SEZG566/SSZG566

Secure Software Engineering

Security Testing

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

T V Rao

DISCLAIMERS

- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Testing for Security

Testing Strategy

The strategy provides a road map that describes the steps to be taken, when, and how much effort, time, and resources will be required

A strategy for software testing integrates the design of software test cases into a well-planned series of steps that result in successful development of the software

The strategy incorporates test planning, test case design, test execution, and test result collection and evaluation

Testing begins at the component level and work outward toward the integration of the entire computer-based system

Different testing techniques are appropriate at different points in time

Software Testing Axioms

- It is impossible to test a program completely.
- Software testing is a risk-based exercise.
- Testing cannot show the absence of bugs.
  - The Pesticide Paradox
    - In 1990, Boris Beizer, coined the term *pesticide paradox* to describe the phenomenon that the more you test software, the more immune it becomes to your tests
- The more bugs you find, the more bugs there are.
- Not all bugs found will be fixed.
- It is difficult to say when a bug is indeed a bug.
- Specifications are never final.
- Software testers are not the most popular members of a project.
- Software testing is a disciplined and technical profession

Software Testing, 2nd Edition By Ron Patton

Software Testing Key Issues (SWEBOK)

Dynamic: The input value alone is not always sufficient to specify a test, since a complex, nondeterministic system might react to the same input with different behaviors, depending on the system state.

Finite: Even in simple programs, so many test cases are theoretically possible that exhaustive testing could require months or years to execute.

Selected: How to identify the most suitable test set under given conditions is a complex problem; in practice, risk analysis techniques and software engineering expertise are applied.

Expected: It must be possible, although not always easy, to decide whether the observed outcomes of program testing are acceptable or not; otherwise, the testing effort is useless.

Security Testing (SWEBOK)

- Security testing is focused on the verification that the software is protected from external attacks.
- Security testing verifies the confidentiality, integrity, and availability of the systems and its data.
- Security testing includes verification against misuse and abuse of the software or system (negative testing).

Security Testing Myth & Reality

Myth :

In the security industry people frequently test against a set of mental criteria that are neither well defined nor complete. As a result of this, many outsiders regard security testing as a black art.

Reality :

It is possible for people without in-depth security knowledge to make impactful security testing.

Penetrate and Patch

A number of software tools exist to support *post facto* security analysis of installed computer systems. Examples include

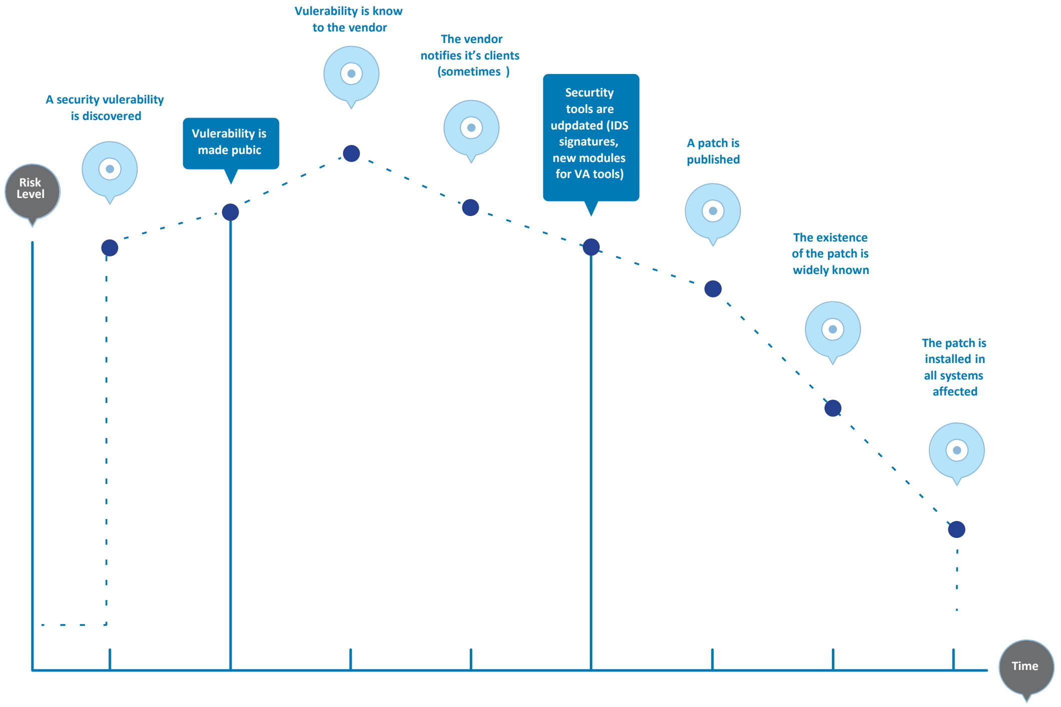
- Nmap (Network Mapper) a free and open-source network scanner.
- Burp Intruder is a powerful tool for automating customized attacks against web applications.
- Metasploit scans vulnerabilities backed by open-source database of known exploits.

