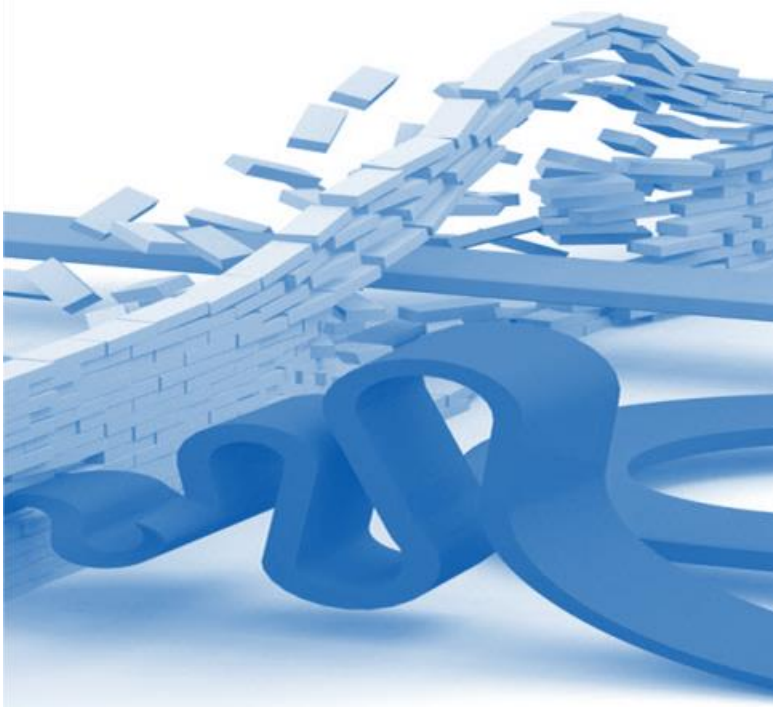


PENETRATION TESTING (PART III)

Prof. Dr. Bernhard Tellenbach

Content

Phases borrowed from PTES:

- **Pre-engagement interactions:** Initial communication and reasoning behind a penetration test
- **Intelligence gathering and threat modeling:** Get a better understanding of the tested organization
- **Vulnerability research, exploitation:** Identify vulnerabilities and demonstrate proof-of-concept or “real” exploits
- **Post exploitation:** Determine the value of the compromised target, maintain control and gain further access to other resources
- **Reporting:** Captures the entire process in a manner that makes sense to the customer and provides the most value to it

Goals

- You can explain typical activities of the threat modeling, vulnerability analysis, exploitation and post exploitations
- You can discuss the main challenges in these phases and what methods or (types of) tools could be used
- You can discuss several reasons why vulnerability scanners might not work perfectly (false positives / false negatives)
- You know the architecture and main features of the Metasploit framework and you can use it for vulnerability testing and exploitation tasks

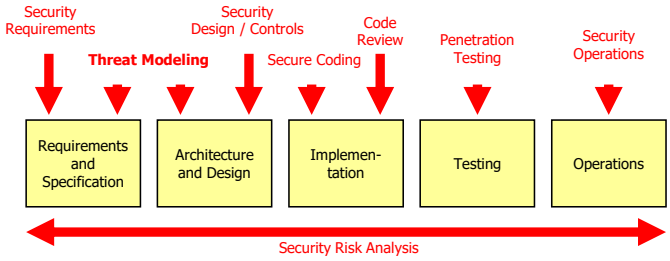
Threat Modeling

Phases borrowed from PTES:

- **Pre-engagement interactions:** Initial communication and reasoning behind a penetration test
- **Intelligence gathering and threat modeling:** Get a better understanding of the tested organization
- **Vulnerability research, exploitation:** Identify vulnerabilities and demonstrate proof-of-concept or “real” exploits
- **Post exploitation:** Determine the value of the compromised target, maintain control and gain further access to other resources
- **Reporting:** Captures the entire process in a manner that makes sense to the customer and provides the most value to it

Threat Modeling – Attacker-Centric

- Threat modelling when building software or systems (see SWS1):
  - Identify possible threats
  - Based on these threats, identify vulnerabilities in the system design
    - Considering already defined security requirements or security controls
  - The identified vulnerabilities provide the basis for additional security requirements
- Methods to identify/document threats
  - STRIDE
  - Attack Trees



Threat Modeling – Defender-Centric

- Attackers use offensive threat modeling – what is the most promising way to get from the attack surface to the targeted asset(s)?
- Attack surface: "… is the set of ways in which an adversary can enter the system and potentially cause damage" [1]
  - Constructed from data collected in the intelligence gathering phase
- Attacker use the system's entry and exit points (i.e. its interfaces) to attack/interact with it
  - Exit points are required to provide feedback to the attacker
- Relevant entry an exit points depend on the attacker model and the scope (e.g., OSSTMM channels) of the analysis
  - There are systematic definitions of the attack surface*

[1] Pratyusa K. Manadhata and Jeannette M. Wing "An Attack Surface Metric" in IEEE Transactions on Software Engineering, 2010

* "The **rav** is a scale measurement of an attack surface, the amount of uncontrolled interactions with a target, which is calculated by the quantitative balance between porosity, limitations, and controls. In this scale, 100 rav (also sometimes shown as 100% rav) is perfect balance and anything less is too few controls and therefore a greater attack surface. More than 100 rav shows more controls than are necessary which itself may be a problem as controls often add interactions within a scope as well as complexity and maintenance issues."

Source: OSSTMM 3 – The Open Source Security Testing Methodology Manual

Please remember that all such calculations should be taken with more than a grain of salt. As we have already said in earlier lectures, there is often little hard evidence that such formulas measure something interesting, or that the measurement correlates with something interesting. Always ask yourself whether this measure makes sense for your use case and your organization.

Threat Modeling – Defender-Centric

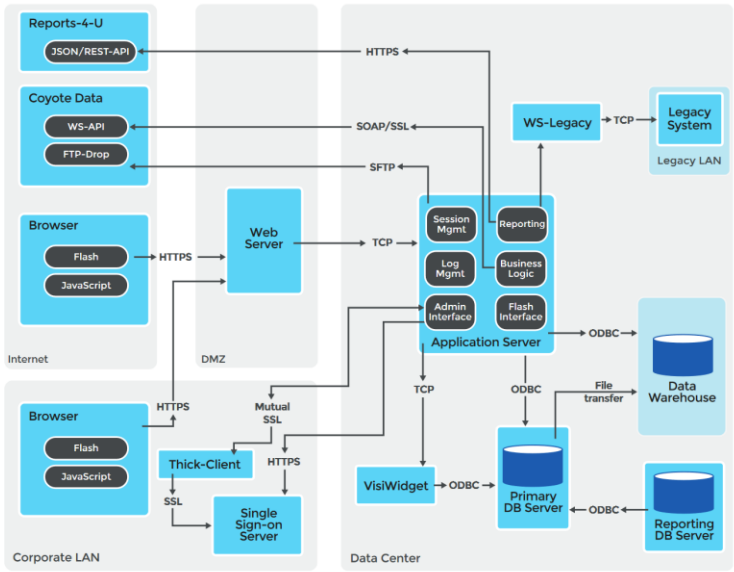
- Penetration testers can use it as part of their service to more accurately simulate an attack from the perspective of a specific threat actor
  1. Understand the organization's defenders and where they have fortified against attacks
  2. Create offensive architecturally-based process flow diagrams that provide a mapping from the identified attack surface to the targeted assets along multiple paths
  3. Mapping of defensive capabilities as potential security check-points and mitigations to be avoided => figure out potential exploits against both defenders and defenses
  4. Identify and prioritize useful targets for the attack and potential attack vectors

A **process flow diagram** (**PFD**) is a diagram commonly used in chemical and process engineering to indicate the general flow of plant processes and equipment. The PFD displays the relationship between *major* equipment of a plant facility and does not show minor details such as piping details and designations. Another commonly used term for a PFD is a *flowsheet*.

