SANS OFFENSIVE OPERATIONS

Overall Product Security Assessment Process

THINK RED / ACT BLUE
POSTER

sans.org/offensive-operations

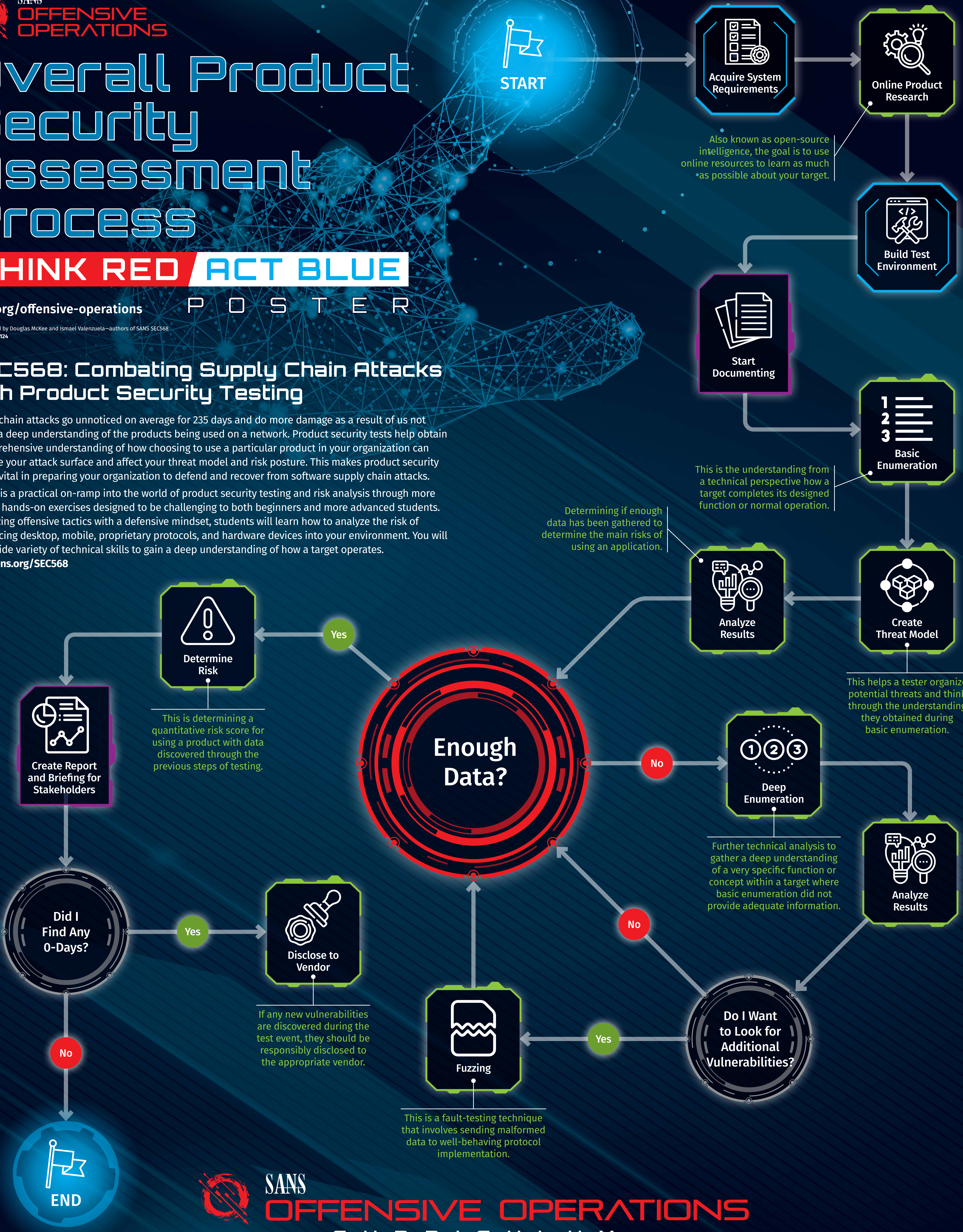
Poster created by Douglas McKee and Ismael Valenzuela—authors of SANS SEC568
OO_SEC568_0124

SEC568: Combating Supply Chain Attacks with Product Security Testing

Supply chain attacks go unnoticed on average for 235 days and do more damage as a result of us not having a deep understanding of the products being used on a network. Product security tests help obtain a comprehensive understanding of how choosing to use a particular product in your organization can increase your attack surface and affect your threat model and risk posture. This makes product security testing vital in preparing your organization to defend and recover from software supply chain attacks.