Such tools are reactive.

- Security analysis needs to be performed as part of the software development process, before software is released.
- Crackers often know about vulnerabilities before system administrators
- System administrators have neither the time nor the inclination to patch software if they have not noticed any security breaches.
- While a patch may close one security hole, it may simultaneously open up others.

www.cigital.com

Window of Vulnerability (OWASP)


Risk Level
A security vulerability is discovered
Vulerability is made pubic
Vulerability is know to the vendor
The vendor notifies it's clients (sometimes)
Securtity tools are udpdated (IDS signatures, new modules for VA tools)
A patch is published
The existence of the patch is widely known
The patch is installed in all systems affected
Time

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Source Code Analyzers

Source Code Analysis Tools

Source Code Analysis Tools (security analyzers) are automated tools for helping analysts find security-related problems in software

- use data- and control-flow analysis to find subtler bugs and to reduce false alarms

Some vulnerabilities (e.g. use of strcpy()) can be detected with high accuracy, others are harder to detect, and, in fact, one can always devise vulnerabilities that are undetectable altogether.

Tools have tradeoff between false alarms (also known as false positives) and missed vulnerabilities (also known as false negatives)

- can be configured to make a tool more sensitive (decreasing false negatives while increasing false positives) or make it less sensitive (increasing false negatives while decreasing false positives)

Some of the tools are listed at https://resources.infosecinstitute.com/topic/secure-coding-top-15-code-analysis-tools/

Capabilities of Security Analyzers

- Examining Calls to Potentially Insecure Library Functions
- Detecting Bounds-Checking Errors and Scalar Type Confusion
- Detecting Type Confusion Among References or Pointers
- Detecting Memory Allocation Errors
- Detecting Vulnerabilities that Involve Sequences of Operations (Control-Flow Analysis)
- Data-Flow Analysis
- Pointer-Aliasing Analysis
- …

Check Calls to Potentially Insecure Library Functions

This security-scanning capability can encompass several components:

- **A database of vulnerable library calls** is the heart of security scanning technology. The vulnerability database must be up to date, and would have to be constantly updated as well to remain relevant.
- **The ability to preprocess source code** is important for C/C++ analyzers, because it lets the analyzer see the same code that will be seen by the compiler. Without this capability there are numerous ways to deceive the analyzer. Many analyzers use heuristics to approximate the functionality of a preprocessor.
- **Lexical analysis** is the process of breaking a program into tokens prior to parsing. Lexical analysis is necessary to reliably distinguish variables from functions and to identify function arguments. These functions can also be performed with heuristics—at the cost of some reliability, however.

Detecting Bounds-Checking Errors and Scalar Type Confusion

Vulnerabilities occur when scalar assignments transparently *change* the value being assigned e.g.

- integer overflow: an integer variable overflows and becomes negative
- integer truncation: an integer value is truncated while being cast to a data type with fewer digits
- unsigned underflow: an unsigned integer value underflows and becomes large

one of these issues results in a vulnerability typically because the affected variable gives the size of a buffer

Detecting Type Confusion Among References or Pointers

Type confusion with pointers or references is a common source of bugs and may result in vulnerabilities unless the type confusion is detected at runtime

- In some cases, static type checking can identify reference type confusion

- Usually fail with a cast between incompatible types having a common superclass allowing (for example) methods written for one data type to be applied to a different data type leading to vulnerabilities

Detecting Memory Allocation Errors

- Memory corruption vulnerabilities can vary from one operating system to the next because the operating systems use different techniques for heap maintenance
- Heap corruption can arise if an attacker is able to overwrite information used to maintain the heap
- A number of circumstances can allow an attacker to corrupt this information. e.g.
  - a buffer overflow in an allocated chunk of memory
  - a double free.
  - a write to freed memory.

