Advanced Network Security

Detecting and Understanding Vulnerabilities

Thomas Martin

t.martin@mmu.ac.uk

February 27, 2018

Outline

1 Introduction

2 Vulnerability Mapping Tools

Outline

1 Introduction

2 Vulnerability Mapping Tools

Introduction

Vulnerability mapping (or vulnerability assessment) is a process of identifying and analysing the critical security flaws in the target environment. This is clearly a vital step in any vulnerability management program. All security controls of an IT infrastructure need to be regularly analysed for known and unknown vulnerabilities.

If it is being performed as part of a penetration test, it occurs after the gathering information stages have been completed. The next step is to investigate the vulnerabilities that may exist in the target infrastructure which could lead to a compromise of the target and violation of the confidentiality, integrity, and availability of a business system.

Assessment Procedures

In this section, we will look at the different types of vulnerabilities, the taxonomies used to classify them, as well as security tools that can assist in finding them.

There are automated vulnerability assessment procedures as well as manual. The temptation may be to rely entirely on the automated procedures. However, these may produce false positives and false negatives. A lack of knowledge on the part of the auditor, or the absence of any technology relevant assessment tools, are likely to decrease the chances of successful penetration testing. Automated tools often fail to identify logic errors, undiscovered vulnerabilities, unpublished software vulnerabilities, and anything that depends on the human element of security.

Terminology

Reminder:

Definition

A **vulnerability** is a security weakness found in a system which can be used by an attacker to perform unauthorized operations.

Definition

An **exploit** is a piece of code (proof-of-concept) written to take advantage of a vulnerability or bug in an automated fashion.

Types of Vulnerabilities

There are three different classes of vulnerabilities. **Design vulnerabilities** are due to weaknesses in the software specifications. **Implementation vulnerabilities** are the technical security glitches found in the code of a system. **Operational vulnerabilities** are those which may arise due to improper configuration and deployment of a system in a particular environment.

The worst of these, from the defender's point of view, is a design vulnerability. To fix these, changes must be introduced into the security requirements. The subsequent changes to the design and implementation can take considerable time and effort to apply.

Types of Vulnerability

Another way to classify vulnerabilities is in terms of how they can be exploited. A **local vulnerability** is where an attacker requires local access in order to trigger the vulnerability. This is used where an attacker already has the ability to execute code with limited permission, and wishes to increase his privileges to gain unrestricted access (*privilege escalation*).

When an attacker has no prior access to a system, but is able to trigger the execution of a piece of code over the network, this is known as a **remote vulnerability**. This type allows an attacker to gain remote access to the computer system without facing any physical or local barriers.

Vulnerability Taxonomy

There have been a number of attempts in the past few years to categorize all vulnerabilities. This is a daunting task, and no one standard has been accepted. Challenges include the fact that a single vulnerability may fall into more than one category, as well as the variety of different technologies involved.

Security Taxonomy	**Link**
Common Weakness Enumeration (CWE)	https://cwe.mitre.org/data/index.html
Common Vulnerabilities and Exposures (CVE)	https://cve.mitre.org/cve/index.html
Seven Pernicious Kingdoms	https://cwe.mitre.org/documents/sources/SevenPerniciousKingdoms.pdf
OWASP Top 10	https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
WASC Threat Classification	http://projects.webappsec.org/w/page/13246978/Threat%20Classification

Common Weakness Enumeration

The Common Weakness Enumeration (CWE) is a formal list of software weakness types. The project (sponsored by Mitre Corporation) aims to:

- Serve as a common language for describing software security weaknesses in architecture, design, or code.
- Serve as a standard measuring stick for software security tools targeting these weaknesses.
- Provide a common baseline standard for weakness identification, mitigation, and prevention efforts.

The CWE effort is attempting to mature the code security assessment industry. Those acquiring software wish to have assurance that it has been reviewed for known types of security flaws. Those providing software assurance tools and services can describe their capabilities in terms of the different CWEs.

Common Weakness Enumeration

The CWE List is offered in a number of different ways. The Dictionary view[1] is a complete enumeration of the contained weaknesses, while the Graphical View [2] allows a user to better understand individual weaknesses through their broader context and relationships.

Each entry contains a great deal of information on the weakness, such as: Description, Time of Introduction, Applicable Platforms, Modes of Introduction, Common Consequences, Likelihood of Exploitation, Enabling Factors for Exploitation, Detection Methods, Demonstrative Examples, Observed Examples (relating to CVEs), Potential Mitigations, Relationships, etc.

[1] http://cwe.mitre.org/data/lists/2000.html
[2] http://cwe.mitre.org/data/index.html

Common Vulnerabilities and Exposures

The Common Vulnerabilities and Exposures (CVE) system provides a reference-method for publicly known information security vulnerabilities and exposures.

