename> Resumes cancelled network scans. The log filename must be either a normal or machine parsable log from the aborted scan. Nmap will start on the machine after the last one successfully scanned in the log file.

Nmap Usage Examples:

nmap -v target.example.com
This example scans all reserved TCP ports (1-1024) on the machine target.example.com. The -v option activates verbose mode.

nmap -sS -O 192.168.10.1/24
This example launches a stealth SYN scan on all active hosts on the 192.168.10.x class 'C' network. This example will attempt to determine the operating system the scanned hosts are running. This scan will require root privileges due to the use of the SYN scan and the OS detection options.

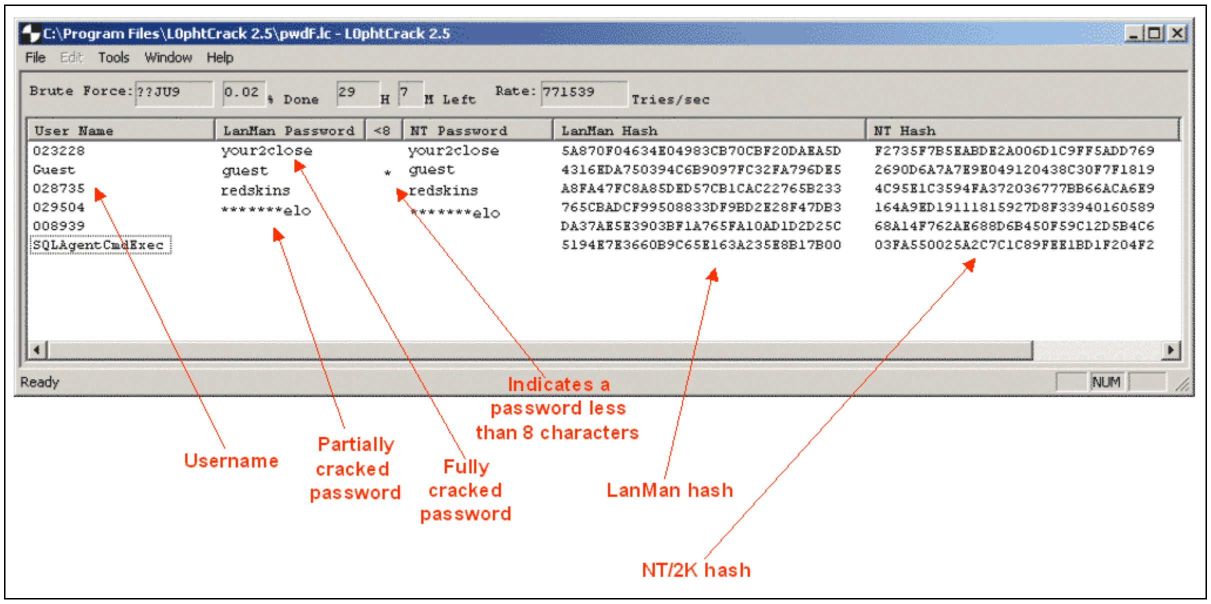
nmap –n –v –sS –O –oN nmap-sS-O_172.30.100.20-31_230.N –oM nmap sS-O_172.30.100.20-31_230.M 172.30.100.20-31,230
A stealth SYN scan of addresses 172.30.100.20 through 172.30.100.31 plus 172.30.11.230. Verbose mode, **-v** and TCP/IP **-O** fingerprinting modes are enabled. The **-n** option configures nmap not to attempt any DNS resolution. Output will be in both human readable format (**-oN)** with filename nmap-sS-O_172.30.100.20-31_230.N and machine readable format (**-oM**) with the filename nmap-sS-O_172.30.100.20-31_230.M.

D.2 L0pht Crack

One of the most popular password crackers is L0pht Crack for Windows NT and 2000. Password crackers for other platforms are included in Appendix C. For obtaining hashes, L0pht crack contains features that can be enabled to capture passwords as they traverse the network, copy them out of the Windows registry and retrieve them from Windows emergency repair disks.

When hashes are obtained, L0phtCrack first performs a dictionary attack. The dictionary used by L0phtCrack is selected by the user, or the included dictionary may be used (although more comprehensive dictionaries are available on the Internet). L0phtCrack hashes each word in the list and compares that hash to the hashes to be cracked. If the compared hashes match, L0phtCrack has found the password.

After L0phtCrack completes the dictionary attack, it iterates through the word list again using a hybrid attack. Finally L0phtcrack resorts to a brute force attack to crack any remaining hashes, trying every possible combination of characters in a set. The set of character's used by L0phtCrack in a brute force attack can be controlled by the user. The larger the set selected the longer the crack will take. Figure D.7 shows a screen capture of L0pht Crack.


C:\Program Files\L0phtCrack 2.5\pwdF.lc - L0phtCrack 2.5
File
Edit
Tools
Window
Help
Ready
NUM
Indicates a password less than 8 characters
Username
Partially cracked password
Fully cracked password
LanMan hash
NT/2K hash

Brute Force: ??JU9 0.02 % Done 29 H 7 M Left Rate: 771539 Tries/sec

User Name	LanMan Password	<8	NT Password	LanMan Hash	NT Hash
023228	your2close		your2close	5A870F04634E04983CB70CBF20DAEA5D	F2735F7B5EABDE2A006D1C9FF5ADD769
Guest	guest	*	guest	4316EDA750394C6B9097FC32FA796DE5	2690D6A7A7E9E049120438C30F7F1819
028735	redskins		redskins	A8FA47FC8A85DED57CB1CAC22765B233	4C95E1C3594FA372036777BB66ACA6E9
029504	*******elo		*******elo	765CBADCF99508833DF9BD2E28F47DB3	164A9ED19111815927D8F33940160589
008939				DA37AE5E3903BF1A765FA10AD1D2D25C	68A14F762AE688D6B450F59C12D5B4C6
SQLAgentCmdExec				5194E7E3660B9C65E163A235E8B17B00	03FA550025A2C7C1C89FEE1BD1F204F2

Figure D.7: L0phtCrack Brute Force Attack

D.3 LANguard

There are numerous freeware, shareware and commercial file integrity checkers available. A popular freeware checker is LANguard File Integrity Checker for Windows NT/2000. It can be configured from either the command line or a GUI. It can also be configured to send e-mail alerts when files are altered. To configure LANguard from the start menu select "LANguard File Integrity Checker configuration" (see Figure D.8).

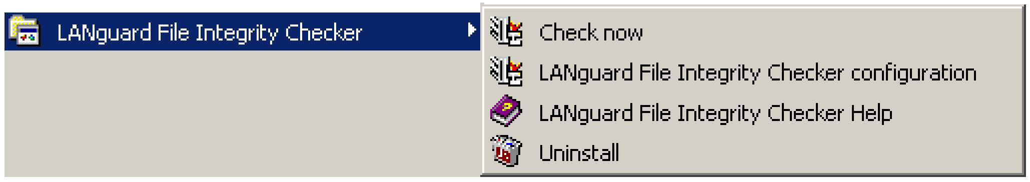
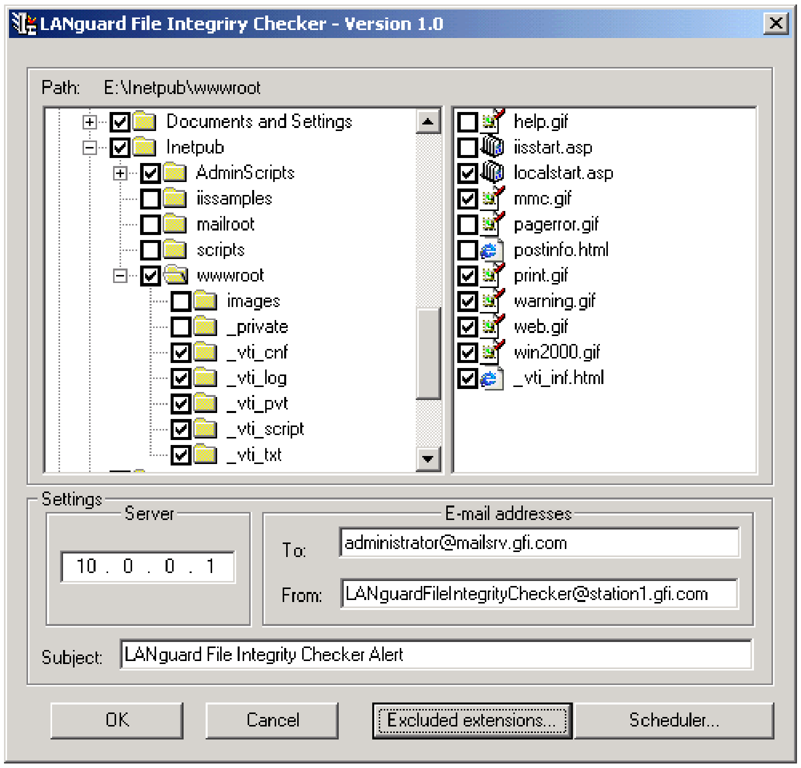