Vulnerabilities Involving Sequences of Operations (Control-Flow Analysis)

File accesses by a program can create vulnerabilities if done incorrectly; operations have to be carried out in the right order

- e.g., a C program first obtains a file handle to check certain properties of the file before it can access the file contents
- the mask governing permissions of newly created files must be set explicitly if a new file may be created
- integer ranges have to be checked before being used without any modification taking place between the time of check and time of use.

Security analyzers often look for specific library function calls and print a warning regardless of whether the operation in question is being carried out correctly, which causes noise

Control-flow analysis be used when some potentially dangerous operation must be preceded by precautionary measures, such as closing and reopening standard file descriptors in C before writing to them, or setting default file permissions before creating a new file

- e.g. a linux file descriptor hangs in system folder if not explicitly closed & stays vulnerable

Data-Flow Analysis

- Security analyzers use data-flow analysis primarily to reduce false positives and false negatives, e.g., many buffer overflows in real code are not exploitable because the attacker cannot control the data that overflows the buffer.
- The data-flow analysis that is often used in security-related applications is *taint analysis.*
  - A variable is tainted if its value can be influenced by a potential attacker
  - If a tainted variable is used to compute the value of a second variable, then the second variable also becomes tainted

Pointer-Aliasing Analysis

Pointer aliasing occurs when two pointers point to the same data

- The data that would be found by dereferencing one of the pointers can change even though the source code contains no mention of that pointer.

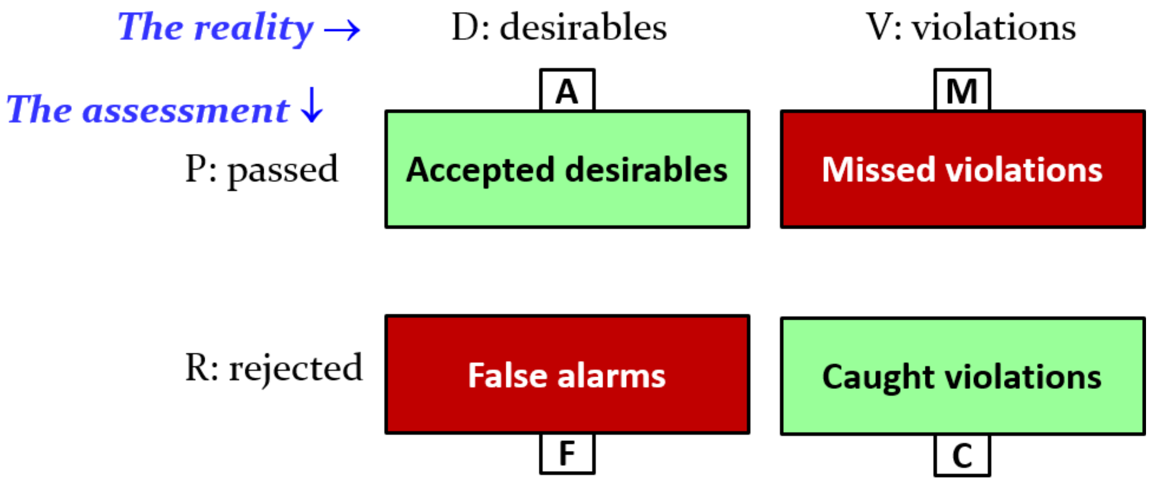
Pointer-aliasing analysis refers to any static technique that tries to solve this problem by tracking which pointers point to what locations

- Related to data flow analysis

Many programming languages allow them to be manipulated in arbitrary ways, for e.g., a loop that increments the value of a pointer until it points to a space character

- Remains a challenging part in static code analysis

Soundness vs Completeness of Source Code Analyzers



In two categories, marked in green, assessment is right: accepted desirables (A), rightly passed, and caught violations (C), rightly rejected.

