4 Introduction to Computer Network Vulnerabilities

4.1 Definition

System vulnerabilities are weaknesses in the software or hardware on a server or a client that can be exploited by a determined intruder to gain access to or shut down a network. Donald Pipkin defines system vulnerability as a condition, a weakness of or an absence of security procedure, or technical, physical, or other controls that could be exploited by a threat [1].

Vulnerabilities exist not only in hardware and software that constitute a computer system but also in policies and procedures, especially security policies and procedures, that are used in a computer network system and in users and employees of the computer network systems. Since vulnerabilities can be found in so many areas in a network system, one can say that a security vulnerability is indeed anything in a computer network that has the potential to cause or be exploited for an advantage. Now that we know what vulnerabilities are, let us look at their possible sources.

4.2 Sources of Vulnerabilities

The frequency of attacks in the last several years and the speed and spread of these attacks indicate serious security vulnerability problems in our network systems. There is no definitive list of all possible sources of these system vulnerabilities. Many scholars and indeed many security incident reporting agencies, such as Bugtraq, the mailing list for vulnerabilities; CERT/CC, the US Computer Emergency Response Team; NTBugtraq, the mailing list for Windows security; RUS-CERT, the German Computer Emergency Response Team; and US DOE-CIAC, the US Department of Energy Computer Incident Advisory Capability, have called attention to not only one but multiple factors that contribute to these security problems and pose obstacles to the security solutions. Among the most frequently mentioned sources of security vulnerability problems in computer

J.M. Kizza, *Guide to Computer Network Security*, Computer Communications and Networks, DOI 10.1007/978-3-319-55606-2_4

networks are design flaws, poor security management, incorrect implementation, Internet technology vulnerability, the nature of intruder activity, the difficulty of fixing vulnerable systems, the limits of effectiveness of reactive solutions, and social engineering [2].

4.2.1 Design Flaws

The two major components of a computer system, hardware and software, quite often have design flaws. Hardware systems are less susceptible to design flaws than their software counterparts owing to less complexity, which makes them easier to test; limited number of possible inputs and expected outcomes, again making it easy to test and verify; and the long history of hardware engineering. But even with all these factors backing up hardware engineering, because of complexity in the new computer systems, design flaws are still common.

But the biggest problems in system security vulnerability are due to software design flaws. A number of factors cause software design flaws, including overlooking security issues all together. However, three major factors contribute a great deal to software design flaws: human factors, software complexity, and trustworthy software sources [3].

4.2.1.1 Human Factors

In the human factor category, poor software performance can be a result of the following:

1. *Memory lapses and attentional failures*: For example, someone was supposed to have removed or added a line of code, tested, or verified, but did not because of simple forgetfulness.
2. *Rush to finish*: The result of pressure, most often from management, to get the product on the market either to cut development costs or to meet a client deadline can cause problems.
3. *Overconfidence and use of nonstandard or untested algorithms*: Before algorithms are fully tested by peers, they are put into the product line because they seem to have worked on a few test runs.
4. *Malice*: Software developers, like any other professionals, have malicious people in their ranks. Bugs, viruses, and worms have been known to be embedded and downloaded in software, as is the case with Trojan horse software, which boots itself at a timed location. As we will see in Sect. 8.4, malice has traditionally been used for vendetta, personal gain (especially monetary), and just irresponsible amusement. Although it is possible to safeguard against other types of human errors, it is very difficult to prevent malice.
5. *Complacency*: When either an individual or a software producer has significant experience in software development, it is easy to overlook certain testing and other error control measures in those parts of software that were tested

previously in a similar or related product, forgetting that no one software product can conform to all requirements in all environments.

4.2.1.2 Software Complexity

Both software professionals and nonprofessionals who use software know the differences between software programming and hardware engineering. In these differences underlie many of the causes of software failure and poor performance. Consider the following:

1. *Complexity*: Unlike hardwired programming in which it is easy to exhaust the possible outcomes on a given set of input sequences, in software programming a similar program may present billions of possible outcomes on the same input sequence. Therefore, in software programming, one can never be sure of all the possibilities on any given input sequence.
2. *Difficult testing*: There will never be a complete set of test programs to check software exhaustively for all bugs for a given input sequence.
3. *Ease of programming*: The fact that software programming is easy to learn encourages many people with little formal training and education in the field to start developing programs, but many are not knowledgeable about good programming practices or able to check for errors.
4. *Misunderstanding of basic design specifications*: This affects the subsequent design phases including coding, documenting, and testing. It also results in improper and ambiguous specifications of major components of the software and in ill-chosen and poorly defined internal program structures.

