CERT'S PODCASTS: SECURITY FOR BUSINESS LEADERS: SHOW NOTES

How to More Effectively Manage Vulnerabilities and the Attacks that Exploit Them

Key Message: Deploy vulnerability exploit prevention and mitigation techniques to thwart attacks and manage the arms race.

Executive Summary

To reduce the security risks posed by software vulnerabilities, the CERT Vulnerability Analysis Team strives to address both the number of vulnerabilities in software that is being developed and the number of vulnerabilities in software that is already deployed. CERT's vulnerability analysis work addresses two areas: [1]

- vulnerability discovery: identify and reduce the number of new vulnerabilities before the software is deployed by helping engineers understand how vulnerabilities are created and found
- vulnerability remediation: deal with existing vulnerabilities in deployed software

In this podcast, Art Manion, a member of CERT's Vulnerability Analysis Team, discusses how security vulnerabilities have evolved and how business leaders can more effectively manage them.

PART 1: VULNERABILITY COUNTS LIKELY LOW BY AN ORDER OF MAGNITUDE

Definitions

Software vulnerabilities, as defined by CERT, include the following:

- implementation defects: a coding error that can be exploited by an attacker
- design time: for example, incorrect design of a protocol or not anticipating a certain type of attack
- changing environmental conditions: vulnerabilities in the operational environment within which the software executes

A software vulnerability allows an attacker to break in, steal data, tamper with data, and deny service.

Impact of Vulnerabilities

Given that we are an internet-connected society, attackers can use this environment to exploit vulnerabilities to commit crimes. An attacker can use a series of vulnerabilities (a kill chain) to realize their goals.

CERT studies vulnerabilities to try to prevent them, tactically through patches and strategically through managing vulnerabilities and establishing policies to mitigate their effects.

Targeted Attacks

Attackers do research to find a person in a specific company who is more likely to respond to the methods used by the attacker. For example, a phishing email that is well-crafted may cause the targeted user to open an attached document or click on a link.

One common category of exploited vulnerabilities is those found in a PDF reader, Adobe Flash, or a Microsoft Office-formatted document. When the user opens the document, the attacker exploits a vulnerability and installs tools on the user's system – one additional step towards their goal.

This example includes a social engineering aspect (phishing email) and a vulnerability and exploit aspect (document that exploits a vulnerability on the user's system).

Well-crafted, legitimate appearing emails from a financial institution are another common approach. A class of attacks in 2009 called Aurora is an example of more sophisticated attacks. "According to McAfee, the primary goal of the attack was to gain access to and potentially modify source code repositories at high tech, security and defense contractor companies," such as RSA, Lockheed Martin, and Google.

Even companies with robust security are vulnerable to a motivated and capable attacker who is able to exploit an unpatched or zero-day vulnerability.

Counting Vulnerabilities

CERT stopped counting vulnerabilities in 2008. Based on public sources (mailing lists, databases, blog posts, Twitter, vendor disclosures, Microsoft and Oracle patch releases), an estimate of new annual vulnerabilities is between 8,000 – 10,000.

CERT is currently focusing on vulnerability discovery through methods such as fuzz testing. Based on these methods, CERT estimates that the annual number of new vulnerabilities is likely low by at least one order of magnitude.

The number of bugs found by the tools CERT is using, which cause crashers (crashing test cases resulting from fuzz testing tools), could easily overwhelm a vendor. Many bugs can introduce vulnerabilities that could be exploited.

Classic Defense Problem

Network and system owners need to defend all points of entry; an attacker only needs one to get in.

PART 2: FUZZ TESTING; SECURITY HYGIENE; VULNERABILITY INTELLIGENCE

Fuzz Testing and Vulnerability Discovery

The purpose of fuzz testing is to determine the robustness of a software program. For example, a properly formed PDF file is provided as an input to a program. It opens properly and displays correctly with Adobe Reader or another PDF reading program. This is the control case.

The input file is modified in some arbitrary, random way (called dumb fuzzing) such as flipping bits, changing some bytes, adding something, or taking something out. The modified file is provided as an input to the program. This testing loop is executed repeatedly using automated test tools.

The interesting case is when the program tries to open the file and then crashes. It's typical to go from 1 million test cases to 1,000 crashers to 100 crashers that look like they might be caused by software vulnerabilities. These are often reported to vendors as potentially serious security bugs.

Attackers have access to these same tools so can use the same approach to discover exploitable vulnerabilities. The vendor corrects certain vulnerabilities and the attacker finds new ones. This is another example of the classic defense/arms race problem.

Effectively, there is an endless supply of vulnerabilities, which is bad news for network and system defenders.

Security Hygiene

There are series of standard defense-in-depth techniques that are also often referred to as security hygiene practices. These include:

- patching systems, and keeping track of which systems are patched
- change management
- inventory management
- scanning for vulnerabilities
- antivirus systems
- intrusion detection systems (IDS)
- intrusion prevention systems (IPS)
- implementing the principle of least privilege

IDS, IPS, and anti-virus help fill the gaps between a patched vulnerability and one that may be exploited for which a patch does not yet exist. These techniques rely on detection signatures, which mean that the vulnerability is known. These do not help protect against unknown vulnerabilities.