LANguard File Integrity Checker
Check now
LANguard File Integrity Checker configuration
LANguard File Integrity Checker Help
Uninstall

Figure D.8: Launching LANGuard File Integrity Checker Configuration

This will open the LANguard configuration window. This window allows the user to select what files, folder (directories) and/or drives to have LANguard monitor. This should include operating system root directory and subdirectories (i.e., winnt), anti-virus program directory, critical boot files, etc. It is important that the checksum be updated whenever files are updated. For example, when new anti-virus signatures are downloaded. In addition, for e-mail updates, the SMTP server IP address and recipient e-mail address need to be configured. See Figure D.9 for more information.


LANguard File Integriry Checker - Version 1.0

Path: E:\Inetpub\wwwroot

✓	Documents and Settings
✓	Inetpub
✓	AdminScripts
☐	iissamples
☐	mailroot
☐	scripts
✓	wwwroot
☐	images
☐	_private
✓	_vti_cnf
✓	_vti_log
✓	_vti_pvt
✓	_vti_script
✓	_vti_txt

☐	help.gif
☐	iisstart.asp
✓	localstart.asp
✓	mmc.gif
☐	pagerror.gif
☐	postinfo.html
✓	print.gif
✓	warning.gif
✓	web.gif
✓	win2000.gif
✓	_vti_inf.html

Settings

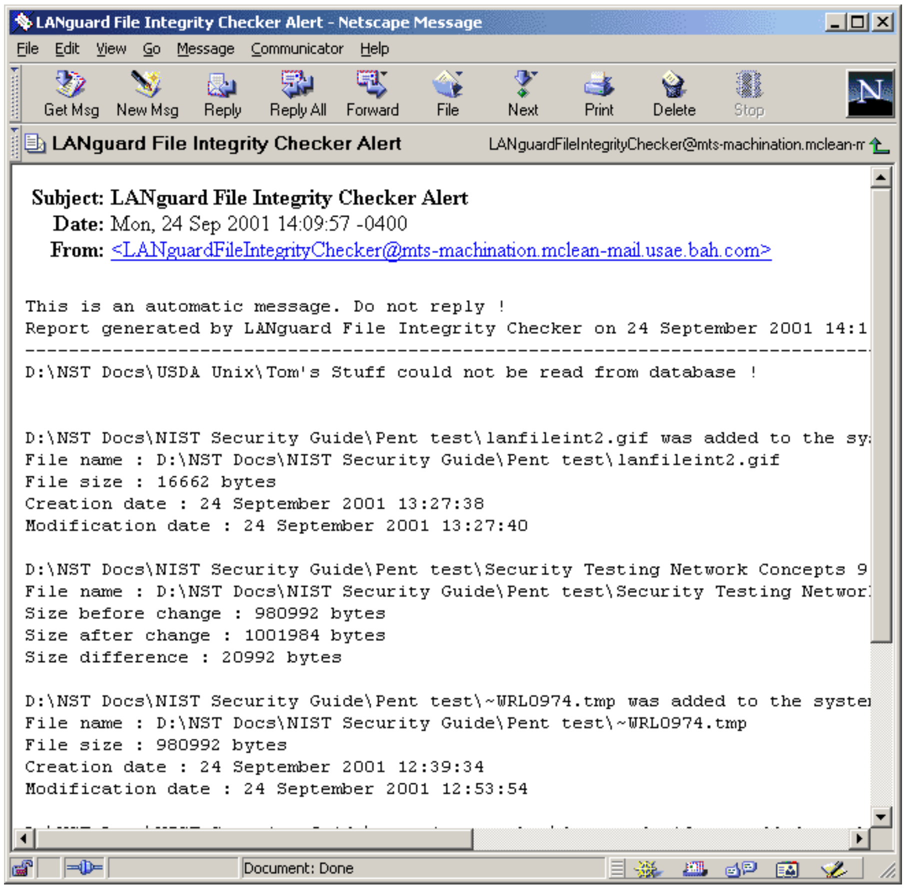
Server: 10 . 0 . 0 . 1

E-mail addresses
To: administrator@mailsrv.gfi.com
From: LANguardFileIntegrityChecker@station1.gfi.com

Subject: LANguard File Integrity Checker Alert

Figure D.9: LANGuard File Integrity Checker Configuration

When configured, an e-mail will be sent to the address specified whenever a comparison is run and changes or additions are detected. This check can be run manually or scheduled to run automatically on a periodic basis. Figure D.10 shows an example LANGuard e-mail.


LANguard File Integrity Checker Alert - Netscape Message
File Edit View Go Message Communicator Help
Get Msg New Msg Reply Reply All Forward File Next Print Delete Stop
LANguard File Integrity Checker Alert
LANguardFileIntegrityChecker@mts-machination.mclean-m
Document: Done

Subject: LANguard File Integrity Checker Alert
Date: Mon, 24 Sep 2001 14:09:57 -0400
From: <LANguardFileIntegrityChecker@mts-machination.mclean-mail.usae.bah.com>

```
This is an automatic message. Do not reply !
Report generated by LANguard File Integrity Checker on 24 September 2001 14:1
-----------------------------------------------------------------------------
D:\NST Docs\USDA Unix\Tom's Stuff could not be read from database !


D:\NST Docs\NIST Security Guide\Pent test\lanfileint2.gif was added to the sy
File name : D:\NST Docs\NIST Security Guide\Pent test\lanfileint2.gif
File size : 16662 bytes
Creation date : 24 September 2001 13:27:38
Modification date : 24 September 2001 13:27:40

D:\NST Docs\NIST Security Guide\Pent test\Security Testing Network Concepts 9
File name : D:\NST Docs\NIST Security Guide\Pent test\Security Testing Networ
Size before change : 980992 bytes
Size after change : 1001984 bytes
Size difference : 20992 bytes

D:\NST Docs\NIST Security Guide\Pent test\~WRL0974.tmp was added to the syste
File name : D:\NST Docs\NIST Security Guide\Pent test\~WRL0974.tmp
File size : 980992 bytes
Creation date : 24 September 2001 12:39:34
Modification date : 24 September 2001 12:53:54
```

Figure D.10: LANGuard File Integrity Checker Notification E-Mail

D.4 Tripwire

Tripwire is a file system integrity-checking program for UNIX and Windows operating systems. Before using Tripwire a configuration file needs to be created that designates the directories and files that are to be verified as well as the attributes verified for each. Tripwire is then run (with the initialize option) to create a database of cryptographic checksums that correspond to the files and directories specified in the configuration file.

To protect the Tripwire program, configuration file, and initialized database against corruption, they should be transferred to a medium that can be designated as physically write-protected, such as a disk or CD-ROM. This read-only version then becomes the authoritative reference program, configuration, and data, which can reliably be used to test the integrity of directories and files on the system.
In addition to one or more cryptographic checksums representing the contents of each directory and file, the Tripwire database also contains information that allows verification of:

- Access permissions and file mode settings, including effective execution settings
- Inode number in the file system
- Number of links
- User ID of the owner
- Group ID of the group of users to which access may be granted
- Size of the item
- Date and time the item was last accessed, the last modification made to the item, and the creation date and time associated with the item's inode.