4.2.1.3 Trustworthy Software Sources

There are thousands of software sources for the millions of software products on the market today. However, if we were required to name well-known software producers, very few of us would succeed in naming more than a handful. Yet we buy software products every day without even ever minding their sources. Most importantly, we do not care about the quality of that software, the honesty of the anonymous programmer, and of course its reliability as long as it does what we want it to do.

Even if we want to trace the authorship of the software product, it is impossible because software companies are closed within months of their opening. Chances are when a software product is 2 years old, its producer is likely to be out of business. In addition to the difficulties in tracing the producers of software who go out of business as fast as they come in, there is also fear that such software may not even have been tested at all.

The growth of the Internet and the escalating costs of software production have led many small in-house software developers to use the marketplace as a giant testing laboratory through the use of beta testing, shareware, and freeware. Shareware and freeware have a high potential of bringing hostile code into trusted systems.

For some strange reason, the more popular the software product gets, the less it is tested. As software products make market inroads, their producers start thinking of producing new versions and releases with little to no testing of current versions. This leads to the growth of what is called a *common genesis* software product, where all its versions and releases are based on a common code. If such a code has not been fully tested, which is normally the case, then errors are carried through from version to version and from release to release.

In the last several years, we have witnessed the growth of the open-source movement. It has been praised as a novel idea to break the monopoly and price gauging by big software producers and most important as a timely solution to poor software testing. Those opposed to the movement have criticized it for being a source of untrusted and many times untested software. Despite the wails of the critics, major open-source products such as Linux operating system have turned out with few security flaws; still there are fears that hackers can look at the code and perhaps find a way to cause mischief or steal information.

There has been a rise recently in Trojan horses inserted into open-source code. In fact security experts are not recommending running readily available programs such as MD5 hashes to ensure that the code hasn't been altered. Using MD5 hashes and similar programs such as MD4, SHA, and SHA-1 continually compares codes generated by "healthy" software to hashes of programs in the field, thus exposing the Trojans. According to the recent CERT advisory, crackers are increasingly inserting Trojans into the source code for tcpdump, a utility that monitors network traffic, and libpcap, a packet capture library tool [4].

However, according to the recent study by the Aberdeen Group, open-source software now accounts for more than half of all security advisories published in the past year by the Computer Emergency Response Team (CERT). Also according to industry study reports, open-source software commonly used in Linux, Unix, and network routing equipment accounted for 16 of the 29 security advisories during the first 10 months of 2002, and there is an upswing in new virus and Trojan horse warnings for Unix, Linux, Mac OS X, and open-source software [4].

4.2.1.4 Software Reuse, Reengineering, and Outlived Design

New developments in software engineering are spearheading new developments such as software reuse and software reengineering. Software reuse is the integration and use of software assets from a previously developed system. It is the process in which old or updated software such as library, component, requirements and design documents, and design patterns is used along with new software.

Both software reengineering and reuse are hailed for cutting down on the escalating development and testing costs. They have brought efficiency by reducing time spent designing or coding, popularized standardization, and led to common "look-and-feel" between applications. They have made debugging easier through use of thoroughly tested designs and code.

However, both software techniques have the potential to introduce security flaws in systems. Among some of the security flaws that have been introduced into programming is first the mismatch where reused requirement specifications and designs may not completely match the real situation at hand and nonfunctional characteristics of code may not match those of the intended recipient. Second, when using object programming, it is important to remember that objects are defined with certain attributes, and any new application using objects defined in terms of the old ones will inherit all their attributes.

In Chap. 6, we will discuss the many security problems associated with script programming. Yet there is now momentum in script programming to bring more dynamism into Web programming. Scripting suffers from a list of problems including inadequate searching and/or browsing mechanisms before any interaction between the script code and the server or client software, side effects from software assets that are too large or too small for the projected interface, and undocumented interfaces.