SEC568 is a practical on-ramp into the world of product security testing and risk analysis through more than 20 hands-on exercises designed to be challenging to both beginners and more advanced students. By utilizing offensive tactics with a defensive mindset, students will learn how to analyze the risk of introducing desktop, mobile, proprietary protocols, and hardware devices into your environment. You will use a wide variety of technical skills to gain a deep understanding of how a target operates.

www.sans.org/SEC568



Also known as open-source intelligence, the goal is to use online resources to learn as much as possible about your target.

This is the understanding from a technical perspective how a target completes its designed function or normal operation.

This helps a tester organize potential threats and think through the understanding they obtained during basic enumeration.

Determining if enough data has been gathered to determine the main risks of using an application.

Further technical analysis to gather a deep understanding of a very specific function or concept within a target where basic enumeration did not provide adequate information.

This is a fault-testing technique that involves sending malformed data to well-behaving protocol implementation.

This is determining a quantitative risk score for using a product with data discovered through the previous steps of testing.

If any new vulnerabilities are discovered during the test event, they should be responsibly disclosed to the appropriate vendor.

SANS OFFENSIVE OPERATIONS CURRICULUM

FOUNDATIONAL

SEC504
Hacker Tools, Techniques, and Incident Handling
GCIH

PENETRATION TESTING: COMPREHENSIVE

SEC560
Enterprise Penetration Testing
GPEN

SEC660
Advanced Penetration Testing, Exploit Writing, and Ethical Hacking
GXPN

PENETRATION TESTING: WEB & CLOUD

SEC542
Web App Penetration Testing and Ethical Hacking
GWAPT

SEC588
Cloud Penetration Testing
GCPN

PENETRATION TESTING: SPECIALIZED

SEC554
Blockchain and Smart Contract Security

SEC575
iOS and Android Application Security Analysis and Penetration Testing
GMOB

SEC568
Combating Supply Chain Attacks with Product Security Testing

SEC617
Wireless Penetration Testing and Ethical Hacking
GAWN

EXPLOIT DEVELOPMENT

SEC760
Advanced Exploit Development for Penetration Testers

PURPLE TEAMING

SEC598
Security Automation for Offense, Defense, and Cloud

SEC599
Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses
GDAT

SEC699
Advanced Purple Teaming – Adversary Emulation & Detection Engineering

RED TEAMING

SEC565
Red Team Operations and Adversary Emulation
GRTP

SEC670
Red Teaming Tools – Developing Windows Implants, Shellcode, Command and Control

Online Product Research

ONLINE PRODUCT RESEARCH

- Vulnerability/CVE databases—Provide intel to know threats to an application
  - **nvd.nist.gov/search**
  - **cve.mitre.org**
  - **cvedetails.com**
  - **vuldb.com**
  - **exploit-db.com**
- Technical Product Documentation—Non-marketing materials such as user, admin and service manuals
- Release notes—Documents released with a new launch of an updated version of a product
  - **productplan.com/glossary/release-notes**
- Github—Many applications have components or additional research on GitHub
  - **github.com/search/advanced**