Source: https://en.wikipedia.org/wiki/Process_flow_diagram

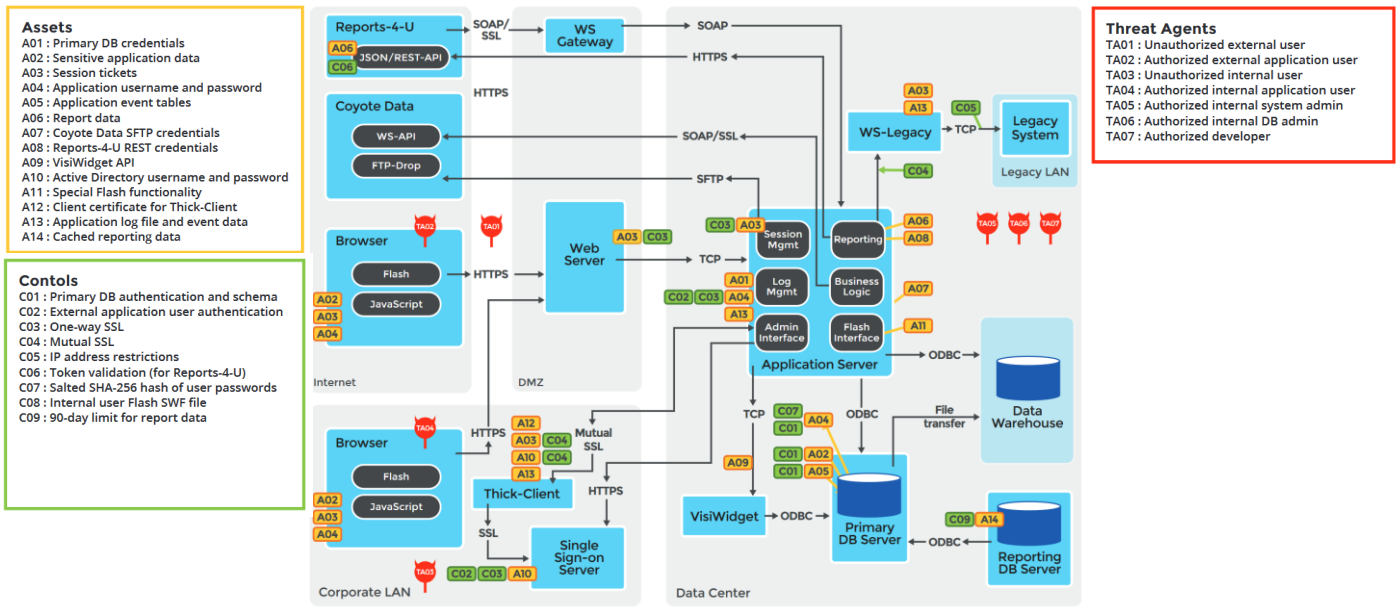
Process Flow Diagram – Example (1)

Components identified in this diagram include the application server, a legacy system, data warehouse, thick client, primary database server, and a single sign-on server.



Source: https://www.synopsys.com/content/dam/synopsys/sig-assets/ebooks/threat-modeling-misconceptions.pdf

Process Flow Diagram – Example (2)



Source: https://www.synopsys.com/content/dam/synopsys/sig-assets/ebooks/threat-modeling-misconceptions.pdf

Primary and Secondary Assets

- Identify and categorize primary and secondary assets
  - Primary asset: Part of the system/functionality under test (in scope)
  - Secondary asset: Not in scope but shared with or linked to the assets in scope
- Primary asset compromise can imply the compromise of a secondary asset
- Secondary assets impact what threat agents one should consider
  - Map threat agents against primary and secondary assets
  - What are the tools, resources and capabilities of the relevant threat agents?
    - Find relevant news of comparable organizations being compromised
    - Anecdotal evidence for the threat model and baseline for the organization

Note that the terms primary and secondary assets are proposed and used in this way by PTES.

Example: Impact of Secondary Assets on Relevant Threat Agents

- Penetration test of a customer relationship management (CRM) application
- Primary assets:
  - Database system for storing the customer information
  - Webserver hosting the CRM frontend
  - …
- Secondary asset:
  - Employee database from human resources is located on the same database server
- Threat agents
  - With primary assets only, insiders won't be relevant (employees have CRM access anyway)
  - Considering secondary assets, insiders become relevant (access to salaries, health data, …)
  - CRM application is a steppingstone to obtain employee information

Threat Modeling – Attack Patterns

- Understanding adversary behavior is important, especially for red-teaming
  - MITRE (/ˈmaɪtə(r)/) CAPEC and ATT&CK organize knowledge about adversary behavior
- MITRE Common Attack Pattern Enumeration and Classification (CAPEC)
  - Focus on application security
  - Enumerates exploits against vulnerable systems
  - Includes social engineering / supply chain
  - Associated with Common Weakness Enumeration (CWE)
- MITRE Adversarial Tactics, Techniques & Common Knowledge (ATT&CK)
  - Focus on network defense
  - Based on threat intelligence and red team research
  - Provides contextual understanding of malicious behavior
  - Supports testing and analysis of defense options

MITRE ATT&CK® and CAPEC are currently probably the most prominent examples of such attack patterns. Many security solutions make use of them when it comes to classifying, referencing or documenting threat behaviors. However, note that there are many other sources describing attack patterns at various levels of detail. For example, the book “Exploiting Software” by Gary McGraw and Greg Hoglund detail 48 attack patterns. This book was one of the first attempts to provide a collection of attacks along with a very detailed description of the attack vector.

MITRE is a **non-profit organization** focused on cybersecurity and solving security challenges to create a safer IT environment for organizations. MITRE developed the **ATT&CK framework** to classify adversarial tactics universally. ATT&CK is also a database that organizations can use to reference and document threat behaviors across the entire attack lifecycle.

MITRE writes that “MITRE ATT&CK® is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.”

Source: https://attack.mitre.org/

CAPEC - The Common Attack Pattern Enumeration and Classification (CAPEC™) effort provides a publicly available catalog of common attack patterns that helps users understand how adversaries exploit weaknesses in applications and other cyber-enabled capabilities. "Attack Patterns" are descriptions of the common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. Attack patterns define the challenges that an adversary may face and how they go about solving it. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples.

Each attack pattern captures knowledge about how specific parts of an attack are designed and executed, and gives guidance on ways to mitigate the attack's effectiveness. Attack patterns help those developing applications or administrating cyber-enabled capabilities to better understand the specific elements of an attack and how to stop them from succeeding.

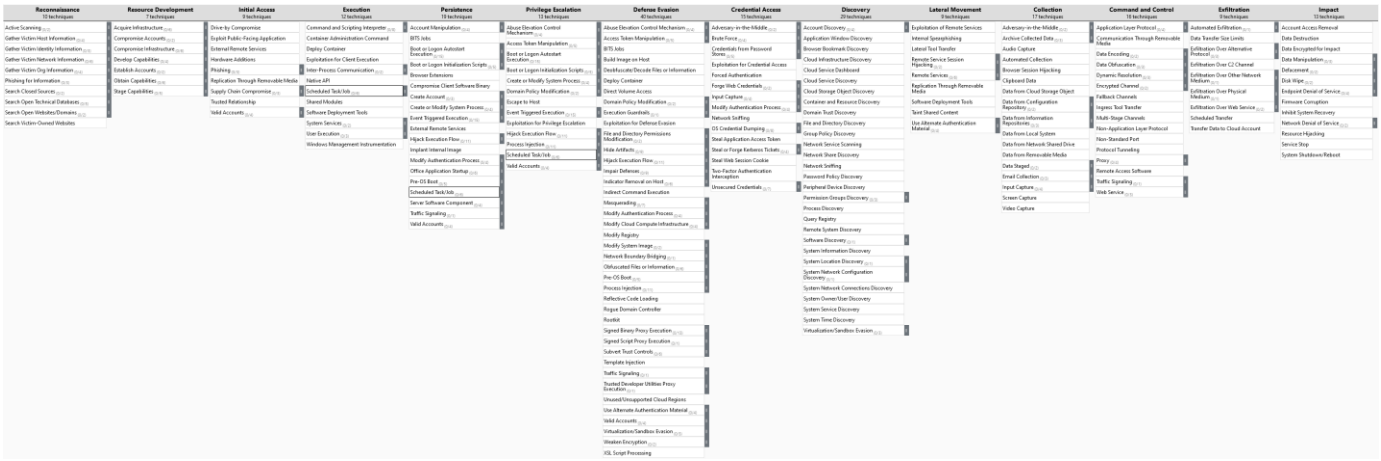
Source: http://capec.mitre.org

Threat Modeling – ATT&CK vs. CAPEC

- Many attack patterns enumerated by CAPEC are employed by adversaries through specific techniques described by ATT&CK
  - Enables contextual understanding of the patterns within an adversary's operational lifecycle.
  - CAPEC attack patterns and related ATT&CK techniques are cross referenced when appropriate

- Use CAPEC for:
  - Application threat modeling
  - Developer training and education
  - Penetration testing
- Use ATT&CK for:
  - Comparing computer network defense capabilities
  - Defending against the APT
  - Hunting for new threats
  - Enhancing threat intelligence
  - Adversary emulation exercises