4.2.2 Poor Security Management

Security management is both a technical and an administrative security process that involves security policies and controls that the organization decides to put in place to provide the required level of protection. In addition, it also involves security monitoring and evaluation of the effectiveness of those policies. The most effective way to meet these goals is to implement security risk assessment through a security policy and secure access to network resources through the use of firewalls and strong cryptography. These and others offer the security required for the different information systems in the organization in terms of integrity, confidentiality, and availability of that information. Security management by itself is a complex process; however, if it is not well organized, it can result in a security nightmare for the organization.

Poor security management is a result of little control over security implementation, administration, and monitoring. It is a failure in having solid control of the security situation of the organization when the security administrator does not know who is setting the organization's security policy, administering security compliance, and who manages system security configurations and is in charge of security event and incident handling.

In addition to the disarray in the security administration, implementation, and monitoring, a poor security administration team may even lack a plan for the wireless component of the network. As we will see in Chap. 17, the rapid growth of wireless communication has brought with it serious security problems. There are so many things that can go wrong with security if security administration is poor. Unless the organization has a solid security administration team with a sound security policy and secure security implementation, the organization's security may be compromised. An organization's system security is as good as its security policy and its access control policies and procedures and their implementation.

Good security management is made up of a number of implementable security components that include risk management, information security policies and procedures, standards, guidelines, information classification, security monitoring, and security education. These core components serve to protect the organization's resources:

- A risk analysis will identify these assets, discover the threats that put them at risk, and estimate the possible damage and potential loss a company could endure if any of these threats become real. The results of the risk analysis help management construct a budget with the necessary funds to protect the recognized assets from their identified threats and develop applicable security policies that provide direction for security activities. Security education takes this information to each and every employee.
- Security policies and procedures to create, implement, and enforce security issues that may include people and technology.
- Standards and guidelines to find ways, including automated solution for creating, updating, and tracking compliance of security policies across the organization.
- Information classification to manage the search, identification, and reduction of system vulnerabilities by establishing security configurations.
- Security monitoring to prevent and detect intrusions, consolidate event logs for future log and trend analysis, manage security events in real time, manage parameter security including multiple firewall reporting systems, and analyze security events enterprise-wide.
- Security education to bring security awareness to every employee of the organization and teach them their individual security responsibility.

4.2.3 Incorrect Implementation

Incorrect implantation very often is a result of incompatible interfaces. Two product modules can be deployed and work together only if they are compatible. That means that the module must be *additive*, that is, the environment of the interface needs to remain intact. An incompatible interface, on the other hand, means that the introduction of the module has changed the existing interface in such a way that existing references to the interface can fail or behave incorrectly.

This definition means that the things we do on the many system interfaces can result in incompatibility that results result in bad or incomplete implementation. For example, ordinary addition of a software or even an addition or removal of an argument to an existing software module may cause an imbalanced interface. This interface sensitivity tells us that it is possible because of interposition that the addition of a simple thing like a symbol or an additional condition can result in an incompatible interface, leading the new symbol or condition to conflict with all applications that have been without problems.

To put the interface concept into a wide system framework, consider a system-wide integration of both hardware and software components with differing technologies with no standards. No information system products, whether hardware or software, are based on a standard that the industry has to follow. Because of this, manufacturers and consumers must contend with the constant problems of system compatibility. Because of the vast number of variables in information systems, especially network systems, involving both hardware and software, it is not possible to test or verify all combinations of hardware and software. Consider, for example, that there are no standards in the software industry. Software systems involve different models based on platforms and manufacturer. Products are heterogeneous both semantically and syntactically.

When two or more software modules are to interface one another in the sense that one may feed into the other or one may use the outputs of the other, incompatibility conditions may result from such an interaction. Unless there are methodologies and algorithms for checking for interface compatibility, errors are transmitted from one module into another. For example, consider a typical interface created by a method call between software modules. Such an interface always makes assumptions about the environment having the necessary availability constraints that the accessibility of local methods to certain states of the module. If such availability constraints are not checked before the modules are allowed to pass parameters via method calls, errors may result.

Incompatibility in system interfaces may be caused by a variety of conditions usually created by things such as:

- Too much detail
- Not enough understanding of the underlying parameters
- Poor communication during design
- Selecting the software or hardware modules before understanding the receiving software
- Ignoring integration issues
- Error in manual entry

Many security problems result from the incorrect implementation of both hardware and software. In fact, system reliability in both software and hardware is based on correct implementation, as is the security of the system.