THREAT MODELING

- Ask: What are we working on?—Having a technical understanding of how an application accomplishes its goal
  - » Document using DFDs
    - **github.com/adamshostack/DFD3**
- Ask: What can go wrong?—Consider what will happen in the absence of mitigations or security controls.
- Create Attack Trees—Attack Trees is a structured approach to analyzing exploitation paths which take the form of conceptual diagrams that show the variety of ways in which something can go wrong, and the reason why they might go wrong.
  - **ncsc.gov.uk/collection/risk-management/using-attack-trees-to-understand-cyber-security-risk**
  - Example: **mitre.org/sites/default/files/2021-11/Playbook-for-Threat-Modeling-Medical-Devices.pdf**

Create Threat Model

Basic Enumeration

BASIC ENUMERATION (BE)

Tools

- Sysinternals
  - » Used in BE to quickly discover running processes, permissions, networking connections, filesystem activity and much more
    - **learn.microsoft.com/en-us/sysinternals**
- Process Hacker
  - » A similar but different version of Process Explorer from Sysinternals. Can provide slightly different functionality such as more robust memory searching.
    - **processhacker.sourceforge.io**
- ProcDot
  - » Tool to help visualize application activity obtained from Systinernal's Procmon
    - **procdot.com**
- Microsoft ASA
  - » Used in BE to create a base line of before and after an application is installed.
    - **github.com/microsoft/AttackSurfaceAnalyzer**
- Regshot
  - » Used to baseline the registry during BE
    - **sourceforge.net/projects/regshot**
- Netstat
  - » Used in BE to check for what ports are open for what PID
    - **linux.die.net/man/8/netstat**
- Wireshark
  - » Used in BE to understand if traffic is encrypted and a known protocol
    - **wireshark.org**
- ADB
  - » Used in BE to connect to an Android device or emulator. Allows to obtain an interactive terminal, upload/download files, capture logs and much more
    - **developer.android.com/tools/adb**
- Logcat
  - » Command line tool up access the log buffers on Android. Used in BE to help understand how an application functions, determine versions of dependencies, look at traffic pre-encryption and more.
    - **developer.android.com/tools/logcat**
- APKLab
  - » Used in BE to automate the unpacking and top level analysis of APKs
    - **github.com/APKLab/APKLab**
- APKLeaks
  - » Used in BE to look for sensitive information stored in APK files.
    - **github.com/dwisiswant0/apkleaks**
- Android Profiler
  - » Can be used in BE to look in memory for sensitive strings
    - **developer.android.com/studio/profile/memory-profiler**
- TCPDump
  - » Used in BE to understand if traffic is encrypted and a known protocol when Wireshark is not practical.
    - **tcpdump.org**
- Burp
  - » Used in BE to inspect web requests and preform high level security scans
    - **portswigger.net/burp**

Terminology

- Installation footprint—Changes to a system due to an installation
  - » Registry Keys—Contain configuration and startup information
    - **learn.microsoft.com/en-us/troubleshoot/windows-server/performance/windows-registry-advanced-users**
  - » Open ports/sockets—Input/output point for an application
    - **ipcisco.com/lesson/network-ports**
  - » Files added—Understand any additions to the filesystem as these could be part of the supply chain
  - » Init scripts—Boot up scripts explain what is running on a static system
    - **oreilly.com/library/view/essential-system-administration/0596003439/ch04s02.html**
- Sensitive information—Anything data that if stolen would weaken the security profile
  - » Usernames/passwords—Protect access to an application or device
  - » Crypto/API keys—Can be similar to a username/password as a token
    - **fortinet.com/resources/cyberglossary/api-key**
  - » PII/PHI—Information about the human element that is private and requires protection
  - » Stored secrets—Anything vital to the operation if stolen would compromise the security posture
    - **cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html**
- Custom Components—Anything created by the vendor where the code is not public but proprietary
- Third-party components—Components integrated into an application or device that are not original to the selling vendor
  - » Linked libraries—These are often DLLs on Windows or .so files on Linux
    - **learn.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library**
    - **tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html**
  - » APIs—Mechanisms that enable two software components to communicate with each other using a set of definitions and protocols
    - **aws.amazon.com/what-is/api**
  - » Sub-components—Helper applications or smaller applications that help make up the larger system, such as a chat application
  - » Network traffic—Standardized traffic such as HTTP vs proprietary protocols
  - » User Input—Any data that is provided externally to the system and is then processed by the system
  - » Forms—An interface where a user enters data
    - **comidor.com/help-center/application-builder/user-fields-user-forms**
  - » Network input—Data which comes from network that is parsed by the application
    - **cloudflare.com/learning/network-layer/what-is-a-packet**
  - » Config files—Configuration files which an application parses during runtime
    - **opensource.com/article/21/6/what-config-files**
  - » File/Image parsing—User supplied images or binary files that are uploaded during runtime that is then parsed by the application
    - **klippa.com/en/blog/information/file-parsing**