For most systems the administrator should configure Tripwire to verify the integrity of all critical operating system directories and files, plus any other directories and files that the administrator considers sensitive. Administrators should pay particular attention to executable programs, daemons, scripts, and the libraries and configuration files associated with them.

The default Tripwire configuration file for most operating systems is adequate, but administrators should carefully review and edit this file to reflect their particular requirements. When choosing which attributes of files and directories to verify, administrators should consider how files and directories are configured on their system. For example log files change as events cause records to be written, so verifying the constancy of the file size for these files is not generally useful. However, monitoring changes to the size of system binaries or to access permissions for log files is usually warranted.

The install process will itemize the steps required to install Tripwire on Unix and Linux systems. It will not cover Windows based systems although the install is relatively easy under Windows when using Tripwire's install program. Before downloading and installing Tripwire, administrators should confirm that the host(s) they are installing Tripwire on include the following software applications:

- An MD5 cryptographic checksum program
- GZIP to uncompress the downloaded file
- PGP to verify the authenticity of the software distribution
- A C compiler.

Download or purchase (as appropriate) Tripwire from http://www.tripwiresecurity.com/. Once Tripwire is downloaded verify the MD5 Checksum.

D.4.1 Installing Tripwire on UNIX

Choose a storage location with sufficient space for the Tripwire distribution. Consult the Tripwire user manual before attempting installation. It contains trouble-shooting suggestions and additional details that are beyond the scope of this implementation.

After downloading Tripwire it will be necessary to unzip it:

```
$ gunzip Tripwire-1_3_1-1_tar.gz
```

To unpack the Tripwire distribution, use the system tar command:

```
$ tar xvf Tripwire-1.3.1-1_tar
```

This command creates a subdirectory named tw_ASR_1.3.1_src. Perform all operations within this subdirectory.

Several files exist in the created directory. One of these files is the Tripwire README file. It outlines the various strategies and settings for configuring and operating. Another file, Ported, lists the platforms and operating systems that Tripwire has been ported to. Find the appropriate operating system in the list and note the system settings. These will be required to build Tripwire successfully. After reviewing both the README and Ported files, an administrator should have a better understanding of how to configure Tripwire for his or her specific system.

Based on the appropriate system settings from the Ported file the administrator will have to change the Makefile to ensure that Tripwire will be correctly tuned for the specific operating system. Administrators will also have to edit the ./include/config.h file to tailor it for their specific system. Paths and names for Tripwire configuration files are specified in ./include/config.h. Administrators will need to decide where they are going to configure Tripwire to store its files. This should be read-only or removable media in order to adequately protect the data from unauthorized changes.

Next administrators will need to create an initial version of the Tripwire configuration files. Various templates for several operating systems are located in the ./config directory as part of the distribution. Administrators will need to copy the appropriate default file for their OS to the directory specified for the Tripwire configuration file location.

```
# cp config/<appropriate OS config>/etc/tw.config
```

Next, Administrators will need to edit the /etc/tw.config file (consult the manual page) to include any local system binaries, other critical files, and any additional files that they wish to monitor. After the configuration is finished, the administrator should compile the Tripwire executable using the make command:

```
$ make
```

Starts the installation using the command **make install** to place the Tripwire binary and man pages into the correct system directories. This action will have to be performed using the root account. The administrator can also place all necessary files into the desired directory manually as follows:

```
# cp man/siggen.8 /usr/local/man/man8/

# cp man/tripwire.8 /usr/local/man/man8/

# cp man/tw.config.5 /usr/local/man/man5/
# cp src/tripwire /usr/local/bin/
# cp src/siggen /usr/local/bin/
```

The Tripwire distribution includes a script-driven testing suite that checks the build process. To run the testing suite, type the following command:

```
$ make test
```

This starts a script that tests the build of Tripwire against a copy of the Tripwire database in the ./test directory. If all goes well, the output of the test matches the expected values that the script provides. For more information on the testing suite, consult the Tripwire User Manual.

For additional details on installation and configuration, consult the Tripwire User Manual, the README file, or the man pages.

D.4.2 Preparing to Use Tripwire

Once Tripwire has been compiled and tested, several additional topics need to be addressed. Not all files need the same level of protection. Tripwire comes with multiple cryptographic signature algorithms. Some execute more quickly than others and some are more secure than others. (See the Tripwire User Manual for a discussion of the individual algorithms.) Administrator will need to tailor their configuration files to reflect this tradeoff between security and performance. Tripwire's default setting is to use two algorithms to calculate cryptographic checksums; MD5 and Snefru. MD5 alone should be sufficient for most files and directories.

Tripwire can be run in one of four modes:

1. Database generation
2. Integrity checking
3. Interactive update mode
4. Database Update

Database generation and Integrity Checking are discussed in the following sections.

(1) Generating the Tripwire Database

Integrity checking requires that a previously generated database exist against which to compare. Such a database is created by the tw.config file. Once tw.config file is configured as desired, insert the prepared floppy disk (or other media as appropriate) and type the following commands:

```
# mount -n /dev/disk /floppy
# tripwire -initialize
```

The first command mounts the floppy. The second command creates a file named tw.db_<the local system's host name> within the directory /floppy/databases/. This file is the authoritative copy to which the Tripwire program refers when checking the file system's integrity for this host. Administrators can choose either the automatic or interactive update modes to maintain this database whenever changes are made to the system that need to be reflected in the Tripwire database.

After completing database generation, place a copy of the Tripwire program and its configuration file on the same disk as the database to protect the Tripwire software and critical files. Restrict access via the ownership and permissions settings on the files written to the disk so that only the root user can read them. Having all files on a write-protected floppy disk allows you to easily identify any changes by comparing versions on the disk to an authoritative reference copy. Once this step is complete, unmount and eject the floppy disk. The commands to execute this step are as follows:

```
# cp /etc/tw.config /floppy
# cp /usr/local/bin/tripwire /floppy
# umount /floppy
```

Shift the write-protect tab on the disk to disable writing to it. This write-protected disk now represents the authoritative reference. Store this floppy in a physically secure location. Create an exact copy of the disk to work with so that original is not used on a daily basis.

(2) Integrity Checking

Obtain the read-only medium containing the authoritative reference from its physically secured storage. Make sure that write protection is enabled and mount the floppy disk as shown below:

```
# mount -n /dev/disk /floppy
# echo 'test' > /floppy/test
/floppy/test: cannot create
```

If the file test exists after the last command, the floppy is not write-protected.

Compare each directory and file with its authoritative reference data. Identify any files whose contents or other attributes have changed. Execute Tripwire directly from the write-protected floppy, specifying which configuration (-c option) and database (-d option) files to use as follows:

```
# cd /floppy
# ./tripwire -c ./tw.config -d ./databases/tw.db_<the local system's host name>
```

Investigate any unexpected changes among those identified. Tripwire will identify the following:

- Files or directories that have changed
- Missing files or directories
- New files or directories.

If any changes cannot be attributed to authorized activities, initiate incident response procedures immediately. Report the incident to the appropriate internal security point of contact. Provide the Tripwire report as additional data if applicable.

Return the authoritative reference data to its physically secured storage. If all changes reported by Tripwire are as expected, follow the organization's procedures for securely updating the authoritative reference copy of the Tripwire database.

When the Tripwire scan and update process are complete unmount the authoritative reference medium and return it to secure storage.

```
# umount /floppy
```

D.5 Snort

Snort, called a lightweight network intrusion detection tool by creator Martin Roesch, is a network intrusion detection system (NIDS) that can be deployed to monitor small TCP/IP networks. Snort will detect a wide variety of suspicious traffic and attack attempts. Snort is publicly available under the GNU General Public License and is free for use in all types of environments. Rule based logging is used to perform content pattern matching and to detect a variety of attacks and probes. A simple language for the rule sets is used to expedite new rules as new attacks and exploits appear.