4.2.4 Internet Technology Vulnerability

In Sect. 4.2.1, we discussed design flaws in technology systems as one of the leading causes of system vulnerabilities. In fact we pointed out that systems are composed of software, hardware, and humanware. There are problems in each one of these components. Since the humanware component is influenced by the technology in the software and hardware, we will not discuss this any further.

The fact that computer and telecommunication technologies have developed at such an amazing and frightening speed and people have overwhelmingly embraced both of them has caused security experts to worry about the side effects of these booming technologies. There were reasons to worry. Internet technology has been and continues to be vulnerable. There have been reports of all sorts of loopholes, weaknesses, and gaping holes in both software and hardware technologies.

According to National Vulnerability Database (NVD), a US government repository of standards-based vulnerability management data using the Security Content Automation Protocol (SCAP), system vulnerabilities have been on the rise ever since system vulnerability data was first captured. The system vulnerability data captured by NVD enables automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security-related software flaws, misconfigurations, product names, and impact metrics. Read more about NVD at https://nvd.nist.gov/home.cfm.

There is agreement among security experts that what is reported represents the tip of the iceberg. Many vulnerabilities are discovered and, for various reasons, are not reported.

Because these technologies are used by many who are not security experts (in fact the majority of users are not security literate), one can say that many vulnerabilities are observed and probably not reported because those who observe them do not have the knowledge to classify what has been observed as a vulnerability. Even if they do, they may not know how and where to report.

No one knows how many of these vulnerabilities are there in both software and hardware. The assumption is that there are thousands. As history has shown us, a few are always discovered every day by hackers. Although the list spans both hardware and software, the problem is more prevalent with software. In fact, software vulnerabilities can be put into four categories:

- Operating system vulnerabilities: Operating systems are the main sources of all reported system vulnerabilities. Going by the SysAdmin, Audit, Network, and Security (SANS) Institute, a cooperative research and education organization serving security professionals, auditors, system administrators, and network administrators, together with the Common Weakness Enumeration (CWE), a community-developed dictionary of weaknesses of software types, has been issuing lists annually: “CWE/SANS Top 25 Most Dangerous Software Errors.” Popular operating systems cause many of the vulnerabilities. This is always so because hackers tend to take the easiest route by exploiting the best-known flaws with the most effective and widely known and available attack tools.
- Port-based vulnerabilities: Besides operating systems, network service ports take second place in sourcing system vulnerabilities. For system administrators, knowing the list of most vulnerable ports can go a long way to help enhance system security by blocking those known ports at the firewall. Such an operation, though not comprehensive, adds an extra layer of security to the network. In fact it is advisable that in addition to blocking and deny-everything filtering, security administrators should also monitor all ports including the blocked ones for intruders who entered the system by some other means. For the most common vulnerable port numbers, the reader is referred to the latest SANS at: https://www.sans.org/security-resources/idfaq/which-backdoors-live-on-which-ports/8/4.
- Application software-based errors.
- System protocol software such as client and server browser.

In addition to highlighting the need for system administrators to patch the most common vulnerabilities, we hope this will also help many organizations that lack the resources to train security personnel to have a choice of either focusing on the most current or the most persistent vulnerability. One would wonder why a vulnerability would remain among the most common year after year, while there are advisories on it and patches for it. The answer is not very farfetched, but simple: system administrators do not correct many of these flaws because they simply do not know which vulnerabilities are most dangerous; they are too busy to correct them all or they do not know how to correct them safely.

Although these vulnerabilities are cited, many of them year after year, as the most common vulnerabilities, there are traditionally thousands of vulnerabilities that hackers often use to attack systems. Because they are so numerous and new ones are being discovered every day, many system administrators may be overwhelmed, which may lead to loss of focus on the need to ensure that all systems are protected against the most common attacks.

Let us take stock of what we have said so far. Lots and lots of system vulnerabilities have been observed and documented by SANS and CWE in their series, "CWE/SANS Top 25 Most Dangerous Software Errors." However, there is a stubborn persistence of a number of vulnerabilities making the list year after year. This observation, together with the nature of software, as we have explored in Sect. 4.2.1, means it is possible that what has been observed so far is a very small fraction of a potential sea of vulnerabilities; many of them probably will never be discovered because software will ever be subjected to either unexpected input sequences or operated in unexpected environments.