In the other two, marked in red, the assessment is off the mark: missed violations (M), wrongly passed; and false alarms (F), wrongly accepted.

https://cacm.acm.org/blogs/blog-cacm/236068-soundness-and-completeness-with-precision/fulltext

Major weaknesses

- Many types of security vulnerabilities are very difficult to find automatically, such as authentication problems, access control issues, insecure use of cryptography, etc. However, tools of this type are getting better.
- High numbers of false positives (or false alarms).
- Frequently can't find configuration issues, since they are not represented in the code.
- Difficult to 'prove' that an identified security issue is an actual vulnerability.
- Many of these tools have difficulty analyzing code that can't be compiled. Analysts frequently can't compile code because they don't have the right libraries, all the compilation instructions, all the code, etc.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

White Box Security Testing

White Box Testing

White box testing consists testing with access to the source code

- it is a good practice to perform white box testing during the unit testing phase

White box testing requires knowing what makes software secure or insecure, how to think like an attacker, and how to use different testing tools and techniques.

- First tester comprehends and analyzes available design documentation, source code, and other relevant development artifacts, so knowing what makes software secure is a fundamental requirement.
- Second, tester creates tests that exploit software, a tester must think like an attacker.
- Third, to perform testing effectively, testers need to know the different tools and techniques available for white box testing

White Box Testing for Security

Data-Flow Analysis

Code-Based Fault Injection

Abuse Cases

Trust Boundaries Mapping

Code Coverage Analysis

Data-Flow Analysis

- The data-flow testing technique is based on investigating the ways values are associated with variables and the ways that these associations affect the execution of the program
  - focuses on occurrences of variables, following paths from the definition (or initialization) of a variable to its uses
- A data-flow analysis for an entire program involving all variables and traversing all usage paths may require immense computational resources
  - however, this technique can be applied for select variables
- The path and the usage of the data can help in identifying suspicious code blocks and in developing test cases to validate the runtime behavior of the software.

Code-Based Fault Injection

- The fault injection technique perturbs program states by injecting software source code to force changes into the state of the program as it executes.
  - Consists of non-intrusively inserting code into the software that is being analyzed and then compiling and executing the modified (or instrumented) software
- This technique forces non-normative behavior of the software, and the resulting understanding can help determine whether a program has vulnerabilities that can lead to security violations.
  - can be used to force error conditions to exercise the error handling code,
  - change execution paths,
  - input unexpected (or abnormal) data,
  - change return values, etc.

Abuse Cases

- Abuse cases help security testers view the software under test in the same light as attackers do.
  - can be used to develop innovative and effective test cases mirroring the way attackers would view the system
- The abuse case can also be applied to interactions between components within the system to capture abnormal behavior, should a component misbehave.
- The practical method for creating abuse cases is usually through a process of brainstorming, involving security, reliability, and subject matter expertise
- Known attack patterns help developing abuse cases.

Trust Boundaries Mapping

- Defining zones of varying trust in an application helps identify vulnerable areas of communication and possible attack paths for security violations.
- For systems that have n-tier architecture or that rely on several third-party components, the potential for missing trust validation checks is high, so drawing trust boundaries becomes critical for such systems
- Combining trust zone mapping with data-flow analysis helps identify data that move from one trust zone to another and whether data checkpoints are sufficient to prevent trust elevation possibilities

Code Coverage Analysis

- Code coverage is a way of determining which code statements or paths have been exercised during testing. It is a test effectiveness measurement
- Help in identifying redundant test cases that do not increase coverage and also help in identifying redundant test cases that do not increase coverage
- There are various measures for coverage, such as path coverage, path testing, statement coverage, multiple condition coverage, and function coverage
- Covering all the code paths or statements does not guarantee that the software does not have faults; however, the missed code paths or statements should definitely be inspected.
- Unexercised code has serious bugs that can be leveraged into a successful attack

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Black Box Security Testing

Black Box Testing for Security

Black box tests can help

- identify implementation errors that were not discovered during code reviews, unit tests, or security white box tests
- discover potential security issues resulting from boundary conditions that were difficult to identify and understand during the design and implementation phases
- uncover security issues resulting from incorrect product builds (e.g., old or missing modules/files)
- detect security issues that arise as a result of interaction with underlying environment (e.g., improper configuration files, unhardened OS and applications)

Black Box Testing Tools

Black box test activities almost universally involve the use of tools to help testers identify potential security vulnerabilities

There are tools that focus on specific areas, including

- network security,
- database security,
- security subsystems, and
- web application security

Network Security Tools

- *Network security* based test tools focus on identifying vulnerabilities on externally accessible network-connected devices e.g. firewalls, servers, and routers

- Network security tools generally begin by using a port scanner to identify all active devices connected to the network, services operating on the hosts, and applications running on each identified service

- Some network scanning tools identify specific security vulnerabilities associated with the scanned host based on information contained within a vulnerability database.

