Assignment 5

Nilesh Raj(2017012599)

Q1- What do you mean by programming for security?

Ans- Secure coding is the practice of developing computer software in a way that guards against the accidental introduction of security vulnerabilities. Defects, bugs and logic flaws are consistently the primary cause of commonly exploited software vulnerabilities. Through the analysis of thousands of reported vulnerabilities, security professionals have discovered that most vulnerabilities stem from a relatively small number of common software programming errors. By identifying the insecure coding practices that lead to these errors and educating developers on secure alternatives, organizations can take proactive steps to help significantly reduce or eliminate vulnerabilities in software before deployment.

Applications:

1- Securing a Desktop

Before understanding ways to secure a desktop, it is important to understand the possible threats to our desktop of the main threats to any desktop is its confidentiality. Unauthorized users should not be able to access a desktop and its workstations directly or through a network.

2- Securing Web Applications

When we think about threats to web applications, we often think about login security and password thefts. But there are many other threats that programmers need to be aware of. The threats and their repercussions are so severe that a non-profit organization called Online Web Application Security Project (OWASP) was set up to identify and counter common threats.

Q2- Explain the concept of Windows OS vulnerabilities?

Ans:- By understanding Windows based vulnerabilities, organizations can stay a step ahead and ensure information availability, integrity, and confidentiality.

Listed below are the Top 10 Windows Vulnerabilities:

1. Web Servers –

   Misconfigurations, product bugs, default installations, and third-party products such as php can introduce vulnerabilities.

2. Microsoft SQL Server –

   Vulnerabilities allow remote attackers to obtain sensitive information, alter database content, and compromise SQL servers and server hosts.

3. Passwords –

   User accounts may have weak, nonexistent, or unprotected passwords. The operating system or third-party applications may create accounts with weak or nonexistent passwords.

4. Workstations –

   Requests to access resources such as files and printers without any bounds checking can lead to vulnerabilities. Overflows can be exploited by an unauthenticated remote attacker executing code on the vulnerable device.

5. Remote Access –

   Users can unknowingly open their systems to hackers when they allow remote access to their systems.

6. Browsers –

   Accessing cloud computing services puts an organization at risk when users have unpatched browsers. Browser features such as Active X and Active Scripting can bypass security controls.

7. File Sharing –

   Peer to peer vulnerabilities include technical vulnerabilities, social media, and altering or masquerading content.

8. E-mail –

By opening a message a recipient can activate security threats such as viruses, spyware, Trojan horse programs, and worms.

9. Instant Messaging –

Vulnerabilities typically arise from outdated ActiveX controls in MSN Messenger, Yahoo! Voice Chat, buffer overflows, and others.

10. USB Devices –

Plug and play devices can create risks when they are automatically recognized and immediately accessible by Windows operating systems.

Q3- Describe any one vulnerability tool?

Ans- Nessus is a remote security scanning tool, which scans a computer and raises an alert if it discovers any vulnerabilities that malicious hackers could use to gain access to any computer you have connected to a network. It does this by running over 1200 checks on a given computer, testing to see if any of these attacks could be used to break into the computer or otherwise harm it.

Who would use a tool like this?

If you are an administrator in charge of any computer (or group of computers) connected to the internet, Nessus is a great tool help keep their domains free of the easy vulnerabilities that hackers and viruses commonly look to exploit.

What Nessus is NOT

Nessus is not a complete security solution, rather it is one small part of a good security strategy. Nessus does not actively prevent attacks, it is only a tool that checks your computers to find vulnerabilities that hackers COULD exploit. IT IS UP TO THE SYSTEM ADMINISTRATOR TO PATCH THESE VULNERABILITIES IN ORDER TO CREATE A SECURITY SOLUTION.

Why Nessus?

If you are familiar with other network vulnerability scanners, you might be wondering what advantages Nessus has over them. Key points include: - Unlike other scanners, Nessus does not make assumptions about your server configuration (such as assuming that port 80 must be the only web server) that can cause other scanners to miss real vulnerabilities.