Besides the inherently embedded vulnerabilities resulting from flawed designs, there are also vulnerabilities introduced in the operating environments as a result of incorrect implementations by operators. The products may not have weaknesses initially, but such weaknesses may be introduced as a result of bad or careless installations. For example, quite often products are shipped to customers with security features disabled, forcing the technology users to go through the difficult and error-prone process of properly enabling the security features by oneself.

4.2.5 Changing Nature of Hacker Technologies and Activities

It is ironic that as "useful" technology develops so does the "bad" technology. What we call useful technology is the development in all computer and telecommunication technologies that are driving the Internet, telecommunication, and the Web. "Bad" technology is the technology that system intruders are using to attack systems. Unfortunately these technologies are all developing in tandem. In fact, there are times when it looks like hacker technologies are developing faster than the rest of the technologies. One thing is clear, though: hacker technology is flourishing.

Although it used to take intelligence, determination, enthusiasm, and perseverance to become a hacker, it now requires a good search engine, time, a little bit of knowledge of what to do, and owning a computer. There are thousands of hacker Web sites with the latest in script technologies and hundreds of recipe books and sources on how to put together an impact virus or a worm and how to upload it.

The ease of availability of these hacker tools; the ability of hackers to disguise their identities and locations; the automation of attack technology which further distances the attacker from the attack; the fact that attackers can go unidentified, limiting the fear of prosecution; and the ease of hacker knowledge acquisition have put a new twist in the art of hacking, making it seem easy and hence attracting more and younger disciples.

Besides the ease of becoming a hacker and acquiring hacker tools, because of the Internet sprawl, hacker impact has become overwhelming, impressive, and more destructive in shorter times than ever before. Take, for example, recent virus incidents such as the "I Love You," "Code Red," "Slammer," and the "Blaster" worms' spread. These worms and viruses probably spread around the world much faster than the human cold virus and the dreaded severe acute respiratory syndrome (SARS).

What these incidents have demonstrated is that the *turnaround time*, the time a virus is first launched in the wild and the time it is first cited as affecting the system, is becoming incredibly shorter. Both the turnaround time and the speed at which the virus or a worm spreads reduce the *response time*, the time a security incident is first cited in the system and the time an effective response to the incident should have been initiated. When the response time is very short, security experts do not have enough time to respond to a security incident effectively. In a broader framework, when the turnaround time is very short, system security experts who develop patches do not have enough time to reverse engineer and analyze the attack in order to produce counter immunization codes. It has been and it is still the case in many security incidents for antivirus companies to take hours and sometime days, such as in the case of the Code Red virus, to come up with an effective cure. However, even after a patch is developed, it takes time before it is filtered down to the system managers. Meantime, the damage has already been done, and it is multiplying. Likewise, system administrators and users have little time to protect their systems.

4.2.6 Difficulty of Fixing Vulnerable Systems

In his testimony to the Subcommittee on Government Efficiency, Financial Management, and Intergovernmental Relations of the US House Committee on Government Reform, Richard D. Pethia, Director, CERT Centers, pointed out the difficulty in fixing known system vulnerabilities as one of the sources of system vulnerabilities. His concern was based on a number of factors, including the ever-rising number of system vulnerabilities and the ability of system administrators to cope with the number of patches issued for these vulnerabilities. As the number of vulnerabilities rises, system and network administrators face a difficult situation. They are challenged with keeping up with all the systems they have and all the patches released for those systems. Patches can be difficult to apply and might even have unexpected side effects as a result of compatibility issues [2].

Besides the problem of keeping abreast of the number of vulnerabilities and the corresponding patches, there are also logistic problems between the time at which a vendor releases a security patch and the time at which a system administrator fixes the vulnerable computer system. There are several factors affecting the quick fixing of patches. Sometimes, it is the logistics of the distribution of patches. Many vendors disseminate the patches on their Web sites; others send e-mail alerts. However, sometimes busy system administrators do not get around to these e-mails and security alerts until sometime after. Sometimes, it can be months or years before the patches are implemented on a majority of the vulnerable computers.