D.5.1 Snort Plug-ins

To allow for easy reporting, notification, and monitoring of log files and alerts generated by Snort, there are many additional programs that have been designed to work in conjunction with Snort. A partial listing of programs designed for use as analysis front ends would include:

- ACID, the Analysis Console for Intrusion Databases, is a PHP-based analysis engine that can search and process a database of incidents generated by Snort. Features for ACID include a query-builder and search interface, a packet view to display packet information for layers 3 and 4, an alert management system, and a charting and statistics generator.
- ARIS Extractor is used to parse the Snort logs before sending the output to SecurityFocus's ARIS database for analysis. The ARIS database is a service that allows network administrators to submit suspicious network traffic and intrusion attempts anonymously so that a detailed analysis and tracking can occur using the data from all contributors.
- Snort Report is a PHP-based front end for Snort that generates easy to read real-time reports from your MySQL or PostgreSQL database.
- SnortSnarf is a Perl program that converts files of alerts from Snort into HTML output. This program can be run on a schedule to generate HTML reports for use in diagnostic inspection and tracking down problems.

D.5.2 Snort Installation

Snort has been released or compiled successfully in multiple packages for different platforms, including:

- Linux
- OpenBSD
- FreeBSD
- NetBSD
- Solaris
- SunOS 4.1.X
- HP-UX
- AIX

- IRIX
- Tru64
- MacOS X Server
- Win9x/NT/2000

The first step to installing Snort on a machine is to download all the appropriate files that will be needed. Depending on what plug-ins or add-on programs are intended to be used, the list of downloads could vary. The main files that are necessary are:

1. Snort
2. Snort Rules
3. WinPcap (for Windows)

Snort and the Snort rules can be found from the official Snort web page, http://www.snort.org. WinPcap is a free packet capture architecture for Windows (http://netgroup-serv.polito.it/winpcap/install/default.htm). The packet filter is a device driver that adds the ability to capture and send raw data from a network card. With WinPcap, there is also the ability to filter and store the captured packets in a buffer. This ability to filter and store raw data packets from a network card is what makes WinPcap such a critical component of the Snort NIDS on the Windows Platform.

D.5.3 Snort Rules

Snort comes with a default rule set, snort.conf. While this file is a good start, many administrators will probably wish to modify it for their particular needs and requirements. Snort rules are simple, yet effective in terms of detection capabilities. As shown in Table D.1, Snort rule sets can include the following rule types:

Rule Types	Function
Protocol	Protocols Snort analyzes for suspicious behavior: TCP/UDP/ICMP/IP
IP Address	Source and/or Destination IP address for matching
Direction	Direction of traffic
Port of Interest	Port traffic should be on to match
Option Fields	Additional options for use in matching rules to packets

Table D.1: Snort Rule Types

The option fields within the Snort rules allow for additional options to be included in the rules. The options for a rule are processed using a logical AND between them. This means that all options for a rule must be true for Snort to perform the rules action. A partial listing of these options is included in Table D.2.

Option Field	Function
content	Search payload for specified pattern
flags	Test TCP flags for specified setting
ttl	Check IP header TTL field
itype	Match on ICMP type field
icode	Match on ICMP code field
minfrag	Set threshold value for IP fragment size
id	Test the IP header for specified value
ack	Look for specific TCP header acknowledgement number
seq	Look for specific TCP header sequence number
logto	Log packets matching the rule to the specified filename
dsize	Match on the size of the packet payload
offset	Sets the offset into the packet payload to begin a content search
depth	Sets the number of bytes from the start position to search through
msg	Sets the message to be sent when a packet generates an event

Table D.2: Snort Rule Options

There are five base action directives that Snort can use when a packet matches a specified rule pattern (see Table D.3).

Rule Action	Function
Pass	Ignore the packet and let it pass
Log	Write the full packet to the logging routine specified at run time
Alert	Generate an event notification using the selected alert method, then log the packet
Activate	Alert, and then turn on another dynamic rule
Dynamic	Remain idle until activated by an Activate rule, then act as a log rule

Table D.3: Snort Rule Actions

Some example Snort rules, as found from http://www.snort.org/docs/lisapaper.txt, are included below.

```
log tcp any any -> 10.1.1.0/24 79
```

The above rule would record all traffic inbound for port 79 (finger) going to the 10.1.1 class C network address space.

An example using an option field is as follows:

```
alert tcp any any -> 10.1.1.0/24 80 (content: "/cgi-bin/phf"; msg: "PHF probe!";)
```

The rule above would detect attempts to access the PHF service on any of the local network’s web servers. When such a packet is detected on the network, an event notification alert is generated and the entire packet is logged using the logging mechanism selected at run time.

Additional Snort rules examples can be found within the document mentioned above or within the snort.conf file that is the default rule set included with Snort.

D.5.4 Snort Usage

There are three main modes for Snort:

1. Sniffer Mode
2. Packet Logger Mode
3. Network Intrusion Detection Mode

The Snort sniffer mode is a basic way to write some or all of the intercepted TCP/UDP/ICMP/IP headers and/or packets to the screen. This is very similar in output to that of the tcpdump.
Table D.4 shows some simple flags to use in sniffer mode:

Flag	Function
-v	Outputs the IP, TCP, UDP, and ICMP headers
-d	Outputs the packet data for IP, TCP, UDP, and ICMP traffic
-e	Outputs the data link layer headers for IP, TCP, UDP, and ICMP traffic

Table D.4: Snort Sniffer Mode Flags

Flags can be combined for cumulative results. To record the packets to disk, the packet logger mode should be used. Table D-5 shows some simple flags to use in packet logger mode.

Flag	Function
-l <log dir>	Packets specified by sniffer mode are placed into <log dir>
-h <home network>	To log relative to home network, specify which network is home
-b	Logs in binary mode, or tcpdump format
-r <log file>	Read mode, plays back logfile to perform additional screening

Table D.5: Snort Logger Mode Flags

Network intrusion detection mode can be configured in many ways. There are several alert output modes in addition to logging methods. The default alert method is to use “full” alerts and to log in decoded ASCII format.

The alert output modes are described in Table D.6.

Alert	Description
-A fast	Simple format with timestamp, alert message, source and destination IPs and ports
-A full	Default mode, print alert message and full packet headers
-A unsock	Send alerts to a UNIX socket so another program can listen on
-A none	Turn off alerting

Table D.6: Snort IDS Mode Flags

Packets can be logged using their default decoded ASCII format, to a binary file, or not at all. To disable packet logging, the –N command line switch should be used.

Additional command line flags and configurations can be found within Snort documentation. Table D.7 contains some links to sites with information and tools for use with Snort.

Site/Tool/Info	Website
ACID	http://www.cert.org/kb/acid/
ARIS	http://aris.securityfocus.com
Incident.org Plugin	http://www.incident.org/snortdb/
Snort	http://www.snort.org
Snort Documentation	http://www.snort.org/documentation.html
Snort User Manual	http://www.snort.org/docs/writing_rules
Snort Downloads	http://www.snort.org/downloads.html
Snort Report	http://www.circuitsmaximus.com
Snorticus Shell Scripts	http://snorticus.baysoft.net/
SnortSnarf	http://www.silicondefense.com/software/snortsnarf/
Whitehats.com	http://www.whitehats.com
WinPcap	http://netgroup-serv.polito.it/winpcap

Table D.7: Snort Web Resources

D.6 Nessus

Nessus is a fast and modular vulnerability scanner released by Renaud Deraison. The freeware client/server tool audits a network remotely to enumerate and test the known vulnerabilities against a database that is updated daily by the Internet security community in the form of plug-ins. Some common plug-ins or security tests are for backdoors, denial of services, firewalls, Windows, etc. The user can extend the test suite by using the Nessus Attack Scripting Language (NASL) to write a new security test. Nessus is composed of a server component installed on a host where all the tests are launched and client software deployed on another system to control the scan. The scan outputs are in the form of complete exportable reports reflecting the detected vulnerabilities, the risk level, and a remedy to the exploit.