- Tools are closely associated with penetration testing

Database Security Test Tools

Tools identify vulnerabilities in a systems database

- incorrect configuration of the database security parameters or
- improper implementation of the business logic used to access the database

Identify vulnerabilities that result in the disclosure or modification of sensitive data in the database

Security Subsystem Tools

- identify security vulnerabilities in specific subsystems
- used to test whether security-critical subsystems have been designed and implemented properly
  - correct operation of random number generators
  - cryptographic processors, and
  - other security-critical components.

Web Application Security Tool

- highlight security issues within applications accessed via the Internet
- these tools generally focus on identifying vulnerabilities and abnormal behavior within applications available over ports 80 (HTTP) and 443 (HTTPS)
  - These ports are allowed through a firewall to support web servers.
  - these tools may also test Web Services based application technologies over the same ports

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Fuzz Testing

Fuzzing

- The term fuzzing is derived from the fuzz utility (of Wisconsin), which is a random character generator for testing applications by injecting random data at their interfaces
- The idea is to look for interesting program behavior that results from noise injection and may indicate the presence of a vulnerability or other software fault.
  - completely random fuzzing is a comparatively ineffective way to uncover problems in an application.
- Fuzzing technology (along with the definition of fuzzing) has evolved to include more intelligent techniques. (Microsoft refers to this as “smart fuzzing”)
  - e.g., fuzzing tools are aware of commonly used Internet protocols, so that testers can selectively choose which parts of the data will be fuzzed.

Kinds of fuzzing

- **Black box**
  - The tool knows nothing about the program or even its inputs. Limited benefits
- **Grammar based**
  - The tool generates input based on known grammar
- **White box**
  - The tool generates new inputs based on the code of the program. Computationally complex

Network-based fuzzing

Act as one of the communicating parties

- Inputs could be produced
  - from scratch (e.g., from a protocol grammar)
  - replay of the previously recorded interaction
  - alterations of recorded interactions

Act as a “man in the middle”

- mutate messages exchanged between parties (using the knowledge of grammar)

SPIKE The fuzzer creation kit

- (http://resources.infosecinstitute.com/)

File format fuzzing

- Instead of just trying really long inputs, a more advanced way to fuzz is to try corner cases in some input format or malformed inputs just outside this input format
  - *CVE-2007-0243 Java JRE GIF Image Processing Buffer Overflow Vulnerability*
    - *Critical: Highly critical   Impact: System access   Where: From remote*
    - *… caused by an error when processing GIF images and can be exploited to cause a heap-based buffer overflow via a specially crafted GIF image with an image width of 0*
  - *Microsoft Security Bulletin MS04-028 Buffer Overrun in JPEG Processing (GDI+)*
    - *Could Allow Code Execution*
    - Impact: Remote Code Execution   Maximum: Critical
    - … cause by a zero sized comment field, without content.

Fuzzing Variations

1. Simple (original) fuzzing
   - try out ridiculously long inputs
   - try really long inputs for string arguments to trigger segmentation faults and hence find buffer overflows
     - e.g. register with Facebook with a 1Mbyte long username
2. Protocol/format/language fuzzing
   - try out strange inputs, given some format/language
3. State-based fuzzing
   - try out strange sequences of input

2 & 3 are essentially forms of model-based testing

State-based Protocol Fuzzing

- Instead of fuzzing the content of individual messages, we can also fuzz the order of messages.
- This is interesting for protocols that have different types of messages, which are expected to come in a particular order:
- This can reveal flaws in the application logic (more specifically, flaws in the implementation of the protocol state machine)
- Essentially this is a from of model-based testing, where we automatically test if an implementation conforms to model (in the form of a finite state machine aka finite automaton), by random test sequences

Dealing with crashes in Fuzz Testing

- One of the most interesting outputs of fuzz testing come from analysis of crashes
  - What is the **root cause** (so it can be fixed)**?**
  - Is there a way to **make the input smaller**, so it is more understandable?
  - Are **two or more crashes signaling the same bug?**
  - Does the crash signal an **exploitable vulnerability**?