MITR ATT&CK – Enterprise Matrix



https://attack.mitre.org/matrices/enterprise/

CAPEC: Attacks - Mechanisms / Domains / Groupings

CAPEC-553: Mobile Device Patterns

View ID: 553
Structure: Implicit

Objective

This view (slice) covers standard attack patterns that target direct expl

Filter

/Attack_Pattern_Catalog/*/*[@ID = (187, 498, 604, 605, 606, 608, 60

Membership

Nature	Type	ID	Name
HasMember		187	Malicious Automated Software Update
HasMember		498	Probe iOS Screenshots
HasMember		604	Wi-Fi Jamming
HasMember		605	Cellular Jamming
HasMember		606	Weakening of Cellular Encryption
HasMember		608	Cryptanalysis of Cellular Encryption
HasMember		609	Cellular Traffic Intercept
HasMember		610	Cellular Data Injection
HasMember		612	WiFi MAC Address Tracking
HasMember		613	WiFi SSID Tracking
HasMember		614	Rooting SIM Cards
HasMember		615	Evil Twin Wi-Fi Attack
HasMember		617	Cellular Rogue Base Station
HasMember		618	Cellular Broadcast Message Request
HasMember		619	Signal Strength Tracking
HasMember		621	Analysis of Packet Timing and Sizes
HasMember		622	Electromagnetic Side-Channel Attack
HasMember		623	Compromising Emanations Attack
HasMember		625	Mobile Device Fault Injection
HasMember		626	Smudge Attack
HasMember	S	627	Counterfeit GPS Signals
HasMember		628	Carry-Off GPS Attack
HasMember		629	Unauthorized Use of Device Resources

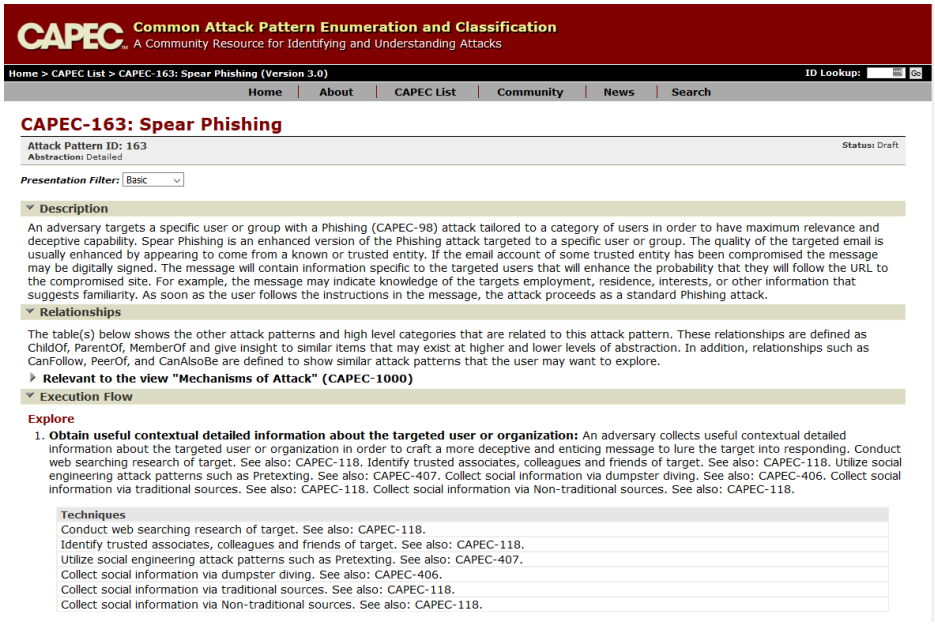
1000 - Mechanisms of Attack

- Engage in Deceptive Interactions - *(156)*
- Abuse Existing Functionality - *(210)*
- Manipulate Data Structures - *(255)*
- Manipulate System Resources - *(262)*
- Inject Unexpected Items - *(152)*
- Employ Probabilistic Techniques - *(223)*
- Manipulate Timing and State - *(172)*
- Collect and Analyze Information - *(118)*
- Subvert Access Control - *(225)*

3000 - Domains of Attack

- Software - *(513)*
- Hardware - *(515)*
- Communications - *(512)*
- Supply Chain - *(437)*
- Social Engineering - *(403)*
- Physical Security - *(514)*

CAPEC – Attack Pattern "Spear Phishing"



CAPEC Common Attack Pattern Enumeration and Classification
A Community Resource for Identifying and Understanding Attacks

Home > CAPEC List > CAPEC-163: Spear Phishing (Version 3.0) ID Lookup: Go

Home | About | CAPEC List | Community | News | Search

CAPEC-163: Spear Phishing

Attack Pattern ID: 163
Abstraction: Detailed
Status: Draft

Presentation Filter: Basic

Description

An adversary targets a specific user or group with a Phishing (CAPEC-98) attack tailored to a category of users in order to have maximum relevance and deceptive capability. Spear Phishing is an enhanced version of the Phishing attack targeted to a specific user or group. The quality of the targeted email is usually enhanced by appearing to come from a known or trusted entity. If the email account of some trusted entity has been compromised the message may be digitally signed. The message will contain information specific to the targeted users that will enhance the probability that they will follow the URL to the compromised site. For example, the message may indicate knowledge of the targets employment, residence, interests, or other information that suggests familiarity. As soon as the user follows the instructions in the message, the attack proceeds as a standard Phishing attack.

Relationships

The table(s) below shows the other attack patterns and high level categories that are related to this attack pattern. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as CanFollow, PeerOf, and CanAlsoBe are defined to show similar attack patterns that the user may want to explore.

Relevant to the view "Mechanisms of Attack" (CAPEC-1000)

Execution Flow

Explore

1. **Obtain useful contextual detailed information about the targeted user or organization:** An adversary collects useful contextual detailed information about the targeted user or organization in order to craft a more deceptive and enticing message to lure the target into responding. Conduct web searching research of target. See also: CAPEC-118. Identify trusted associates, colleagues and friends of target. See also: CAPEC-118. Utilize social engineering attack patterns such as Pretexting. See also: CAPEC-407. Collect social information via dumpster diving. See also: CAPEC-406. Collect social information via traditional sources. See also: CAPEC-118. Collect social information via Non-traditional sources. See also: CAPEC-118.

Techniques
Conduct web searching research of target. See also: CAPEC-118.
Identify trusted associates, colleagues and friends of target. See also: CAPEC-118.
Utilize social engineering attack patterns such as Pretexting. See also: CAPEC-407.
Collect social information via dumpster diving. See also: CAPEC-406.
Collect social information via traditional sources. See also: CAPEC-118.
Collect social information via Non-traditional sources. See also: CAPEC-118.

CAPEC – SQL Injection

CAPEC-66: SQL Injection (Release 1.6)

SQL Injection

Attack Pattern ID: 66 *(**Standard** Attack Pattern Completeness: **Complete**)* **Typical Severity:** High **Status:** Draft

Description

Summary

This attack exploits target software that constructs SQL statements based on user input. An attacker crafts input strings so that when the target software constructs SQL statements based on the input, the resulting SQL statement performs actions other than those the application intended.

SQL Injection results from failure of the application to appropriately validate input. When specially crafted user-controlled input consisting of SQL syntax is used without proper validation as part of SQL queries, it is possible to glean information from the database in ways not envisaged during application design. Depending upon the database and the design of the application, it may also be possible to leverage injection to have the database execute system-related commands of the attacker's choice. SQL Injection enables an attacker to talk directly to the database, thus bypassing the application completely. Sucessful injection can cause information disclosure as well as ability to add or modify data in the database. In order to successfully inject SQL and retrieve information from a database, an attacker:

Attack Execution Flow

Explore

1. **Survey application**:

   The attacker first takes an inventory of the functionality exposed by the application.

   Attack Step Techniques

ID	Attack Step Technique Description	Environments
1	Spider web sites for all available links	env-Web
2	Sniff network communications with application using a utility such as WireShark.	env-ClientServer env-Peer2Peer env-CommProtocol

   Outcomes

ID	type	Outcome Description
1	Success	At least one data input to application identified.
2	Failure	No inputs to application identified. Note that just because no inputs are identified does not mean that the application will not accept any.

Experiment

1. **Determine user-controllable input susceptible to injection**:

   Determine the user-controllable input susceptible to injection. For each user-controllable input that the attacker suspects is vulnerable to SQL injection, attempt to inject characters that have special meaning in SQL (such as a single quote character, a double quote character, two hyphens, a paranthesis, etc.). The goal is to create a SQL query with an invalid syntax.