D.6.1 Nessus Plug-ins

By default, Nessus can perform various security tests classified in the following plug-ins families:

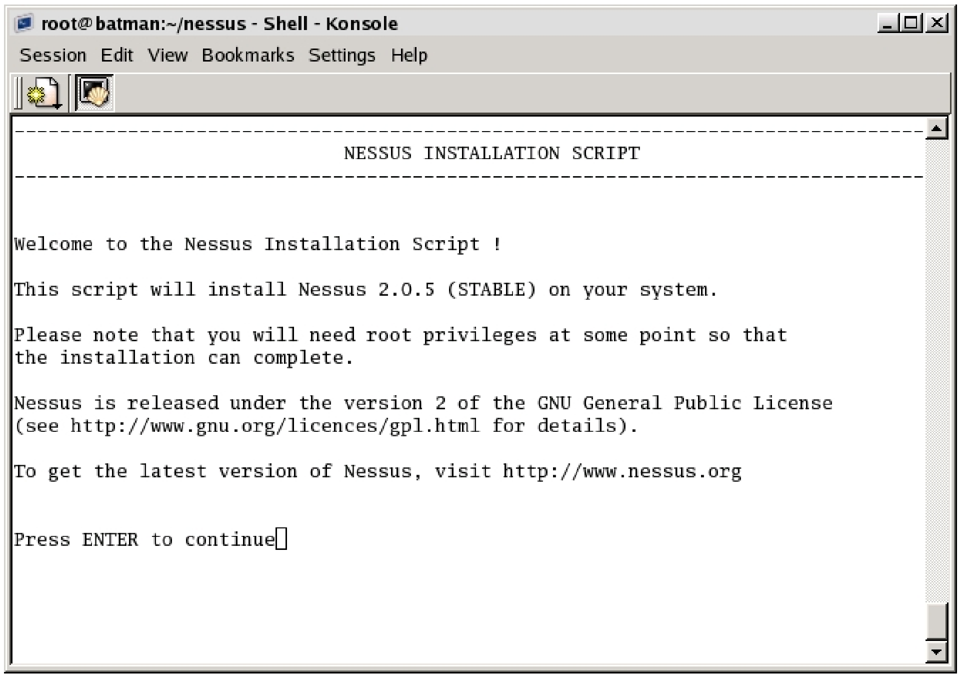
- Backdoors
- CGI abuses
- CISCO
- Default Unix Accounts
- Denial of Service
- Finger abuses
- Firewalls
- FTP
- Gain a shell remotely
- Gain root remotely
- General
- Misc.
- Netware
- Port scanners
- Remote file access
- RPC
- Settings
- SMTP problems
- SNMP
- Untested
- Useless services
- Windows
- Windows: User management

Refer to the following web page for a complete list of security checks:
http://cgi.nessus.org/plugins/dump.php3.

D.6.2 Nessus Installation and Usage

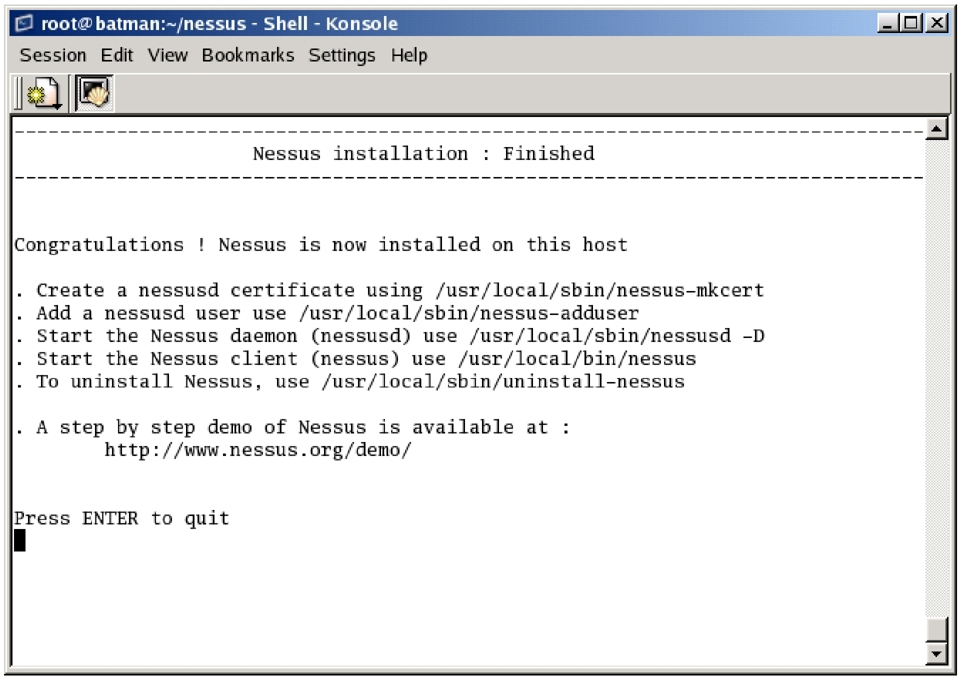
The Nessus server component runs on POSIX systems, i.e. Solaris, FreeBSD, GNU/Linux and others. The Nessus client software works with GTK, which is a set of Widgets used by many open-sourced programs. There is also a client program, which is designed specially for the Windows platform. The installation packages can be downloaded from the official Nessus web page, http://www.nessus.org/download.html.

1. Download the script nessus-installer.sh and execute the **`sh nessus-installer.sh`** command to install the standalone package.


root@batman:~/nessus - Shell - Konsole
Session Edit View Bookmarks Settings Help

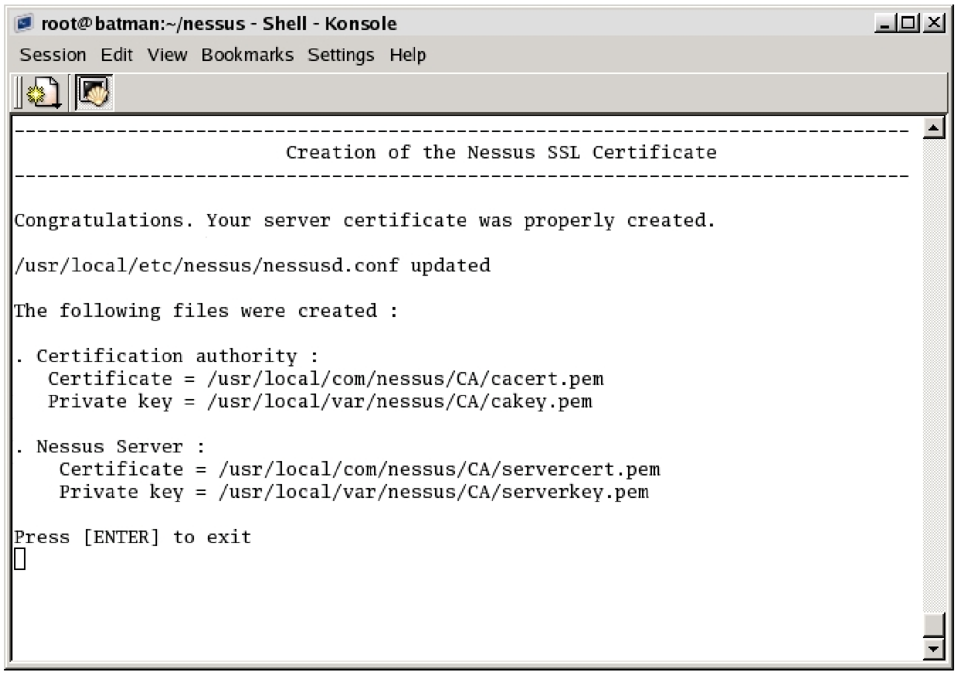
```
--------------------------------------------------------------------------------
                            NESSUS INSTALLATION SCRIPT
--------------------------------------------------------------------------------


Welcome to the Nessus Installation Script !

This script will install Nessus 2.0.5 (STABLE) on your system.

Please note that you will need root privileges at some point so that
the installation can complete.

Nessus is released under the version 2 of the GNU General Public License
(see http://www.gnu.org/licences/gpl.html for details).

To get the latest version of Nessus, visit http://www.nessus.org


Press ENTER to continue
```