CVE Identifiers are unique, common identifiers for publicly known information security vulnerabilities in publicly released software packages. CVEs are assigned by a CVE Numbering Authority (CNA), of which there are three types:

1. The MITRE Corporation functions as Editor and Primary CNA
2. Various CNAs assign CVE entries for their own products
3. Red Hat also provides CVE numbers for Open Source projects that are not a CNA

CVEs can exist for any publicly released software, including commercial software, betas and pre-releases, but not custom built software or web services.

Common Vulnerabilities and Exposures

Each entry in the CVE database has several fields:
CVE-ID: The actual CVE identifier, i.e. CVE-2012-2234. Note that from January 2014, the syntax changed to **CVE + Year + Arbitrary Digits**.
Description: Text description of the issue (or placeholder when the issue is under embargo).
References: URLs and other information for the issue.
Date Entry Created: The date the entry was created (by MITRE, not the CNA).
Phase/Votes/Comments/Proposed (All legacy, no longer used).
The CVE database can be searched by downloading the master copy from MITRE, or online at the NIST site[3].

[3] `http://web.nvd.nist.gov/view/vuln/search`

Outline

1 Introduction

2 Vulnerability Mapping Tools

Nessus

Nessus was created by Renaud Deraison in 1998 as a free remote security scanner. In 2005, his company (Tenable Network Security) changed it to a proprietary (closed source) product. A free version is still available for home use, but there is a charge for the full-featured versions.

The basic tasks it performs are:

- Scan for vulnerabilities that would allow remote access
- Misconfigurations
- Default or common passwords
- Denial of service attacks against the TCP/IP stack

Nessus

Features in the commercial version include the ability to perform scans to determine compliance with PCI, CIS, FDCC, and other standards, technical support, SCADA vulnerability testing, patch auditing, searching for sensitive data such as credit card details, etc.

Nessus runs on both Unix and Windows systems, and is managed from the browser. As well as port scanning, it has thousands of vulnerability checks called *plugins*. These are obtained from a feed provided by Tenable.

Open Vulnerability Assessment System (OpenVAS)

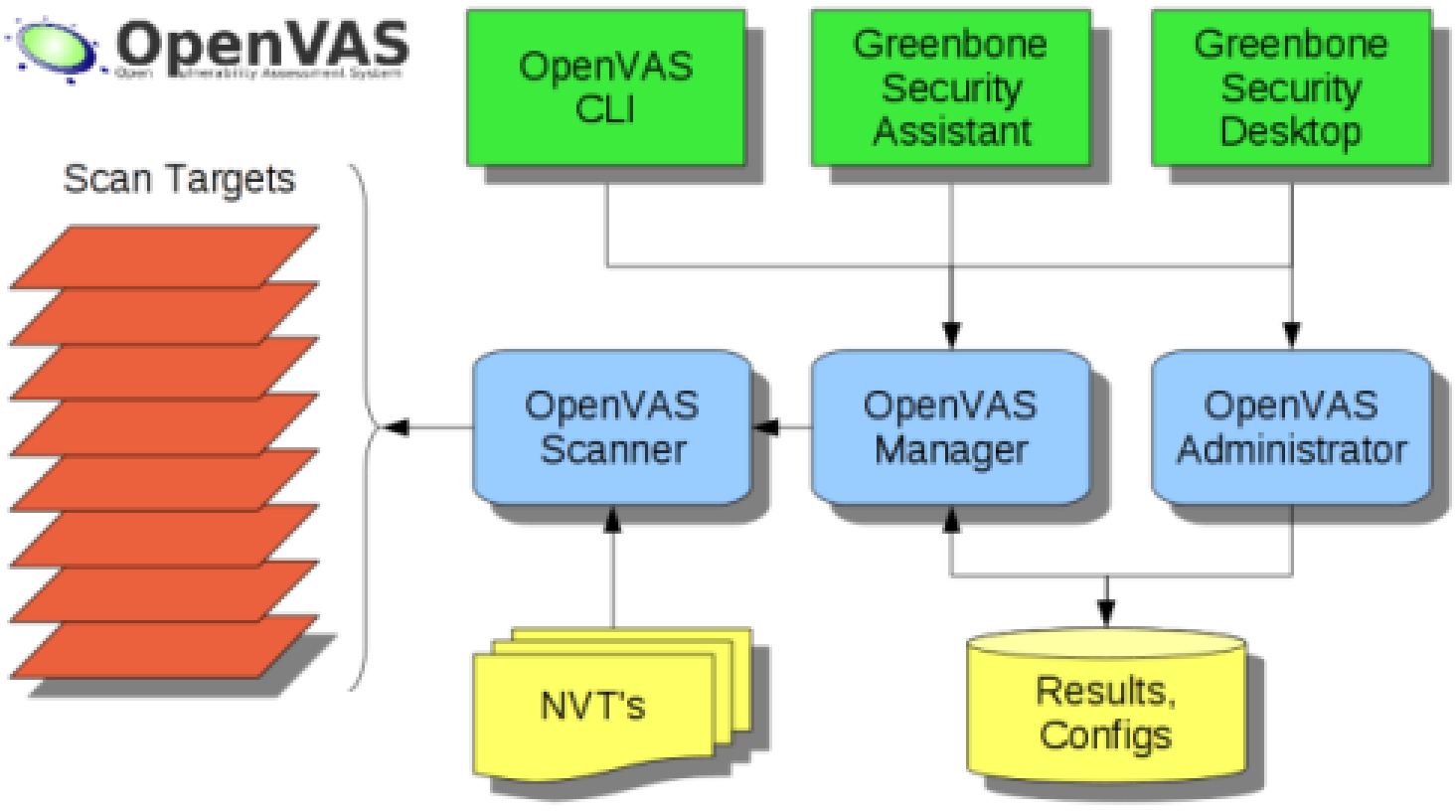
OpenVAS is a collection of security tools and services that provide a platform for vulnerability assessment. It began as a fork of the previously open source Nessus tool (and was named GNessUs). It has been developed for a client-server architecture, where the client requests a specific set of network vulnerability tests against its target from the server. OpenVAS has a number of different components:

- **OpenVAS Scanner** manages the execution of Network Vulnerability Tests (NVT), which can be updated on a daily basis via NVT Feeds.
- **OpenVAS Client** is the desktop/CLI-based tools to control the scan execution using the OpenVAS Transfer Protocol.

OpenVAS

- **OpenVAS Manager** provides central service for vulnerability scanning. Stores configurations and scan results, performs scheduled scans, report generation, scan results filtering, and aggregation activity.
- **Greenbone Security Assistant** is a web service that runs on top of OMP (OpenVAS Management Protocol). Allows the user to configure, manage, and administer the scanning process.
- **OpenVAS Administrator** is responsible for handling the user administration and feed management.

OpenVAS


OpenVAS
OpenVAS CLI
Greenbone Security Assistant
Greenbone Security Desktop
Scan Targets
OpenVAS Scanner
OpenVAS Manager
OpenVAS Administrator
NVT's
Results, Configs

OpenVAS

There are several tools integrated within the OpenVAS system.

AMap	Application protocol detection tool
ike-scan	IPsec VPN scanning, fingerprinting, and testing
Ldapsearch	Extract information from LDAP dictionaries
Nikto	Web server assessment tool
NMap	Port scanner
Ovaldi	Open Vulnerability and Assessment Language interpreter
pnscan	Port scanner
Portbunny	Port scanner
Seccubus	Automates the regular OpenVAS scans
Slad	Security Local Auditing Daemon tools include John-the-Ripper (JTR), Chkrootkit, ClamAV, Snort, Logwatch, Tripwire, LSOF, TIGER, TrapWatch, LM-Sensors
Snmpwalk	SNMP data extractor
Strobe	Port scanner
w3af	Web application attack and audit framework

Fuzzy Analysis

Fuzzy analysis is a sophisticated software testing technique used by auditors and developers to test applications against unexpected, invalid, and random sets of data input. The way the application reacts is observed and recorded, in particular any crash or exception thrown. This is one way of uncovering major vulnerabilities in the software. Problems that can be found include:

- Buffer overflows
- Format strings
- Code injections
- Dangling pointers
- Denial of Service conditions

Fuzzy Analysis

Any application that accepts untrusted sources of data input should consider them to be insecure and inconsistent. For example, the trust boundary between an application and an Internet user is unpredictable. So all data inputs should be fuzzed and verified against known and unknown vulnerabilities.

Fuzzy analysis is a relatively simple and effective solution that can be incorporated into a quality assurance and security testing process. It is also known as robustness testing or negative testing.

Fuzzy Analysis

There are six common steps taken in a typical fuzzy analysis:

1. Identify the target
2. Identify the inputs
3. Generate fuzz data
4. Execute fuzz data
5. Monitor the output
6. Determine the exploitability

Fuzzy Analysis

There exists a number of tools for performing fuzzy analysis:

BED (Bruteforce Exploit Detector) is designed to fuzz plaintext protocols. It supports ftp, smtp, pop, http, irc, imap, pjl, lpd, finger, socks4, and socks5.

JBroFuzz if a platform for web application fuzzy testing. The program takes a URL and a selection of part of the web request to fuzz. The auditor can either manually craft the request, or predefined payloads can be used.

In Class Task

Suppose you are given the responsibility of ensuring the safe web-browsing of all citizens in a country.

- What are some common attacks that threaten everyday web users?
- Each of these attacks are exploiting some vulnerabilities. Place each vulnerability in to one of the "Design", "Implementation", or "Configuration" category.
- What options are their to ensure (or even encourage) safe browsing at the national level?

Relevant texts

Stallings, Cryptography and Network Security: Principles and Practice, Pearson, Chapter 1.

Simpson and Backman, Hands-On Ethical Hacking and Network Defense, Cengage Learning PTR, Chapter 1.

Weidman, Penetration Testing: A Hands-On Introduction to Hacking, No Starch Press, Chapters 1 and 8.

Thank you

Any Questions?