   Attack Step Techniques

ID	Attack Step Technique Description	Environments
1	Use web browser to inject input through text fields or through HTTP GET parameters.	env-Web
2	Use a web application debugging tool such as Tamper Data, TamperIE, WebScarab,etc. to modify HTTP POST parameters, hidden fields, non-freeform fields, etc.	env-Web

MITRE CAPEC

- CAPEC is a valuable resource because it provides detailed information about typical attacks, including
  - A general description of the attack
  - How to execute the attack (attack execution flow)
  - Related vulnerabilities (by CVE numbers)
  - Probing techniques
  - Attacker skill required
  - Consequences of successful exploitation
  - Solutions and mitigations
- The combined use of threat modeling and attack simulations is a research topic
  - E.g., threat modeling language for based on the MITRE Enterprise ATT&CK Matrix [1]
  - E.g., as the basis for automatic attack graph generation [2]

[1] Xiong, W., Legrand, E., Åberg, O. *et al.*
Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix.
Softw Syst Model **21,** 157–177 (2022).
https://link.springer.com/article/10.1007/s10270-021-00898-7

[2] Johnson, P., Vernotte, A., Ekstedt, M., Lagerström, R.:
pwnpr3d: an attack-graph-driven probabilistic threat-modeling approach,
Availability, Reliability and Security (ARES), 2016 11th International Conference on, pp. 278–283.
IEEE (2016)
https://ieeexplore.ieee.org/document/7784583

Vulnerability Analysis

Phases borrowed from PTES:

- **Pre-engagement interactions:** Initial communication and reasoning behind a penetration test
- **Intelligence gathering and threat modeling:** Get a better understanding of the tested organization
- **Vulnerability research, exploitation:** Identify vulnerabilities and demonstrate proof-of-concept or “real” exploits
- **Post exploitation:** Determine the value of the compromised target, maintain control and gain further access to other resources
- **Reporting:** Captures the entire process in a manner that makes sense to the customer and provides the most value to it

Vulnerability Analysis

- Vulnerability analysis is the process of discovering and confirming security issues in systems and applications which can be leveraged by an attacker
  - Vulnerabilities can range anywhere from host and service misconfiguration to insecure application design
- The process used to look for vulnerabilities varies and is highly dependent on the particular component being tested
  - Web application, network, building access, employees,..
- Possible goals of the analysis:
  - Validate that vulnerabilities exist
    - as «employee» (with credentials)
    - as «outsider» (without credentials)
    - with/without considering the impact of mitigations
  - Validate mitigation is in place and working (=not exploitable)

Vulnerability Analysis – Discovery

- Port scanners
  - An answer on TCP port 23 (telnet) might be a vulnerability
- Scanners for detecting well-known vulnerabilities
  - Product and version identification based, lookup in vulnerability database
  - Recipe / signature based, execution of the recipe/script
    - E.g., to test for a known buffer overflow
    - E.g., for misconfigurations – usage of insecure TLS versions or cipher suites
    - E.g., for policy violations - usage of common/default passwords
- Finding new vulnerabilities
  - Scanners for specific vulnerability types
    - sqlmap: SQL injection vulnerabilities
    - XSStrike: Cross-site scripting vulnerabilities
  - Web application scanners
    - Performs tests for various types of vulnerabilities found in web applications
    - Entry points (URLs) are crawled and/or supplied by the user
  - Source code scanners
    - Mainly for custom made software
  - Manual analysis of the source code
  - Manual analysis of the system under test
    - E.g., using hardening guides (CIS Benchmarks) for configuration issues

A Recipe for a Generic Vulnerability Scanner

- Determine all assets reachable over the network
  - IP addresses, open ports, domains and sub-domains
  - Using tools like nmap, sublist3r, …
- Determine the products/technologies they run
  - Using tools like nmap or technology-specific scanners
    - E.g., BuiltWith or Wappalizer for web-technologies used in a web application
- Match databases of known vulnerabilities with the list with the identified products/technologies and their versions
- Run product/technology specific recipes and/or scanners
  - Proprietary scanners (e.g., Nessus) use their own scan engine
  - "Meta-scanners" (e.g. scanmeter.io) use multiple open-source scanners and merge their results

Vulnerability Analysis - Challenges

- New vulnerability data is added continuously: when to scan?
- Analysis based on product and version can be misleading
  - Products might be forked/copied and given new names, and version numbers
    ⚠ **false negative**
  - Products might have been patched without modifying their version
    ⚠ **false positive**
- Results might depend on time/location/user account etc.
  - E.g., scan from intranet vs. extranet with different filters/controls inbetween
  - E.g., authenticated vs. unauthenticated

Explaining Common Release-Numbering Confusion

Backporting has a number of advantages for customers, but it can create confusion when it is not understood. Customers need to be aware that just looking at the version number of a package will not tell them if they are vulnerable or not. For example, stories in the press may include phrases such as "upgrade to Apache httpd 2.0.43 to fix the issue," which only takes into account the upstream version number. This can cause confusion as even after installing updated packages from a vendor, it is not likely customers will have the latest upstream version. They will instead have an older upstream version with backported patches applied.

Also, some security scanning and auditing tools make decisions about vulnerabilities based solely on the version number of components they find. This results in false positives as the tools do not take into account backported security fixes.

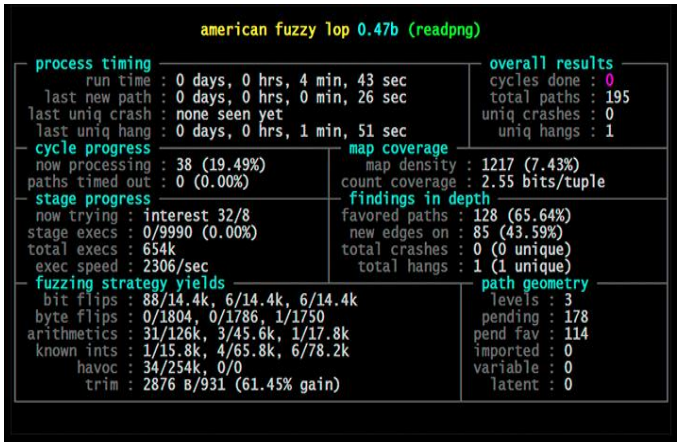
Since the introduction of Red Hat Enterprise Linux, we have been careful to explain in our security advisories how we fixed an issue, whether by moving to a new upstream version or by backporting patches to the existing version. We have attached CVE names to all our advisories since January 2000, allowing customers to easily cross-reference vulnerabilities and find out how and when we fixed them, independent of version numbers.

We also supply OVAL definitions (machine-readable versions of our advisories) that third-party vulnerability tools can use to determine the status of vulnerabilities, even when security fixes have been backported. In doing this, we hope to remove some of the confusion surrounding backporting and make it easier for customers to always keep up to date with the latest security fixes.

https://access.redhat.com/security/updates/backporting

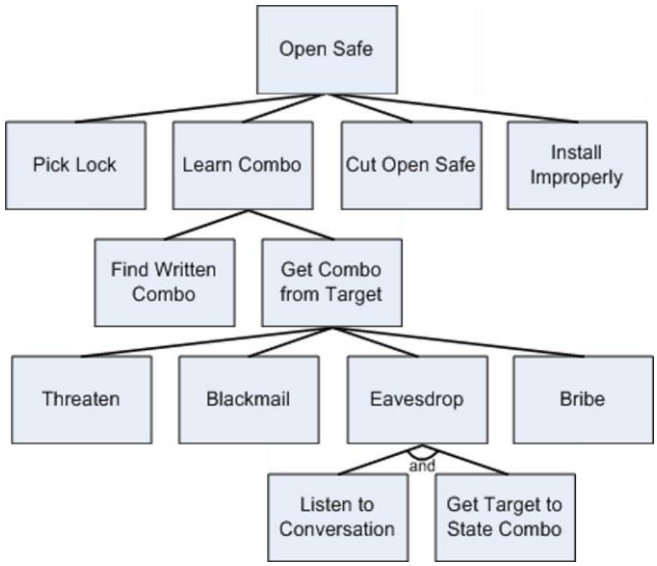
Vulnerability Analysis – Active

- Make use of fuzzers to discover yet unknown vulnerabilities
  - Black-box vs. more advanced fuzzers
  - Ideally, product is available for «offline» testing
  - Application or protocol specific fuzzers (e.g., for X.509 certificates, SOAP,…)
- Example: American fuzzy lop
  - Goal - Discover clean, interesting test cases that trigger new internal states in the targeted binary
  - Approach - Compile-time instrumentation and genetic algorithms to determine interesting input



Vulnerability Analysis - Tracking

- During a penetration test, it should be tracked what has been analyzed
- Attack trees are a suitable tool do document extensive tests with many different attack patterns
- An attack tree evolves as new systems, services and potential vulnerabilities are identified
- For teams of testers: Avoids repeating work already done by others when attack vectors "merge"



Vulnerability Scanner - nessus® (1)

- The most popular generic vulnerability scanner is nessus®
  - Generic in the sense that it tests at a wide range of "layers"
  - From OS vulnerabilities to server and web application vulnerabilities
- nessus® is a commercial product owned by Tenable®
  - A free version for non-commercial use and limited to scanning a total of 16 IPs is available
- The vulnerability tests are performed by plugins written in the Nessus Attack Scripting Language (NASL)
  - Greenbone Vulnerability Manager (GVM, formerly OpenVAS), also makes use of NASL
  - It is possible to write own plugins but Tenable does not seem to want this
    - The latest documentation on NASL is from 2005
  - Tenable Research has published 169'138 plugins
    - They cover 68'131 CVE IDs and 30'939 Bugtraq IDs (March 2022)

Vulnerability Scanner - nessus® (2)