Other techniques include white listing, i.e., only allow access to entities (users, applications, etc.) that you know should be allowed.

Vulnerability Intelligence (see Resources)

Given an estimated 8,000 new vulnerabilities each year, organizations need to ask how many of these truly matter. It is unlikely that all 8,000 will be exploited even if you have all of the systems and applications that contain these vulnerabilities.

Typically, there are a couple of dozen that are widely exploited. Often these can be managed by deploying a handful of patches and a handful of browser configuration settings such as turning off Java and Flash.

CERT advocates removing Flash support and 3D rendering support from Adobe Reader as these are common attack vectors and are features of PDF readers that are not widely used.

Taking such actions removes an entire class of attack. Finding configuration options and mitigation steps that remove sets of vulnerabilities is one aspect of using vulnerabilities intelligence.

Applying this type of thinking to the vulnerabilities that are most likely is one effective approach for managing and prioritizing vulnerabilities.

PART 3: GAME CHANGERS THAT CAN THWART ATTACKS

Exploit Mitigation/Exploit Prevention

These types of operating-system-level techniques are based on the premise that organizations are not going to be able to patch or know about all vulnerabilities before they are exploited.

Such techniques make it very difficult for the attacker to do something useful even when they do get in. In other words, these techniques prevent attackers from executing arbitrary code, such as running a program of their choice, installing a back door, installing a Trojan horse, installing remote control software, or installing something that is going to exfiltrate an organization's data.

These techniques have been known in the research community for years. They are available for Linux, BSD, Apple OSX, and Microsoft Windows operating system.

Managing the Arms Race: Microsoft's Enhanced Mitigation Experience Toolkit

EMET is one example of a small utility that exposes these mitigation techniques to the system user or administrator. Turning on EMET tools buys a great deal in terms of defending against unpatched, zero-day attacks.

One example of how these tools can be effective is in defending against a typical stack buffer overflow. An attacker provides too much input, it flows past what the program has allotted in memory to store such input, and the attacker can overwrite parts of the program that control its execution. By doing so, they can then run whatever software they store past the end of stack memory; *advantage attacker*.

The first EMET technique is called Data Execution Prevention (DEP) or No Execute (NX). This technique marks stack memory as not executable and it is enforced by hardware and software. If DEP is turned on, an attacker installing a program in stack memory cannot execute it; *advantage defender*.

To get around DEP, attackers use a technique to find somewhere else in program memory where they can execute their attack code. They get into the system, install their program at a known location, jump to it from the stack, and execute their program; *advantage attacker*.

The second EMET technique is called Address Space Layout Randomization (ASLR). Given the attacker is trying to find a known, consistent location in memory to install and run their code, ASLR randomizes where programs are loaded in memory every time a program runs or every time a system boots. Now the attacker can no longer jump to a known location. Attackers want their exploits to be reliable and repeatable using automated techniques, at least 80% to 90% of the time. ASLR prevents this; *advantage defender*.

To get around ASLR, attackers get into a system and look for a ROP (Return Oriented Programming) gadget – a handful of instructions that exist in a legitimate program, which can be used by an attacker to execute their code; *advantage attacker*.

ASLR is only effective if everything loaded in memory is randomized; *advantage defender*. Otherwise an attacker can find the gadgets that they need to execute.

Impacts of Using Exploit Mitigation Techniques

Given today's computing power, techniques such as EMET do not impact system performance. They do not introduce enough overhead to be noticed.

The biggest impact is applicability compatibility. Older applications, those that were not designed with DEP and ASLR in mind, may not run or may crash when these techniques are enabled. The mitigation approach is to do extensive testing.

Once test cases can run successfully, it is critical to deploy EMET techniques.

Resources

[1] CERT Vulnerability Analysis website

US-CERT Vulnerability Notes database

CERT/CC blog

- AMD video drivers prevent the use of the most secure setting for Microsoft's Exploit Mitigation Experience Toolkit (EMET), June 6, 2012
- CERT Basic Fuzzing Framework 2.5 Released, April 30, 2012.
- CERT Failure Observation Engine 1.0 Released, April 23, 2012.
- A Security Comparison: Microsoft Office vs. Oracle Openoffice, April 13, 2011. (Links to Microsoft's EMET information)

CERT podcasts

- The Power of Fuzz Testing to Reduce Security Vulnerabilities, May 2010.

Managing Security Vulnerabilities Based on What Matters Most, July 2008.

Jeffers, David. "A Sinister New Breed of Malware is Growing." CIO.com, August 17, 2012. Contains links to descriptions for Stuxnet, Duqu, Flame, and Gauss.

Microsoft. The Enhanced Mitigation Experience Toolkit.

TechNet blog. On the effectiveness of DEP and ASLR, December 8, 2010.

Vulnerability intelligence

- Chuvakin, Anton. "Threat and Vulnerability Intelligence." ISSA, November 2003.
- Guido, Dan. "The Exploit Intelligence Project." iSEC Partners, April 2011.
- Guido, Dan. "Mobile Exploit Intelligence Project." Trail of Bits and iSEC Partners, April 2012.