Analyze Results

NETWORK ANALYSIS

Tools

- Wireshark
  - » Used in network analysis to inspect network traffic
    - **wireshark.org**
- TCPDump
  - » Used in network analysis to inspect network traffic when Wireshark is not practical.
    - **tcpdump.org**
- HTTP ToolKit
  - » Used in network analysis to inspect network traffic particularly useful for dealing with mobile applications
    - **httptoolkit.com**
- Mitmproxy
  - » Used to decrypt traffic for deeper examination during network analysis
    - **mitmproxy.org**
- SSLStrip
  - » Used to decrypt traffic for deeper examination during network analysis
    - **kali.org/tools/sslstrip**
- Frida
  - » Can be used to help decrypt traffic or examine traffic in memory especially useful for mobile or embedded systems
    - **frida.re**
- APKLab
  - » Used in network analysis to automate the removal of cert pinning
    - **github.com/APKLab/APKLab**
- Scapy
  - » A python library used to manipulate and document networking packets for network analysis
    - **scapy.readthedocs.io/en/latest/index.html**
- Binwalk
  - » For network analysis can be used to determine format and unpack data within packet payloads
    - **github.com/ReFirmLabs/binwalk**
- File
  - » Linux command which can identify files in networking payloads
    - **geeksforgeeks.org/file-command-in-linux-with-examples**
- Entropy Tests
  - » Mathematical test to determine if a group of bytes is encrypted, compressed or plaintext
    - **citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.9861&rep=rep1&type=pdf**
- Readelf
  - » Linux command which can identify if binary data is in the elf file format
    - **man7.org/linux/man-pages/man1/readelf.1.html**
- Strings
  - » Linux command which can identify strings within binary data
    - **linux.die.net/man/1/strings**

Terminology

- » Simplify Scope—Reducing the number of packets that need to be analyzed to a fair representation
- » Embedded networking—When the payload of a packet contains redundant networking information such as IP addresses.
- » Enumerate Patterns—Comparing bytes across multiple packets and network flows to determine the purpose of those bytes
- » Clear text clues—Unencrypted ASCII data present in a packet which can be used to identify the purpose of a byte or bytes
- » Examine Externally—Extracting payload data and using tools not traditionally used to analysis networking data to assist with analysis

Determine Risk

DETERMINE RISK

- DREAD—Damage, Reproducibility, Exploitability, Affected users, Discoverability
  - » DREAD is a framework for scoring threats and prioritizing which ones may take precedence for remediation.
    - **learn.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers#the-dread-approach-to-threat-assessment**
    - **microsoftpressstore.com/store/writing-secure-code-9780735617223**
    - **adam.shostack.org/modsec08/Shostack-ModSec08-Experiences-Threat-Modeling-At-Microsoft.pdf**
  - » DREAD is effective when combined with a scoring rubric which is influenced by technical analysis.