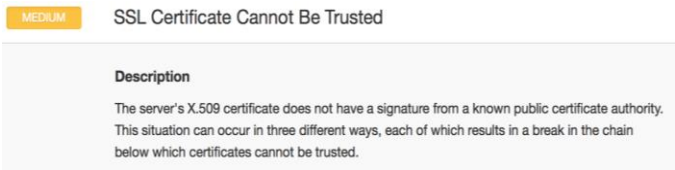
- nessus® performs the scan using one or multiple daemons
  - Daemons are usually located in the intranet
  - A scan from the Internet can complement the internal view
  - Consider detective/preventive controls:
    - Firewalls or preventive controls might impact the scan results – place daemons accordingly
    - You might not want alarms from vulnerability scanning in your detective controls
- Configuring scans (browser or command-line)
  - What IP range(s) a daemon scans
  - What scan type / plugins to use
  - User roles/profiles/credentials to use for authenticated scans
  - Disable the Safe Checks option to execute plugins belonging to unsafe categories
    - ACT_DESTRUCTIVE_ATTACK, ACT_DENIAL, ACT_KILL_HOST, ACT_FLOOD

Example: Scan of a Linux Server (1)

localhost (octopus, slides) / Hosts / localhost — Host Details

Severity ▲	Plugin Name	Plugin Family	Count
MEDIUM	SSL Certificate Cannot Be Trusted	General	4
MEDIUM	SSL Certificate with Wrong Hostname	General	3
MEDIUM	SSL Medium Strength Cipher Suites Supported	General	2
MEDIUM	SSL Weak Cipher Suites Supported	General	2
MEDIUM	Apache mod_status /server-status Information Disclosure	Web Servers	1
MEDIUM	DNS Server Cache Snooping Remote Information Disclosure	DNS	1
LOW	SSL Anonymous Cipher Suites Supported	Service detection	2
LOW	SSL RC4 Cipher Suites Supported	General	2
LOW	SSH Server CBC Mode Ciphers Enabled	Misc.	1
LOW	SSH Weak MAC Algorithms Enabled	Misc.	1
INFO	netstat portscanner (SSH)	Port scanners	14

Example: Scan of a Linux Server (2)

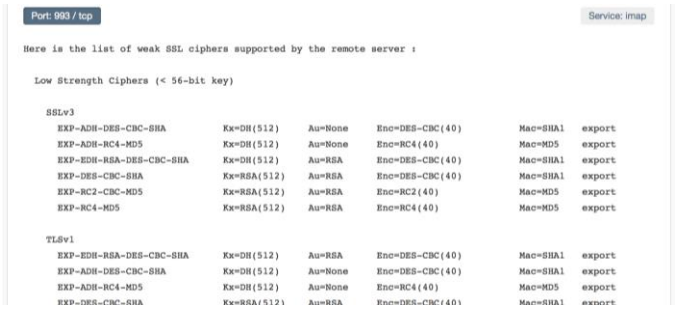


- Nessus provides detailed information about the problems detected:

- For Nessus, this is a Medium Risk, but in fact it is "wanted behaviour"
  - False positive in this case
  - But it may be a true positive in other cases

Example: Scan of a Linux Server (3)

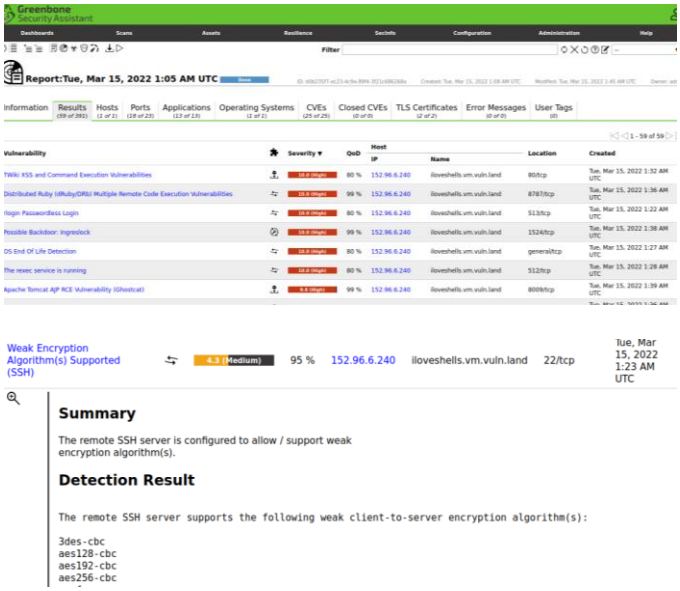
- Example of a true positive



- Ciphers with 40-bit keys are indeed supported
- Medium risk is a reasonable classification because it is not something an attacker can easily exploit

Scanning – Vulnerability Scanning (6)

- Greenbone Vulnerability Manager (GVM, formerly OpenVAS) - Many similarities with nessus
  - Originally a fork of nessus
- There are ready-to-go docker containers (free)
- Limitation: You have the community feed only which lacks vulnerabilities for many enterprise products



OpenVAS = Open Vulnerability Assessment Scanner

Greenbone Vulnerability Manager (GVM, formerly OpenVAS)

The OpenVAS project was originally initiated as a response to the commercialization of Nessus. OpenVAS built on the final open-source version of Nessus and has been significantly improved during the recent years. This origin is also the reason why GVM makes use of the Nessus Attack Scripting Language (NASL) for implementing vulnerability checks.

To achieve better visibility, less misunderstanding, and better differentiation from other OpenVAS-based products, the owner and maintainer of OpenVAS, Greenbone Networks, renamed the public vulnerability feed to Greenbone Community Feed and the feed development was internalized. Furthermore, the release scheme has been changed from a 14-day delay to a daily publication without delay, now excluding vulnerability tests for enterprise products.

Another major change was the transition to a modern infrastructure, namely GitHub and a community forum in 2018. In 2019, the branding separation was finally completed. OpenVAS now represents the actual vulnerability scanner, as it did originally, and the “S” in “OpenVAS” now stands for “Scanner” rather than “System”. These changes were accompanied by an updated OpenVAS logo. The framework in which OpenVAS is embedded is the Greenbone Vulnerability Management (GVM).

Source and more details: https://greenbone.github.io/docs/history.html

Exploitation

Phases borrowed from PTES:

- **Pre-engagement interactions:** Initial communication and reasoning behind a penetration test
- **Intelligence gathering and threat modeling:** Get a better understanding of the tested organization
- **Vulnerability research, exploitation:** Identify vulnerabilities and demonstrate proof-of-concept or “real” exploits
- **Post exploitation:** Determine the value of the compromised target, maintain control and gain further access to other resources
- **Reporting:** Captures the entire process in a manner that makes sense to the customer and provides the most value to it

Exploitation

- The exploitation phase focuses solely on establishing access to a system or resource by bypassing security restrictions
- An exploit is a piece of software, a chunk of data, or a sequence of commands that takes advantage of a vulnerability
  - It is usually used to gain control of a computer system, to extract or manipulate data, to do privilege escalation, or to DoS it
- Ultimate goal: Precision Strike
  - Attack vector(s) can be assessed in terms of their probability to succeed and the probability of getting detected when executing the attack
    - Identified in the thread modelling and vulnerability analysis phase
  - For a precision strike, choose the most promising attack vector (in terms of success probability and probability of getting detected)

Precision strike - The main focus of a penetration test is to simulate an attacker in order to represent a simulated attack against the organization. The value brought through a penetration test is generally not through smash and grab techniques where the attacks are noisy in nature and in an attempt to try every exploit. This approach may be particularly useful at the end of a penetration test to gauge the level of incident response from the organization, but in most cases the exploitation phase is a accumulation of specific research on the target.

Attack Vectors

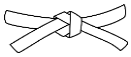
- Attack vectors to establishing access (exploits) are manifold
  - Web application attacks *[SWS1]*
    - e.g., XSS+Cookie stealing, SQL injection,…
  - Memory based exploits *[SWS2 Exploitation lecture and lab]*
    - e.g., buffer/heap overflows, use-after-free,…)
  - Man-in-the-Middle (MitM) attacks *[IS]*
  - USB/Flash Drive deployment ⚠
  - Password cracking or breaking encryption *[IS]*
  - Hardware-based (e.g., exploiting direct memory access)
- And if nothing else works exploit the human factor (social engineering)!
- The value from a penetration test comes from creativity and the ability to identify exposures and exploit them in a precise manner

USB/Flash drive deployment: Make sure that the piece of software (malware) on these media runs on the target system only. Otherwise, if you send the USB stick to the victim with Swiss Post but picked the wrong address or the postman put the stick in the wrong mailbox, it could be considered a cyber crime.