2. After answering a few questions, Nessus is compiled and installed on the system. The following figure shows that the program has been installed successfully and the various Nessus commands.


root@batman:~/nessus - Shell - Konsole
Session Edit View Bookmarks Settings Help

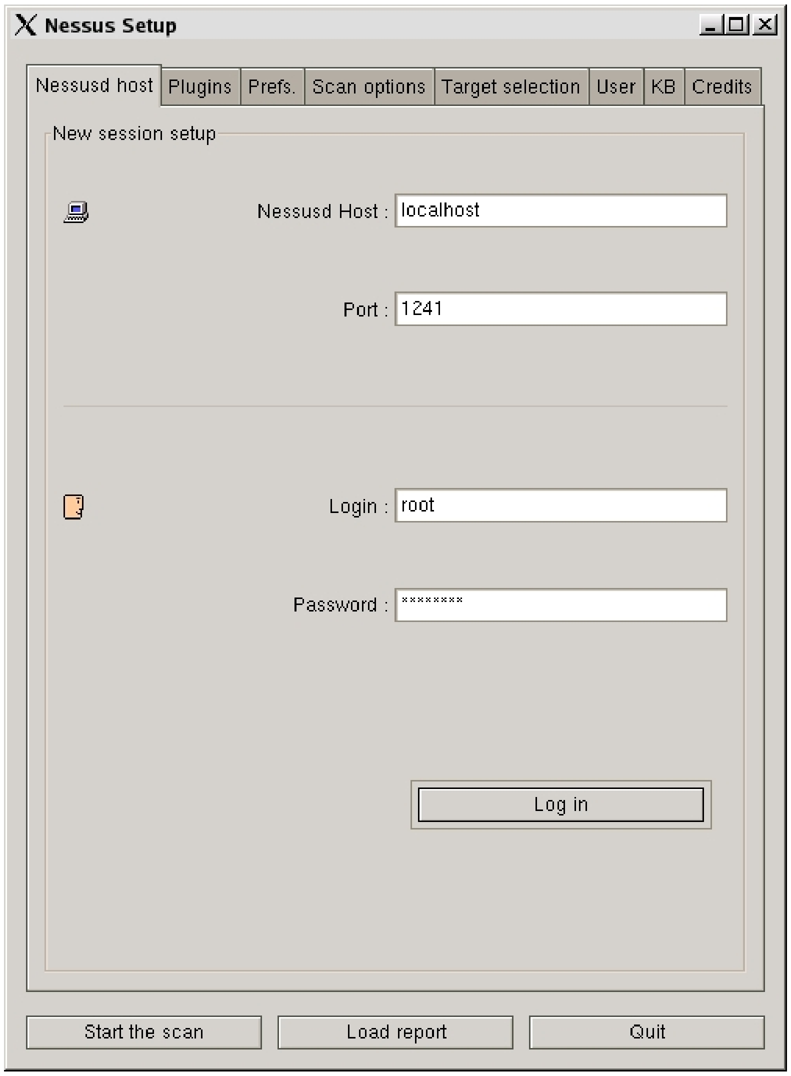
```
--------------------------------------------------------------------------------
                     Nessus installation : Finished
--------------------------------------------------------------------------------


Congratulations ! Nessus is now installed on this host

. Create a nessusd certificate using /usr/local/sbin/nessus-mkcert
. Add a nessusd user use /usr/local/sbin/nessus-adduser
. Start the Nessus daemon (nessusd) use /usr/local/sbin/nessusd -D
. Start the Nessus client (nessus) use /usr/local/bin/nessus
. To uninstall Nessus, use /usr/local/sbin/uninstall-nessus

. A step by step demo of Nessus is available at :
        http://www.nessus.org/demo/


Press ENTER to quit
```

3. Run the **`/usr/local/sbin/nessus-mkcert`** command to create a nessusd certificate. The following figure shows that the certificate has been successfully created.


root@batman:~/nessus - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
-------------------------------------------------------------------------------
                        Creation of the Nessus SSL Certificate
-------------------------------------------------------------------------------

Congratulations. Your server certificate was properly created.

/usr/local/etc/nessus/nessusd.conf updated

The following files were created :

. Certification authority :
   Certificate = /usr/local/com/nessus/CA/cacert.pem
   Private key = /usr/local/var/nessus/CA/cakey.pem

. Nessus Server :
    Certificate = /usr/local/com/nessus/CA/servercert.pem
    Private key = /usr/local/var/nessus/CA/serverkey.pem