Finding errors before crash

- **Compile** the program with **Address Sanitizer** (ASAN)
  - https://github.com/google/sanitizers/wiki/AddressSanitizer
- ASAN instruments accesses to arrays to check for overflows, and use-after-free errors. (Alarm to be raised in case of deviation/violation)
  - Carry out Fuzz testing
  - Did the execution result in ASAN-signaled error?
  - If so, check for exploitability
- Similarly, it is possible to *compile with other sorts of error checkers* for the purposes of testing

CERT Basic Fuzzing Framework (BFF)

- The CERT Basic Fuzzing Framework (BFF) is a software testing tool that finds defects in applications that run on the Linux and Mac OS X platforms
  - uses mutational (taking well-formed input data and corrupting it in various ways) fuzzing on software that consumes file input.
  - automatically collects test cases that cause software to crash in unique ways, and associated debugging information
  - helps efficiently discover and analyze security vulnerabilities found via fuzzing
  - Uses machine learning techniques to minimize the manual effort for the fuzzing.
- CERT used the BFF to find a number of critical vulnerabilities in products such as Adobe Reader and Flash Player; Foxit Reader; Apple QuickTime, Preview, and Mac OS X; Xpdf; Poppler; etc.

Fuzzing – Pros & Cons

Advantages	Issues
• Simple to design and perform automated tests • Find bugs or crashes not easily visible via other testing techniques • Bugs found are sometimes severe and include defects that could be exploitable in the wild, including unhandled exceptions, crashes, memory faults/leaks and so on • Usually very inexpensive to implement	• Generally finds very simple faults • Often takes an extremely long time to run • Crashes can often be difficult to analyze, especially when using black-box fuzzing • Mutation templates for applications with complex inputs can often be time-consuming to produce

http://resources.infosecinstitute.com/

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Penetration Testing

Background of Penetration Testing

- By the 1970s, the US government was regularly using teams to assess the security of computer systems by trying to penetrate them. These teams were referred to as red teams, or tiger teams.
- The penetration testing had been largely focused on the environments:
  - Attempt to compromise the security of the Computer operating systems, along with their access control mechanisms by penetration testing.
  - In a networked computer context, operating system configurations, including their respective network services, are usual targets for penetration tests. These operating system components have offered countless opportunities for penetration testing over the years
- Recently push was for applying penetration testing techniques “up” the software abstraction levels, beyond the operating system and network services, towards the application software itself

Security is not compositional

- According to Leslie Lamport, a Turing Award winner, Security is not compositional.
- Two components that are secure on their own are not necessarily secure when used in combination
  - A change to one component might not break that component, but could break the whole system due to the lack of compositionality
- Upon change to the software, the configuration, the network topology, and so on, potentially new vulnerabilities are created

An art or a science?

- Pen testers must be creative. They think about how a system is put together, and where assumptions made by designers represent weaknesses
- Pen testers will cleverly adapt the weaknesses to gain a foothold in one place. They may use that foothold to exploit a weakness somewhere else.
- Systems could be incorrectly built or misconfigured in the some ways.
- Tools are built to systematically look for weakness patterns, and exploit them.

Thus Pen testing is both an art and a science.

Penetration Testing Skills

- A pen tester needs to know a lot about the target domain.
- For example, if the pen tester is attacking web applications, then the pen tester needs to know how the web works. Need to know how systems are built in that domain.
  - What protocols allow applications to communicate. For the web, that's HTTP and TCP, and IP.
  - The languages that are used to build applications like PHP, Java, or Ruby for talking about the web.
  - Frameworks that used to build applications or application components like, for the web, Ruby on Rails, DreamWeaver, Drupal, and so on.
- Pen tester also need to know common weaknesses from that domain.
  - For example, the bugs that are common to web applications like SQL injections or cross-site scripting, or cross-site request forgery. Or common misconfigurations or bad designs, like the use of default passwords or hidden files.