Exploitation - Challenges

- Developing exploits for vulnerabilities where there is no exploit yet
  - For example, vulnerabilities discovered in the vulnerability analysis phase using fuzzing
- Security controls that interfere with the successful execution of an attack vector
  - Anti-virus, application whitelisting, IPS, WAF,…
  - Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR), Stack Canaries,…
- Circumvention is often possible but difficult
  - Anti-Virus / IPS / WAF
    - Transform attack payload by encoding, packing or encrypting it
    - Exploit and payload is memory resident only
  - Application whitelisting
    - Processes in memory are usually not checked => process injection and in-memory malware
  - DEP, ASLR, Stack Canaries => see SWS2 Exploitation lab

Exploitation – Level of Expertise

- The exploitation phase is probably the most challenging one with respect to required expertise (at least for technical attack avenues)
- The  in exploitation:
  - Use publicly available exploits with exploitation frameworks
  - Use well-known social-engineering avenues
- The  in exploitation:
  - Use, test and modify (e.g., for evasion) publicly available exploits
  - Tune exploitation frameworks to your needs
  - Use social-engineering attacks tailored to your target
- The  in exploitation:
  - Zero-Day Angle - Reverse engineer, fuzz, analyse code to discover and exploit vulnerabilities that have not been discovered
  - Reproduce environment of target including countermeasure technology

Post Exploitation

- Post-exploitation phase: Determine the value of the compromised targets and maintain control for later use.
  - Value usually depends on the sensitivity of the data stored on the target and its usefulness in further compromising the victim.
- Is the most dangerous phase since the tester now has access to the client's (critical) systems where she can cause harm.
  - Rules of Engagement to ensure that the day to day operations and data of the client are not exposed to risk: http://www.pentest-standard.org/index.php/Post_Exploitation
- Pivoting/island hoping/lateral movement is often done in this phase
  - Refers to using a compromised system to attack other systems on the victim's network, benefiting from the "insider" position to potentially avoid some controls that may prohibit direct access to the real target

Exploitation Tools – The Metasploit Framework

Exploitation Tools and Frameworks

- Metasploit (Ruby) [free/commercial]
  - https://www.rapid7.com/products/metasploit/
- Exploit Pack (Java) [trial/commercial]
  - http://exploitpack.com/ ⚠
  - 39'000+ exploits
  - 100$/month or 900$/month (with "0-days")
- Core Certified Exploit Library [commercial]
  - https://www.coresecurity.com/core-labs/exploits
  - Claim: Verified exploits, secure and effective
- IMMUNITY CANVAS [commercial]
  - http://www.immunitysec.com
  - Over 800 exploits

Metasploits Modules (03/2022):

2205 exploits

1165 auxiliary

395 post

596 payloads

45 encoders

11 nops

9 evasion

Exploit Pack is claiming to include 39'000+ exploits, including 0-day exploits. According to their website: "Exploit Pack is a multiplatform exploitation framework with more than 39.000+ exploits, implants, agents, undetectable and ready to help you get root on your next target. Like any tool of this type, it requires some basic knowledge and expertise in the matter. Exploit Pack has been designed to be used by hands-on security professionals to support their testing process. "

However, once upon a time the original author of the exploit pack wrote: "So, basically the packs are a bunch of exploits that I grab all over the internet, but having always the quantity in mind, instead of quality, because what I learn from my experience is that I prefer to have everything instead of just one exploit that works with that technology etc. Everything that you could find in exploit-db, packetstorm, full disclosure, etc will be in the professional pack some of them are tweaked by me, others are just raw code that need attention before (you) run it by a click but hey, you will have 33400+ exploits in your arsenal». Furthermore, while it is basically open-source and anyone can check for it, there are some rumours that it might contain «unwanted» functionality."

The current version is not open-source anymore and includes 0-days. However, how likely is it, that this statement is true? That would be really cheap 0-days, after all...

Metasploit Framework (MSF)

- The Metasploit Framework (MSF) is an advanced open-source platform for developing, testing, and using exploit code
- It is known for releasing some of the most technically sophisticated exploits for many of the vulnerabilities known to the public
- Among its many other features are its capability to interoperate with other relevant tools like port- or vulnerability scanners
- Versions
  - Metasploit Framework (free)
    - Web-based GUI has been discontinued in 2019 for the free version
  - Metasploit Pro (non-free)
    - Web-based GUI, automation, reporting, IDS/IPS/AV evasion, wizards, evidence collection,…
  - Compare: https://www.rapid7.com/products/metasploit/download/editions/

Console Interface - msfconsole

- All-in-one console interface to the frameworks' functions
  - Run in quite mode (no banner):
    `> msfconsole -q`
  - Full readline support, tabbing and command completion
  - Execution of external commands is possible
  - List and description of core commands: https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/

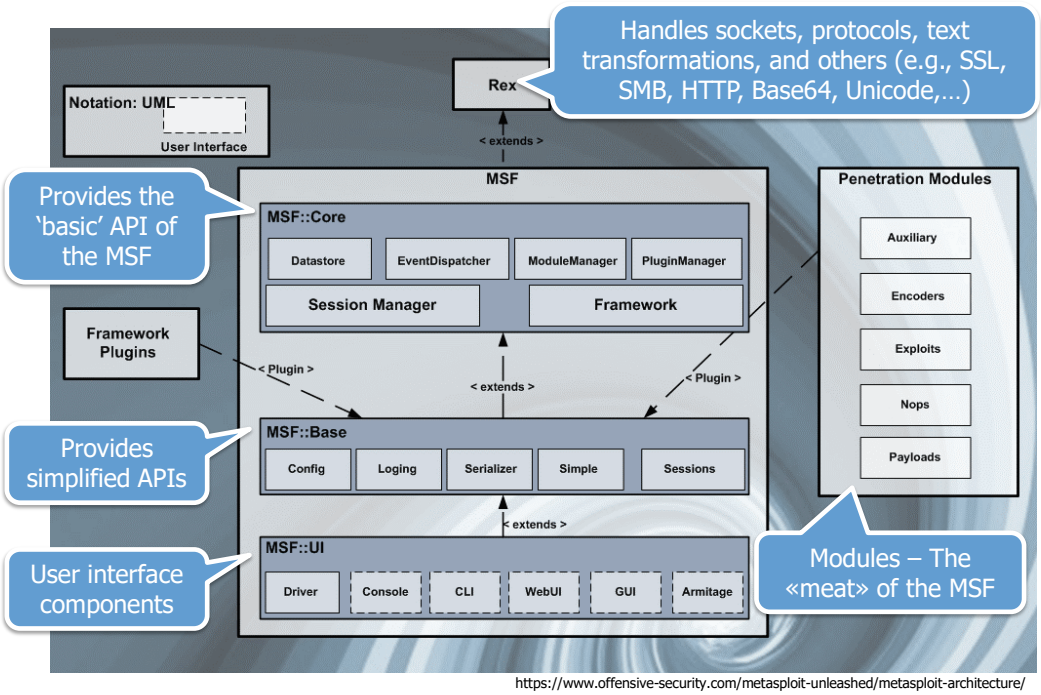
```
msf6 > help

Core Commands
=============

    Command       Description
    -------       -----------
    ?             Help menu
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    debug         Display information useful for debugging
    exit          Exit the console
    features      Display the list of not yet released features that can be op
                  ted in to
    get           Gets the value of a context-specific variable
    getg          Gets the value of a global variable
    grep          Grep the output of another command
    help          Help menu
    history       Show command history
    load          Load a framework plugin
    quit          Exit the console
    repeat        Repeat a list of commands
    route         Route traffic through a session
    save          Saves the active datastores
    sessions      Dump session listings and display information about sessions
    set           Sets a context-specific variable to a value
    setg          Sets a global variable to a value
    sleep         Do nothing for the specified number of seconds
    spool         Write console output into a file as well the screen
    threads       View and manipulate background threads
    tips          Show a list of useful productivity tips
    unload        Unload a framework plugin
    unset         Unsets one or more context-specific variables
    unsetg        Unsets one or more global variables
    version       Show the framework and console library version numbers
```


The GNU Readline Library is a utility which aids in the consistency of user interface across discrete programs which provide a command line interface.

More details can be found here:

http://cnswww.cns.cwru.edu/php/chet/readline/readline.html#Top

Metasploit Architecture - Overview



https://www.offensive-security.com/metasploit-unleashed/metasploit-architecture/

Metasploit Plugins

- Plugins work directly with the API
  - They manipulate the framework as a whole
  - Plugins hook into the event subsystem
  - They automate specific tasks which would be tedious to do manually
- Plugins only work in the msfconsole
  - Plugins can add new console commands
  - They extend the overall Framework functionality

Metasploit Mixins

Aside from plugins, there is the concept of mixins referring to Ruby's concept of 'including' one class into another.

Mixins can override (a subset of) a class' methods or simply add new methods/features or allow modules to have different 'flavors'.

connect() is e.g. implemented by the TCP mixin and then overloaded by FTP, SMB and others.