Press [ENTER] to exit
```

4. Execute the **`/usr/local/sbin/nessus-adduser`** command to create a Nessus user account that is userd to perform a scan.
5. To update the script automatically, use **`/usr/local/sbin/nessus-update-plugins`** command. This will download the current security checks from the Nessus site.
6. Start the Nessus daemon (nessusd) by executing the **`/usr/local/sbin/nessusd –D`** command.
7. Run the **`/usr/local/bin/nessus`** command to start the Nessus client (nessus) that can be used to configure and perform the vulnerability audits.
8. Enter the user name and password in order to operate the program.



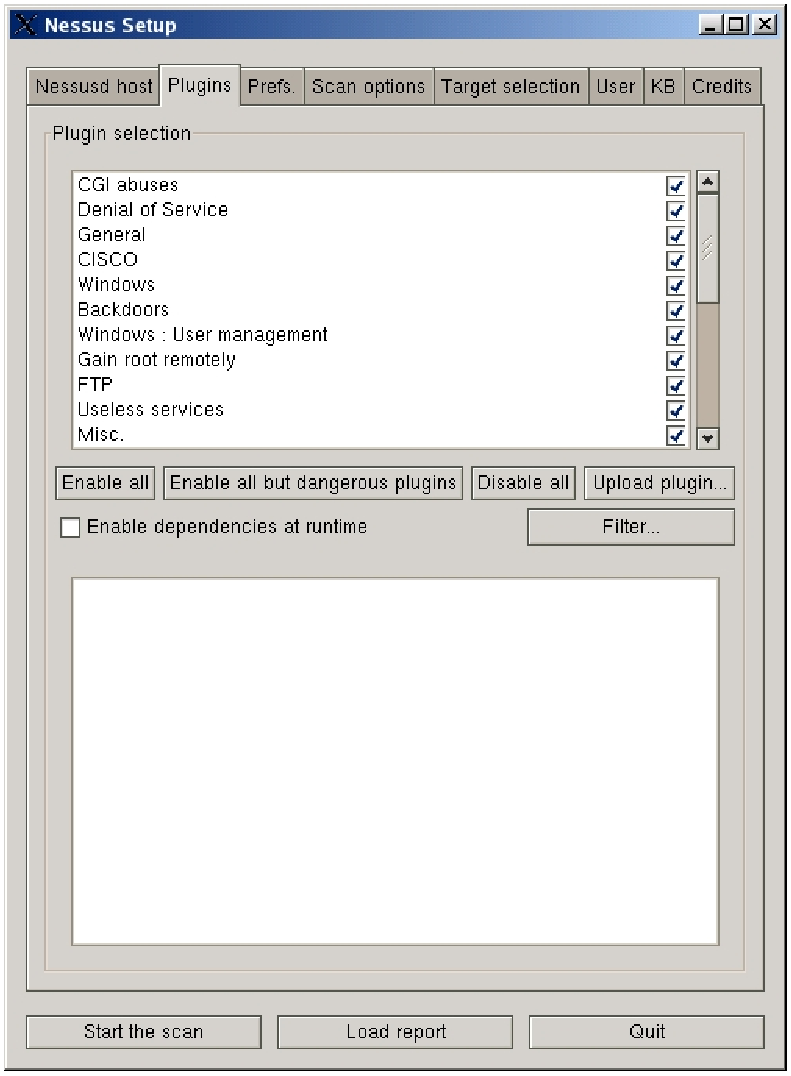
Nessusd Host : localhost

Port : 1241

Login : root

Password : ********

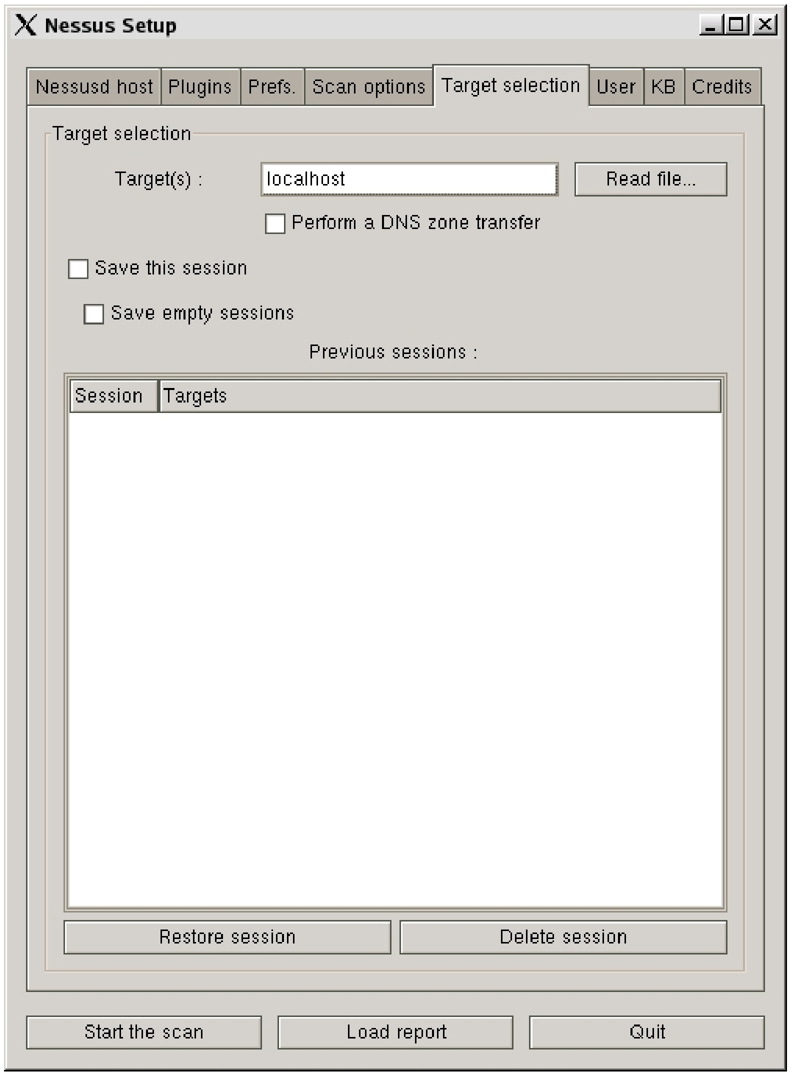
9. Select the different plug-ins containing the security checks that will be used to scan a host. Note: Nessus includes various Denial of Service tests that may crash a vulnerable target system.



CGI abuses	✓
Denial of Service	✓
General	✓
CISCO	✓
Windows	✓
Backdoors	✓
Windows : User management	✓
Gain root remotely	✓
FTP	✓
Useless services	✓
Misc.	✓

☐ Enable dependencies at runtime

10. Choose the target host or system and initiate the scan.



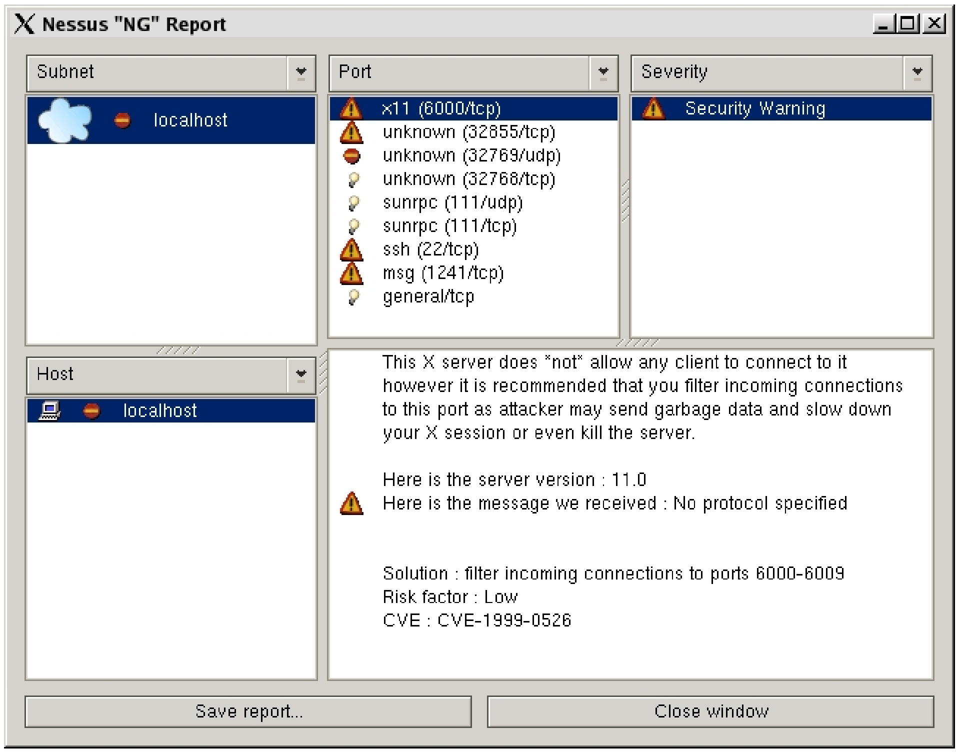
Target(s) : localhost

☐ Perform a DNS zone transfer

☐ Save this session

☐ Save empty sessions

11. At the completion of the scan, a report reflects the open ports, detected services, security impact and severity, and recommended solution. The report can be saved in various formats, i.e. HTML, XML, others.



localhost

x11 (6000/tcp)
unknown (32855/tcp)
unknown (32769/udp)
unknown (32768/tcp)
sunrpc (111/udp)
sunrpc (111/tcp)
ssh (22/tcp)
msg (1241/tcp)
general/tcp

Security Warning

localhost

```
This X server does *not* allow any client to connect to it
however it is recommended that you filter incoming connections
to this port as attacker may send garbage data and slow down
your X session or even kill the server.

Here is the server version : 11.0
Here is the message we received : No protocol specified