Many system administrators are facing the same chronic problems: the never-ending system maintenance, limited resources, and highly demanding management. Under these conditions, the ever-increasing security system complexity, increasing system vulnerabilities, and the fact that many administrators do not fully understand the security risks, system administrators neither give security a high enough priority nor assign adequate resources. Exacerbating the problem is the fact that the demand for skilled system administrators far exceeds the supply [2].

4.2.7 Limits of Effectiveness of Reactive Solutions

Going by daily reports of system attacks and hacks, the number of system attacks is steadily on the rise. However, a small percentage of all attacks is reported, indicating a serious and growing systems security problem. However, given that just a small percentage of all attacks is reported, this table indicates a serious growing system security problem. As we have pointed out earlier, hacker technology is becoming more readily available, easier to get and assemble, more complex, and their effects more far reaching. All these indicate that urgent action is needed to find an effective solution to this monstrous problem.

The security community, including scrupulous vendors, have come up with various solutions, some good and others not. In fact, in an unexpected reversal of fortunes, one of the new security problems is to find a “good” solution from among thousands of solutions and to find an “expert” security option from the many different views.

Are we reaching the limits of our efforts, as a community, to come up with a few good and effective solutions to this security problem? There are many signs to support an affirmative answer to this question. It is clear that we are reaching the limits of effectiveness of our reactive solutions. Richard D. Pethia gives the following reasons [2]:

- The number of vulnerabilities in commercial off-the-shelf software is now at the level that it is virtually impossible for any but the best resourced organizations to keep up with the vulnerability fixes.
- According to *World Internet Usage and Population Statistics*, (http://www.internetworldstats.com/stats.htm), with a 2016 global population of 7,340,093,980, there are currently 3,611,375,813 Internet users representing almost half of the global population at 49.2%. This represents a growth of 90.4% since 2000. This is a phenomenal growth and it continues to grow at a rapid pace. At any point in time, there are millions of connected computers and smart mobile devices that are vulnerable to one form of attack or another.
- Attack technology has now advanced to the point where it is easy for attackers to take advantage of these vulnerable machines and harness them together to launch high-powered attacks.
- Many attacks are now fully automated, thus reducing the turnaround time even further as they spread around cyberspace.
- The attack technology has become increasingly complex and in some cases intentionally stealthy, thus reducing the turnaround time and increasing the time it takes to discover and analyze the attack mechanisms in order to produce antidotes.
- Internet users have become increasingly dependent on the Internet and now use it for many critical applications so that a relatively minor attack has the potential to cause huge damages.

Without being overly pessimistic, these factors, taken together, indicate that there is a high probability that more attacks are likely and since they are getting more complex and attacking more computers, they are likely to cause significant devastating economic losses and service disruptions.

4.2.8 Social Engineering

According to John Palumbo, social engineering is an outside hacker's use of psychological tricks on legitimate users of a computer system in order to gain the information (usernames and passwords) one needs to gain access to the system [5].

Many have classified social engineering as a diversion, in the process of system attack, on people's intelligence to utilize two human weaknesses: first, no one wants to be considered ignorant and second is human trust. Ironically, these are two weaknesses that have made social engineering difficult to fight because no one wants to admit falling for it. This has made social engineering a critical system security hole.

Many hackers have and continue to use it to get into protected systems. Kevin Mitnick, the notorious hacker, used it successfully and was arguably one of the most ingenious hackers of our time; he was definitely very gifted with his ability to socially engineer just about anybody [5].

Hackers use many approaches to social engineering, including the following [6]:

- *Telephone*. This is the most classic approach, in which hackers call up a targeted individual in a position of authority or relevance and initiate a conversation with the goal of gradually pulling information out of the target. This is done mostly to help desks and main telephone switch boards. Caller ID cannot help because hackers can bypass it through tricks and the target truly believes that the hacker is actually calling from inside the corporation.
- *Online*. Hackers are harvesting a boom of vital information online from careless users. The reliance on and excessive use of the Internet have resulted in people having several online accounts. Currently an average user has about four to five accounts including one for home use, one for work, and an additional one or two for social or professional organizations. With many accounts, as probably any reader may concur, one is bound to forget some passwords, especially the least used ones. To overcome this problem, users mistakenly use one password on several accounts. Hackers know this, and they regularly target these individuals with clever baits such as telling them they won lotteries or were finalists in sweepstakes where computers select winners or they have won a specific number of prizes in a lotto, where they were computer selected. However, in order to get the award, the user must fill in an online form, usually Web-based, and this transmits the password to the hacker. Hackers have used hundreds of tricks on unsuspecting users in order for them to surrender their passwords.
- *Dumpster diving* is now a growing technique of information theft not only in social engineering but more so in identity theft. The technique, also known as trashing, involves an information thief scavenging through individual and company dumpsters for information. Large and critical information can be dug out of dumpsters and trash cans. Dumpster diving can recover from dumpsters and trash cans individual social security numbers, bank accounts, individual vital records, and a whole list of personal and work-related information that gives the hackers the exact keys they need to unlock the network.
- *In person* is the oldest of the information-stealing techniques that predates computers. It involves a person physically walking into an organization's offices and casually checking out note boards, trash diving into bathroom trash cans and company hallway dumpsters, and eating lunches together and initiating conversations with employees. In big companies, this can be done only on a few occasions before trusted friendships develop. From such friendships, information can be passed unconsciously.
- *Snail mail* is done in several ways and is not limited only to social engineering but has also been used in identity theft and a number of other crimes. It has been in the news recently because of identity theft. It is done in two ways: the hacker picks a victim and goes to the Post Office and puts in a change of address form to a new box number. This gives the hacker a way to intercept all snail mail of the victim. From the intercepted mail, the hacker can gather a great deal of information that may include the victim's bank and credit card account numbers and access control codes and pins by claiming to have forgotten his or her password or pin and requesting a reissue in the mail. In another form, the hacker drops a bogus survey in the victim's mailbox offering baits of cash award for completing

a "few simple" questions and mailing them in. The questions, in fact, request far more than simple information from an unsuspecting victim.
- *Impersonation* is also an old trick played on unsuspecting victims by criminals for a number of goodies. These days the goodies are information. Impersonation is generally acting out a victim's character role. It involves the hacker playing a role and passing himself or herself as the victim. In the role, the thief or hacker can then get into legitimate contacts that lead to the needed information. In large organizations with hundreds or thousands of employees scattered around the globe, it is very easy to impersonate a vice president or a chief operations officer. Since most employees always want to look good to their bosses, they will end up supplying the requested information to the imposter.

Overall, social engineering is a cheap but rather threatening security problem that is very difficult to deal with.

4.3 Vulnerability Assessment

Vulnerability assessment is a process that works on a system to identify, track, and manage the repair of vulnerabilities on the system. The assortment of items that are checked by this process in a system under review varies depending on the organization. It may include all desktops, servers, routers, and firewalls. Most vulnerability assessment services will provide system administrators with:

- Network mapping and system fingerprinting of all known vulnerabilities
- A complete vulnerability analysis and ranking of all exploitable weaknesses based on potential impact and likelihood of occurrence for all services on each host
- Prioritized list of misconfigurations

In addition, at the end of the process, a final report is always produced detailing the findings and the best way to go about overcoming such vulnerabilities. This report consists of prioritized recommendations for mitigating or eliminating weaknesses, and based on an organization's operational schedule, it also contains recommendations of further reassessments of the system within given time intervals or on a regular basis.

4.3.1 Vulnerability Assessment Services

Due to the massive growth of the number of companies and organizations owning their own networks, the growth of vulnerability monitoring technologies, the increase in network intrusions and attacks with viruses, and worldwide publicity of such attacks, there is a growing number of companies offering system vulnerability services. These services, targeting the internals and perimeter of the system, Web-based applications, and providing a baseline to measure subsequent attacks against, include scanning, assessment and penetration testing, and application assessment.

4.3.1.1 Vulnerability Scanning

Vulnerability scanning services provide a comprehensive security review of the system, including both the perimeter and system internals. The aim of this kind of scanning is to spot critical vulnerabilities and gaps in the system's security practices. Comprehensive system scanning usually results in a number of both false positives and negatives. It is the job of the system administrator to find ways of dealing with these false positives and negatives. The final report produced after each scan consists of strategic advice and prioritized recommendations to ensure that critical holes are addressed first. System scanning can be scheduled, depending on the level of the requested scan, by the system user or the service provider, to run automatically and report by either automated or periodic e-mail to a designated user. The scans can also be stored on a secure server for future review.