The Scanner mixin overloads run() and changes run() for run_host() and run_range()

Metasploit Object Model and Modules

- The MSF is composed of modules
  - All modules are Ruby classes inheriting from the type-specific class
- Module types:
  - Exploits – Can exploit a vulnerability and place and execute a payload
    - Exploits that do not allow to place and execute a payload on the target, are of the auxiliary type (e.g., that cause a DoS only)
  - Auxiliary – Include port scanners, fuzzers, sniffers, and more.
  - Payloads – Payloads usually consist of some form of code
  - Encoders - Encoders ensure that payloads make it to their destination
  - Nops - Nops keep the payload sizes consistent
- Location of modules in Kali distros:
  - /usr/share/metasploit-framework/modules/
  - Custom modules (user-built): ~/.msf4/modules/

There are a few types of modules. The module type depends on the purpose of the module and the type of action that the module performs. The following are module types that are available in the Metasploit Framework:

- **Exploit** - An exploit module executes a sequence of commands to target a specific vulnerability found in a system or application. An exploit module takes advantage of a vulnerability to provide access to the target system. Exploit modules include buffer overflow, code injection, and web application exploits.
- **Auxiliary** - An auxiliary module does not execute a payload. It can be used to perform arbitrary actions that may not be directly related to exploitation. Examples of auxiliary modules include scanners, fuzzers, and denial of service attacks.
- **Post-Exploitation** - A post-exploitation module enables you to gather more information or to gain further access to an exploited target system. Examples of post-exploitation modules include hash dumps and application and service enumerators.
- **Payload** - A payload is the shell code that runs after an exploit successfully compromises a system. The payload enables you to define how you want to connect to the shell and what you want to do to the target system after you take control of it. A payload can open a Meterpreter or command shell. Meterpreter is an advanced payload that allows you to write DLL files to dynamically create new features as you need them.
- **NOP generator** - A NOP generator produces a series of random bytes that you can use to bypass standard IDS and IPS NOP sled signatures. Use NOP generators to pad buffers.

Source: https://docs.rapid7.com/metasploit/msf-overview

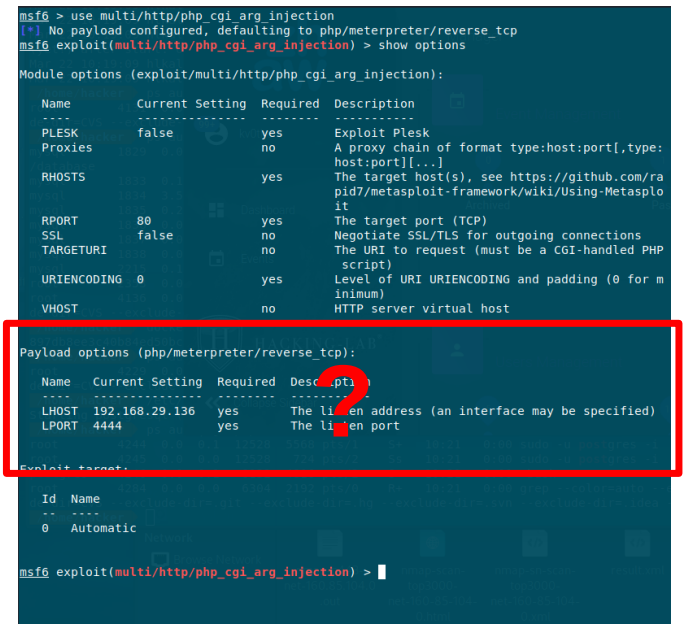
Exploits

- Active exploits exploit a specific host, run until completion and exit
  - Brute-force modules exit when a shell opens from the victim
  - Module execution stops if an error is encountered
  - -j: run exploit in background
- Passive exploits wait for incoming connections from client software
  - Exploit modules in exploit/…/browser/
    - All types of clients, not just browsers!
  - Shells get created in the background:
    - -l: Show the sessions
    - -i: Interact with a shell.

```
msf6 exploit(windows/browser/adobe_geticon) > search type:exploit path:browser -s disclosure_date -r

Matching Modules
================

   #   Name                                                                  Disclosure Date  Rank
   -   ----                                                                  ---------------  ----
   0   exploit/multi/browser/chrome_cve_2021_21220_v8_insufficient_validation  2021-04-13       manual
   1   exploit/osx/browser/osx_gatekeeper_bypass                             2021-03-25       manual
   2   exploit/multi/browser/chrome_simplifiedlowering_overflow              2020-11-19       manual
   3   exploit/osx/browser/safari_ln_operator_side_effect                    2020-03-18       manual
   4   exploit/multi/browser/chrome_jscreate_sideeffect                      2020-02-19       manual
   5   exploit/windows/browser/chrome_filereader_uaf                         2019-03-21       manual
   6   exploit/multi/browser/chrome_array_map                                2019-03-07       manual
   7   exploit/multi/browser/chrome_object_create                            2018-09-25       manual
   8   exploit/multi/browser/msfd_rce_browser                                2018-04-11       normal
   9   exploit/osx/browser/safari_proxy_object_type_confusion                2018-03-15       manual
   10  exploit/apple_ios/browser/webkit_createthis                           2018-03-15       manual
   11  exploit/windows/browser/exodus                                        2018-01-25       manual
   12  exploit/windows/browser/cisco_webex_ext                               2017-01-21       great
   13  exploit/windows/browser/firefox_smil_uaf                              2016-11-30       normal
   14  exploit/apple_ios/browser/webkit_trident                              2016-08-25       manual
   15  exploit/apple_ios/browser/safari_jit                                  2016-08-25       good
   16  exploit/windows/browser/samsung_security_manager_put                  2016-08-05       excellent
   17  exploit/windows/browser/ms16_051_vbscript                             2016-05-10       normal
   18  exploit/osx/browser/adobe_flash_delete_range_tl_op                    2016-04-27       great
   19  exploit/osx/browser/safari_user_assisted_applescript_exec             2015-10-16       manual
   20  exploit/android/browser/stagefright_mp4_tx3g_64bit                    2015-08-13       normal
   21  exploit/multi/browser/adobe_flash_opaque_background_uaf               2015-07-06       great
   22  exploit/multi/browser/adobe_flash_hacking_team_uaf                    2015-07-06       great
   23  exploit/multi/browser/adobe_flash_nellymoser_bof                      2015-06-23       great
   24  exploit/multi/browser/adobe_flash_shader_job_overflow                 2015-05-12       great
   25  exploit/multi/browser/adobe_flash_shader_drawing_fill                 2015-05-12       great
   26  exploit/multi/browser/firefox_pdfjs_privilege_escalation              2015-03-31       manual
   27  exploit/multi/browser/adobe_flash_net_connection_confusion            2015-03-12       great
   28  exploit/windows/browser/adobe_flash_worker_byte_array_uaf             2015-02-02       great
   29  exploit/windows/browser/x360_video_player_set_text_bof                2015-01-30       normal
   30  exploit/windows/browser/malwarebytes_update_exec                      2014-12-16       good
   31  exploit/windows/browser/adobe_flash_pcre                              2014-11-25       normal
   32  exploit/windows/browser/ms14_064_ole_code_execution                   2014-11-13       good
   33  exploit/android/browser/samsung_knox_smdm_url                         2014-11-12       excellent
   34  exploit/windows/browser/adobe_flash_uncompress_zlib_uninitialized     2014-11-11       good
   35  exploit/windows/browser/adobe_flash_cas132_int_overflow               2014-10-14       great
   36  exploit/windows/browser/adobe_flash_copy_pixels_to_byte_array         2014-09-23       great
   37  exploit/windows/browser/advantech_webaccess_dvs_getcolor              2014-07-17       normal
   38  exploit/multi/browser/adobe_flash_uncompress_zlib_uaf                 2014-04-28       great
   39  exploit/multi/browser/adobe_flash_pixel_bender_bof                    2014-04-28       great
   40  exploit/windows/browser/adobe_flash_domain_memory_uaf                 2014-04-14       great
   41  exploit/multi/browser/firefox_webidl_injection                        2014-03-17       excellent
   42  exploit/windows/browser/ms14_012_textrange                            2014-03-11       normal
   43  exploit/osx/browser/safari_user_assisted_download_launch              2014-03-10       manual
   44  exploit/windows/browser/getgodm_http_response_bof                     2014-03-09       normal
   45  exploit/windows/browser/ms14_012_cmarkup_uaf                          2014-02-13       normal
   46  exploit/windows/browser/adobe_flash_avm2                              2014-02-05       normal
   47  exploit/multi/browser/firefox_proxy_prototype                         2014-01-20       manual
   48  exploit/windows/browser/wellintech_kingscada_kxclientdownload         2014-01-14       good
```

Using Exploits

- Searching for exploits:
  `search type:exploit <keyword>`
- Selecting an exploit for use:
  `use <exploit identifier>`
- Show configuration options:
  `show options`
- Show information about exploit:
  `show info`
- Additional search options:
  `help search`



Payloads

- Main payload types: single, stagers and stages.
  - Singles are payloads that are self-contained (fire-and-forget)
    - Victim does not have to connect back to the attacker
    - Payload can be quite big, vulnerability must allow submission/injection of many bytes
    - E.g., payload adding a user to the target system or running calc.exe
  - Stagers setup a network connection between attacker and victim to fetch more stuff
    - Designed to be small to work with vulnerabilities that allow to submit/inject few bytes only
  - Stages are payload components downloaded by Stagers modules
    - E.g., Meterpreter, VNC Injection, iPhone 'ipwn' Shell

Reflective DLL Injection is a technique whereby a stage payload is injected into a compromised host process running in memory, never touching the host hard drive.