Categories of Penetration Testing Tools

- **Host-Based Tools**
- **Network-Based Tools**
- **Application Testing Proxies**
- **Application Scanning Tools**
- Tools integrated with other IT security technologies, viz. firewalls, intrusion detection and prevention systems

Host-Based Tools

- Host-based testing tools test the local operating system to assess its technical strengths and weaknesses, e.g. Dan Farmer’s COPS program, which evaluated the security posture of a UNIX computer
    - e.g., evaluate file access control mechanisms for opportunities for attackers to affect the security of the host
    - examine every file, configuration data (including registry keys on Windows systems), installed patch inventory, and so on from the perspective of every ID on the system
    - Check common operating system configuration mistakes and omissions such as dangerous setuid files, unnecessary network services enabled, and excessive privileges for user accounts

Network-Based Tool

- Network-based testing tools assess the security configuration of a computer operating system from afar—across a network, e.g. Chris Klaus's Internet Security Scanner (ISS) program
- Examine a target computer(s) for weaknesses that may be exploitable from a remote networked location using a database of vulnerabilities
  - Advantage- Scale: A huge number of computers can be evaluated across a network;
  - Limitation- Coverage: Network-based testing can only evaluate the externally accessible interfaces

Nmap for network probing

Nmap stands for "network mapper". Free, open source (commercial versions too) http://nmap.org/

Figures out

- what **hosts** are available on the network,
- what **services** (application name and version) those hosts are offering,
- what **operating systems** (and OS versions) they are running,
- what type of **packet filters/firewalls** are in use
- *… etc.*

Works by **sending raw IP packets** into the network and **observing the effects**

- Standard "ping" protocol, Looks for HTTPS(port 443) or HTTP(port 80) servers, - Probes to other TCP ports

Probes that elicit different responses on different OSes ("fingerprinting")

Can be stealthy!

- Control the rate of scanning to "work under the radar"

Application Testing Proxies

Application Testing Proxies

- enable the security tester to look behind the graphical user interface when testing a web application or web service
- Requests to and responses from the server are intercepted, observed, and optionally manipulated

Web applications are common pen testing targets

- Web proxies sit *between* the browser and server
- Displaying exchanged packets
- Modifying them as directed by the tester

Application Scanning Tools

- These tools do penetration testing scans of general purpose web-based software applications
- connect to web applications and attempt a series of well defined tests for each data field, cookie, etc.
- Such tools initiate a “learning mode” in which they observe the normal operation of a web application. Based on learning, they attempt to exploit common web application defects such as data overruns, SQL injection, and cross-site scripting (XSS)

https://insights.sei.cmu.edu/blog/10-types-of-application-security-testing-tools-when-and-how-to-use-them/

https://insights.sei.cmu.edu/blog/decision-making-factors-for-selecting-application-security-testing-tools/

Ethical Hacking

- Penetration testing tools are meant to reveal security vulnerabilities so that they can be fixed, not so that they can be exploited for the purposes of crime or harm.
- But it is true that people will use these penetration testing tools for nefarious purposes.
- In that way, they are sort of two way tools.
  - Just as guns can be used to defend, guns can be used to attack. Ethical hacking is not to be someone who uses pen testing tools to attack.

Hacker Hat Types

Black hat hackers

- are criminals who break into computer networks with malicious intent.
  - They may also release malware that destroys files, holds computers hostage, or steals passwords, credit card numbers, and other personal information
  - Hacking can operate like big business,boast partners, resellers, vendors, and associates, and they buy and sell licenses for malware to other criminal organizations for use in new regions or markets.
  - Some countries are lax with them unless they harm their own citizens

White hat hackers

- sometimes also called "ethical hackers" or "good hackers", exploit computer systems or networks to identify their security flaws so they can make recommendations for improvement.
  - White hat hackers use the same hacking methods as black hats, but with the permission of the system owner first.
  - work with network operators to help fix the issue before others discover it.
  - Never cross the law.

Gray hat hackers

- a blend of both black hat and white hat.
  - Often look for vulnerabilities in a system without the owner's permission.
  - If issues are found, they report them to the owner, sometimes requesting a small fee to fix the problem.

https://www.kaspersky.com/resource-center/definitions/hacker-hat-types

Software Security Engineering, Julia H. Allen, et al, Pearson, 2008.
Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2018.
http://resources.infosecinstitute.com
www.owasp.com
www.microsoft.com

Thank You!