Solution : filter incoming connections to ports 6000-6009
Risk factor : Low
CVE : CVE-1999-0526
```

For a detailed demonstration, refer the following web page.

Appendix E. Index

A

Accountability, 3-16
acronyms, 1-4
administrative security, 2-2
Aerosol, C-8
AirSnort, C-8
Audience, 1-3
Authentication, 3-16
automated audit tools, 3-7
Availability, 3-16

B

banner grabbing, 3-2, 3-4
BlackIce, C-9
Blue Teaming, 3-12
brute force, 3-6, D-8
buffer overflow, 3-15
buffer overflows, 3-7
Bugtraq, 2-1

C

C&A, 2-1, 2-4
certification and accreditation. *See* C&A
CGI attacks, 3-7
checksum, 3-8, D-9, D-11
Chief Information Officer. *See* CIO
CIO, 2-4, 4-1, A-1, B-1
communication security, 2-2
complete mediation, 3-18
Compliance, 3-16
compromise recording, 3-18
computer security, ii, 2-2, 2-5
configuration checklists, 3-17
configuration control board, 3-17
configuration management, 3-5, 3-17
consistency, 3-16
cost of testing, 4-2
cost/benefit analysis, 2-4
Crack 5, C-3
CRC, 3-8, A-1
CyberCop Scanner, C-6
Cyclic Redundancy Check. *See* CRC

D

defense in depth, 3-18
denial of service. *See* DoS
dictionary attack, 3-6, D-8
DNS, 1-2, 3-13, 3-20, A-1, C-5, D-6, D-7
Domain Name System. See DNS
DoS, 3-4, 3-12, 3-19, A-1, D-1
Dsniff, C-2
DUMPSec, C-4

E

emanations security, 2-2
Ethereal, C-2

F

fail-safe, 3-18
false positive errors, 3-4
file and directory permissions, 3-15
file descriptor attacks, 3-15
file integrity checker, 3-8
Firewalk, C-4
Firewalls, 1-1, 3-2, 3-10, 3-20, B-1
freeware, 1-1, 3-8, 3-14, 3-19, C-2, C-4, C-6, C-8, D-9
Fscan, C-4

G

Graphical User Interface. See GUI
GUI, A-1, C-2, C-3, C-5, D-9

H

Host Based Firewalls, C-9
host-based scanners, 3-5
hybrid attack, 3-6, D-8

I

ICMP, 3-2, A-1, D-4, D-6, D-17, D-18, D-19
identify benefits of testing, 4-2
IDS, 3-3, 3-7, 3-19, A-1, C-2, D-20
IMP 2.0, C-3
impact analysis, 4-1
Information Systems Security Manager. *See* ISSM
Information Systems Security Officer. *See* ISSO
Information Systems Security Officers. *See* ISSO
Information Systems Security Program Managers. See ISSM, See ISSM

information technology, ii, 1-3
Information Technology Laboratory, ii
Integrity Checkers, 3-1, 3-8, 3-20, 3-21, C-1
Internet, ii, 2-1, 3-2, 3-6, 3-7, 3-9, 3-10, A-1, B-1, C-6, D-8
InterNIC, 3-13
intrusion detection system. See IDS
IP, 3-2, 3-3, 3-7, 3-8, 3-12, 3-13, C-4, D-1, D-9, D-17, D-18, D-19
ISS Internet Scanner, C-6
ISSM, 2-4, A-1
ISSO, 2-5, A-1
ITL, ii, B-1

J

John the Ripper, C-3

K

kernel flaws, 3-15
Kismet, C-8

L

L0pht Crack, 3-6, C-3, D-8
LAN, 1-3, 3-10, 4-2, A-1, C-8
LANguard Network Scanner, C-4
LanMan password hashes, 3-6
LDAP, 3-14, A-1
Life Cycle, 2-1
Lightweight Directory Access Protocol. See LDAP
Linux, 1-4, 3-6, B-1, C-1, C-2, C-3, C-4, C-6, C-7, C-8, D-1, D-11, D-16
Local Area Network. *See* LAN
Log Review, 3-1
Log Reviews, 3-7, 3-20, 3-21

M

McAfee Personal Firewall, C-9
MD5 cryptographic checksum, D-11
misconfiguration, 3-17

N

National Institute of Standards and Technology, ii, 2-3, B-1
NDS Snoop, C-4
NeoWatch Personal Firewall, C-9
Nessus, C-6
Net Barrier, C-9
NetBIOS, 3-1, 3-14, C-4
Network Administrators, 2-5
Network Scanning, 3-1, 3-2, 3-3, 3-13, 3-19, 3-21
network sniffers, 3-6, 3-12, 3-19, C-2, C-8
network testing, 1-1
network traffic, 3-4, 3-7, C-2, D-16
network-based scanners, 3-5
NIS, 3-14, A-1
NIST, ii, iii, 2-3, 4-1
Nmap, C-4, D-1, D-4, D-5, D-6
Norton Personal Firewall, C-9
Null scan, D-2
Nwpcrack, C-3

O

open design, 3-18
Operating System. See OS
operations security, 2-2
organizational security policy, 3-16
Organizational Standards, 3-16
OS, 3-7, A-1, C-4, D-7, D-12
OS fingerprinting, 3-7

P

packet sniffer, 3-7
password crackers, C-3
Password Cracking, 3-1, 3-6, 3-20, 3-21
PBX, 3-10, A-1
PC Viper, C-9
Penetration Testing, 3-1, 3-11, 3-12, 3-13, 3-19, 3-21
perimeter defense systems, 3-20
personnel security, 2-2
PhoneSweep, C-7
physical security, 2-2
Plain Old Telephone System. See POTS
port scanner, 3-2, 3-3, C-4, D-1
POTS, A-1
prioritize systems for testing, 4-2
Privacy, 3-16
Private Branch Exchange. *See* PBX
psychological acceptability, 3-18

R

race conditions, 3-15
Red Teaming, 3-12
references, 1-1, 1-4, 3-2
Request For Comments. See RFC
RFC, A-1, D-2

risk assessment, 2-1
routers, 1-3, 3-2, 3-20
rules of engagement, 3-12, 3-15

S

SAINT, C-6
SANS, 2-1, A-1, B-1
SARA, C-6
SATAN, C-6
scanning and enumeration tools, C-4
Securepoint, C-9
SecureScanNX, C-6
Security evaluation, 2-1
security policy, 2-1, 2-2, 2-4, 2-5, 3-3, 3-4, 3-7, 3-12, 3-16, 3-21
security principals, 3-17
security program, 1-3, 3-11
security requirements, 2-3, 2-4, 2-5, 3-21
security testing and evaluation, 2-2
Senior IT Management, 2-4
separation of privilege, 3-18
shareware, 1-1, 3-9, D-9
Simple Network Management Protocol. See SNMP
simplicity, 3-18
SINUS, C-9
SMB probes, 3-7
SmoothWall, C-9
Sniffer Wireless, C-8
SNMP, A-1
Snort, 3-7, C-2, D-16, D-17, D-18, D-19, D-20
snort.conf, D-17
social engineering, 3-15, 3-19
Solarwinds, C-4
source code, 1-1
ST&E, 2-2, 2-3, A-1
Stealth FIN, D-2, D-6
stealth port scans, 3-7
SuperScan, C-4
surface vulnerability, 3-3
Sygate Personal Firewall, C-9
symbolic links, 3-15
SYN Stealth scan. See TCP SYN scan
System Administration, Networking, and Security. *See* SANS
system development, 2-1, 2-3, 2-4
system sensitivity and criticality, 4-1

T

T.Rex, C-9
TCP, 3-1, 3-2, 3-3, A-1, B-1, C-2, C-4, D-1, D-4, D-6, D-7, D-16, D-17, D-18, D-19
TCP connect scan, D-1
TCP SYN scan, D-1
TCP/IP, 3-1, 3-2, A-1, B-1, D-4, D-6, D-7, D-16
tcpdump, 3-7, D-19
TCPDump, C-2
Telesweep, C-7
THC, C-7
ToneLoc, C-7
Transmission Control Protocol/Internet Protocol. See TCP/IP
Tripwire, C-1, D-11, D-12, D-13, D-14, D-15
Trojan. See Trojans, See Trojans, See Trojans
Trojans, 3-3, 3-9, 3-15

U

UDP, 3-2, A-1, C-4, D-6, D-17, D-19
Unix, 1-4, 3-6, 3-15, C-1, C-2, C-3, C-4, C-6, C-7, C-8, D-1, D-11
User Datagram Protocol. See UDP

V

Virus Detection, 3-1
Virus Detectors, 3-9, 3-20, 3-21
viruses, 3-9, 3-20, 3-21
vulnerability scanner, 3-3, 3-4
Vulnerability Scanning, 3-1, 3-3, 3-19, 3-21
vulnerability scanning tools, C-6

W

WAN, 4-2, A-1
War Dialing, 3-1, 3-10, 3-20, 3-21
war dialing tools, C-7
War Driving, 3-10, 3-20, 3-21
WaveStumbler, C-8
WEPCrack, C-8
whois, 3-13
Wide Area Network. See Wide Area Network
Windows, 1-4, 3-2, 3-6, 3-7, 3-15, C-1, C-2, C-3, C-4, D-2, D-8, D-9, D-11, D-17
Windows 2000, 1-4, 2-1, 3-2, B-1, C-1, C-3, D-8, D-9, D-17
Windows NT, 1-4, 3-2, C-1, C-3, D-8, D-9, D-17
Windows passwords, 3-6
WinDump, C-2
Winproxy, C-9
Wireless LANs, 3-10, 3-11

wireless networking, 3-20
wireless networking tools, C-8
worm. See worms
worms, 3-9

X

XMAS Tree, D-2

Z

ZoneAlarm, C-9