4.3.1.2 Vulnerability Assessment and Penetration Testing

This phase of vulnerability assessment is a hands-on testing of a system for identified and unidentified vulnerabilities. All known hacking techniques and tools are tested during this phase to reproduce real-world attack scenarios. One of the outcomes of these real-life testings is that new and sometimes obscure vulnerabilities are found, processes and procedures of attack are identified, and sources and severity of vulnerabilities are categorized and prioritized based on the user-provided risks.

4.3.1.3 Application Assessment

As Web applications become more widespread and more entrenched into e-commerce and all other commercial and business areas, applications are slowly becoming the main interface between the user and the network. The increased demands on applications have resulted into new directions in automation and dynamism of these applications. As we saw in Chap. 6, scripting in Web applications, for example, has opened a new security paradigm in system administration. Many organizations have gotten sense of these dangers and are making substantial progress in protecting their systems from attacks via Web-based applications. Assessing the security of system applications is, therefore, becoming a special skills requirement needed to secure critical systems.

4.3.2 Advantages of Vulnerability Assessment Services

Vulnerability online services have many advantages for system administrators. They can, and actually always do, provide and develop signatures and updates for new vulnerabilities and automatically include them in the next scan. This eliminates the need for the system administrator to schedule periodic updates.

Reports from these services are very detailed not only on the vulnerabilities, sources of vulnerabilities, and existence of false positives, but they also focus on vulnerability identification and provide more information on system configuration that may not be readily available to system administrators. This information alone goes a long way in providing additional security awareness to security experts about additional avenues whereby systems may be attacked. The reports are then encrypted and stored in secure databases accessible only with the proper user credentials. This is because these reports contain critically vital data on the security of the system and they could, therefore, be a pot of gold for hackers if found. This additional care and awareness adds security to the system.

Probably, the best advantage to an overworked and many times resource-strapped system administrator is the automated and regularly scheduled scan of all network resources. They provide, in addition, a badly needed third-party "security eye," thus helping the administrator to provide an objective yet independent security evaluation of the system.

Exercises

1. What is a vulnerability? What do you understand by a system vulnerability?
2. Discuss four sources of system vulnerabilities.
3. What are the best ways to identify system vulnerabilities?
4. What is innovative misuse? What role does it play in the search for solutions to system vulnerability?
5. What is incomplete implementation? Is it possible to deal with incomplete implementation as a way of dealing with system vulnerabilities? In other words, is it possible to completely deal with incomplete implementation?
6. What is social engineering? Why is it such a big issue yet so cheap to perform? Is it possible to completely deal with it? Why or why not?
7. Some have described social engineering as being perpetuated by our internal fears. Discuss those fears.
8. What is the role of software security testing in the process of finding solutions to system vulnerabilities?
9. Some have sounded an apocalyptic voice as far as finding solutions to system vulnerabilities. Should we take them seriously? Support your response.
10. What is innovative misuse? What role does it play in the search for solutions to system vulnerabilities?

Advanced Exercises

1. Why are vulnerabilities difficult to predict?
2. Discuss the sources of system vulnerabilities.
3. Is it possible to locate all vulnerabilities in a network? In other words, can one make an authoritative list of those vulnerabilities? Defend your response.
4. Why are design flaws such a big issue in the study of vulnerability?

5. Part of the problem in design flaws involves issues associated with software verification and validation (V&V). What is the role of V&V in system vulnerability?

References

1. Pipkin D (2000) Information security: protecting the global enterprise. Prentice Hall PTR, Upper Saddle River
2. Pethia RD. Information technology—essential but vulnerable: how prepared are we for attacks? http://www.cert.org/congressional_testimony/Pethia_testimony_Sep26.html
3. Kizza JM (2003) Ethical and social issues in the information age, 2nd edn. Springer, New York
4. Hurley J, Hemmendinger E. Open source and Linux: 2002 poster children for security problems. http://www.aberdeen.com/ab_abstracts/2002/11/11020005.htm
5. Palumbo J. Social engineering: what is it, why is so little said about it and what can be done? SANS, http://www.sans.org/rr/social/social.php
6. Granger S. Social engineering fundamentals, part I: Hacker Tactics. http://www.securityfocus.com/infocus/1527