DEEP ENUMERATION

- Attacker's Input—Any data that is provided externally to the system and is then processed by the system that can be controlled by an attacker
  - » Forms—An interface where a user enters data
    - **comidor.com/help-center/application-builder/user-fields-user-forms**
  - » Network input—Data which comes from network that is parsed by the application
    - **cloudflare.com/learning/network-layer/what-is-a-packet**
  - » Config files—Configuration files which an application parses during runtime
    - **opensource.com/article/21/6/what-config-files**
  - » File/Image parsing—User supplied images or binary files that are uploaded during runtime that is then parsed by the application
    - **klippa.com/en/blog/information/file-parsing**
    - **adam.shostack.org/modsec08/Shostack-ModSec08-Experiences-Threat-Modeling-At-Microsoft.pdf**

Deep Enumeration

Analyze Results

BINARY CODE ANALYSIS

Tools

- DotPeak
  - » Tool used to decompile .NET
    - **jetbrains.com/decompiler**
- dnSpy
  - » Tool used to decompile .NET which includes a debugger
    - **github.com/dnSpyEx/dnSpy**
- Asar
  - » NPM package to unpack electron applications
    - **npmjs.com/package/asar**
- Uncompyle
  - » Tool used to decompile python binaries
    - **pypi.org/project/uncompyle6**
- Ghidra
  - » C/C++ disassembler and decompiler
    - **ghidra-sre.org**
- IDA
  - » C/C++ disassembler and decompiler, free and paid version
    - **hex-rays.com/ida-pro**
- Jadx
  - » Java decompiler, primarily designed for Android APKs
    - **github.com/skylot/jadx**
- Bytecode Viewer
  - » Java Reversing engineering suite which integrates 6 different decompilers and malicious code scanning
    - **bytecodeviewer.com**

Fuzzing

FUZZING

Tools

- AFL++
  - » Industry standard coverage-guided fuzzer for open and closed source fuzzing
    - **github.com/AFLplusplus/AFLplusplus**
- WinAFL
  - » A fork of AFL for use on Windows targets
    - **github.com/googleprojectzero/winafl**
- Aflnet
  - » A greybox fuzzer based on AFL for networking protocol implementations.
    - **github.com/aflnet/aflnet**
- Libafl
  - » AFL fuzzer written in rust as a library
    - **github.com/AFLplusplus/LibAFL**
- Honggfuzz
  - » A security oriented, feedback-drive easy-to-use fuzzer
    - **github.com/google/honggfuzz**
- Jackalope
  - » Jackalope is a customizable, distributed, coverage-guided fuzzer that is able to work with black-box binaries.
    - **github.com/googleprojectzero/Jackalope**
- LibFuzzer
  - » LibFuzzer is an in-process, coverage-guided, evolutionary fuzzing engine.
    - **llvm.org/docs/LibFuzzer.html**
- BooFuzz
  - » Network fuzzer built on the older Sulley
    - **boofuzz.readthedocs.io/en/stable**
- Scapy
  - » Scapy has a built in fuzz function which can be use used to fuzz network layers
    - **scapy.readthedocs.io/en/latest/usage.html#fuzzing**
- WTF
  - » What the fuzz or wtf is a distributed, code-coverage guided, snapshot-based fuzzer designed for targets running on Microsoft Windows.
    - **github.com/0vercl0k/wtf**

Terminology

- Identify target code path—Choosing a single specific code function within the application to fuzz
- Determine input type—Choosing the type of data which will be fuzzed i.e. image, network packets, pdf, json etc.
- Create Test Harness—A helper application developed to bridge the gap between how the fuzzer expects input to occur and how input actually happens in the application
  - **bishopfox.com/blog/fuzzing-aka-fuzz-testing**
- Build Input Corpus—A set of interesting test cases which exercise as many code branches as possible under test. A corpus is shared across fuzzer runs and grows over time.
  - **chromium.googlesource.com/chromium/src/+/main/testing/libfuzzer/efficient_fuzzing.md**
  - **github.com/strongcourage/fuzzing-corpus**
- Code Coverage analysis—The percentage of code covered by your fuzz target. The higher the percentage the better the fuzzer.
  - **chromium.googlesource.com/chromium/src/+/main/testing/libfuzzer/efficient_fuzzing.md#Code-coverage**
- Triage Crashes—Examining each crash discovered by a fuzzer to determine whether the crash might be worth investigating further or is a valid finding.
  - **trustfoundry.net/2020/01/22/introduction-to-triaging-fuzzer-generated-crashes**