Nessus is very extensible, providing a scripting language for you to write tests

Q4- What is linux OS vulnerability? How it can be managed?

Ans- Vulnerabilities:

#1 CVE-2017-18017

Linux Kernel netfilter: xt_TCPMSS

Vulnerability score: Critical — 9.8

Affected versions: Linux kernel before 4.11, and 4.9.x before 4.9.36

You might recognize this oldy-but-goody from our post covering top open source vulnerabilities in 2017. This comes as a reminder that vulnerabilities won't just go away if they are not attended to. Organizations that are still using this vulnerable version need to remediate before the hackers locate it.

This component sits on the Linux kernel, and helps filter network communication by defining the maximum segment size allowed for accepting TCP headers. These controls are crucial to avoid overflow.

#2 CVE-2017-18202

mm/oom_kill.c file

Vulnerability score: Critical — 9.8

Versions: before 4.14.4

This vulnerability lies in the mm/oom_kill.c file in the Linux kernel, a file that helps us kill a process when memory runs low. Vulnerable versions of the file might mishandle gather operations, opening the door to DoS attacks, possibly triggering a copy_to_user call within a certain time window.

Happily, the Linux kernel community has got us covered. You can learn more about the vulnerability and its remediation here, here and here.

#3 CVE-2017-15126

fs/userfaultfd.c

Vulnerability score: High — 8.1

Versions: before 4.13.6.

The security issue in this kernel vulnerability is local memory corruption. More specifically, this is a use-after-free vulnerability, a specific type of memory corruption bug that can be exploited to execute arbitrary code or even enable full remote code execution.

In this case, the flaw was discovered in fs/userfaultfd.c in Linux kernel versions preceding 4.13.6, and is related to the handling of fork failure when dealing with event messages.

#4 CVE-2018-1000026

bnx2x network card driver

Vulnerability score: High — 7.7

Versions: at least v4.8 onwards

This one is an insufficient input validation vulnerability affecting the bnx2x network card driver in the Linux kernel from version 4.8 and above. An attacker can exploit the vulnerability by passing on a very large and specially crafted packet to the bnx2x card from an untrusted guest virtual machine, knocking it offline and causing a DoS to the targeted system.

#5 CVE-2018-8822

NCPFS implementation

Vulnerability score: High — 7.8

Versions: through 4.15.11, and in drivers/staging/ncpfs/ncplib_kernel.c in the Linux kernel 4.16-rc through 4.16-rc6

This Linux kernel security issue is in vulnerable versions of the NCPFS implementation in the Linux kernel, because it doesn't perform sufficient boundary-checks on user-supplied data, or validate reply lengths from the server.

How they can be managed:

Software upgrades

The most obvious corrective measure is keeping software packages up-to-date. This way when a security leak has been found, and the vendor released an update, it can be fixed quickly. Having a good patch management process in place is key.

Upgrading on a regular basis keeps your system protected

Install security updates only?

If you rather don't want to install all updates, you can opt for doing only the security updates. Depending on your Linux distribution, determine if you can enable this. Sometimes we have to script a little bit to achieve the same, but it is doable. For example, with Debian and Ubuntu, we can filter out the security-related repositories, and only upgrade packages which are referred to in our custom file.

Benefits

The big benefit of only applying security updates is that you solve vulnerabilities, yet don't update or upgrade the whole system. This way it is much quicker and easier to detect if something does not work after the change. It also allows you to use a double schedule, like installing security patches daily, the remaining patches one a week or month.

Test before you patch

If you or your colleagues are scared of applying patches, then create your own test system. Do a maximum installation of your favorite Linux distribution and install all types of software, you would otherwise find also on other systems within your network. Have this system perform security updates very regularly and report on any issues.

Detecting issues with patches can be as simple as adding the system to your monitoring system. Have it test all the processes you expect it to be running. If suddenly something stops, you can quickly determine if this is the result of an applied patch.

Automatic security upgrades

Some Linux distributions allow also for automatic patching, so you don't have to write your own script if automation important. For Debian and Ubuntu system there is the unattended-upgrades utility. Configure it to do security patches only and let it run daily.