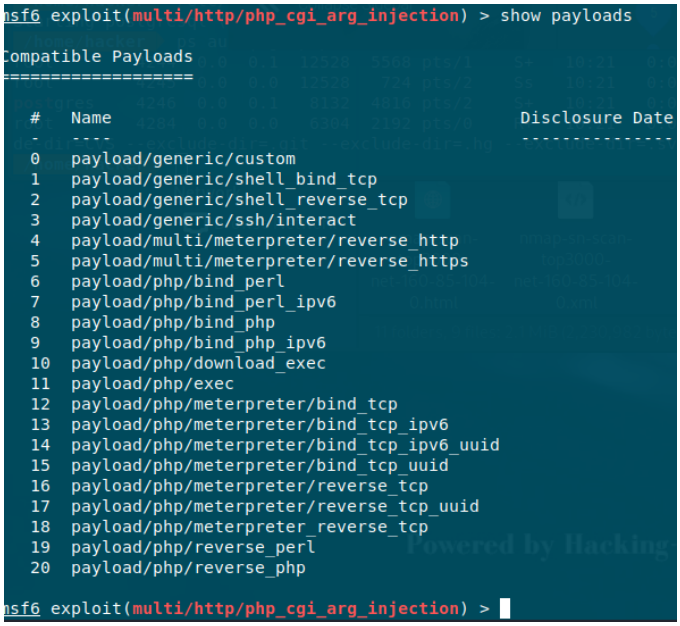
The VNC and Meterpreter payloads both make use of reflective DLL injection. You can read more about this from Stephen Fewer, the creator of the reflective DLL injection method:

http://blog.harmonysecurity.com/2008/10/new-paper-reflective-dll-injection.html

Payloads

- Use with an exploit:
  - Show payloads for this exploit:
    `show payloads`
  - Use a specific payload
    `set payload <payload>`
  - Show payload configuration options:
    `show options`
- Payloads as a standalone executable:

```
use <payload>
generate -f exe -o mlwr.exe …
```



Meterpreter Payload (1)

- With an old school «shell» you have
  - Poor automation support
  - Reliant on the shell's built-in commands
  - Limited to installed applications
- With Meterpreter things get easy
- Need to upload a file?
  `meterpreter> upload bkdr.exe bkdr.exe`
- Need to execute a file?
  `meterpreter> execute -H  -f bkdr.exe`
- Tools on your machine need to talk to systems in the victim's internal network 192.168.2.0/24 using your meterpreter session with id 10?
  `msf6> route add 192.168.2.0 255.255.255.0 10`

Usage: **execute** -f file [options]
Executes a command on the remote machine.
OPTIONS:

```
    -H        Create the process hidden from view.
    -a <opt>  The arguments to pass to the command.
    -c        Channelized I/O (required for interaction).
    -d <opt>  The 'dummy' executable to launch when using -m.
    -f <opt>  The executable command to run.
    -h        Help menu.
    -i        Interact with the process after creating it.
    -k        Execute process on the meterpreters current desktop
    -m        Execute from memory.
    -s <opt>  Execute process in a given session as the session user
    -t        Execute process with currently impersonated thread token
```

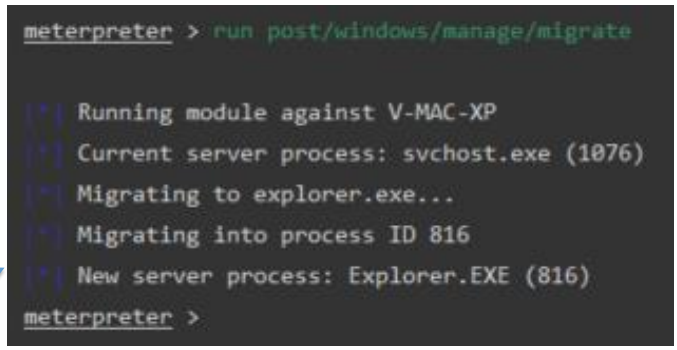
Usage: **upload** [options] src1 src2 src3 ... destination
Uploads local files and directories to the remote machine.

OPTIONS:

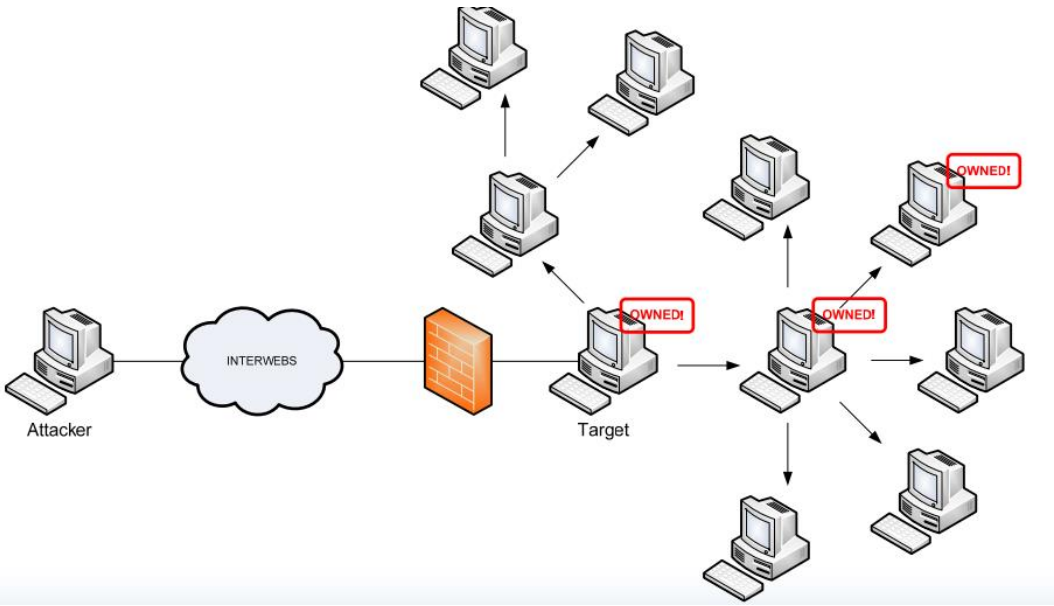
-h Help banner
-r Upload recursively

Meterpreter Payload (2)

- An advanced post-exploitation system
  - Based on library injection technology
  - Stealthy - No disk access and no new process
  - In-memory DLL injection
- What you can do with Meterpreter
  - Command execution & manipulation
  - In-memory process migration
  - Registry interaction
  - File system interaction
  - Complete API scripting
  - Network pivoting & port forwarding

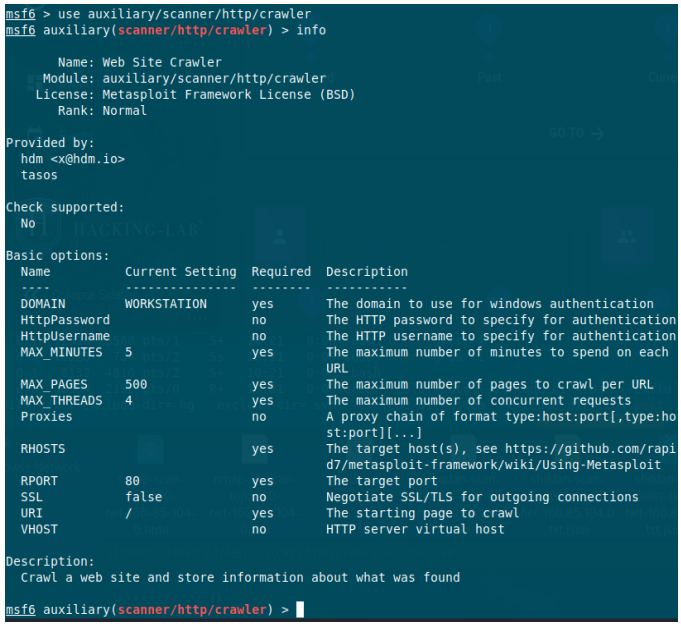


Meterpreter - Pivoting



Auxiliaries (exploits without payload and more)

- Auxiliaries
  - Denial of service methods and exploits
  - Information leakage exploits
  - Discovery and fingerprinting
  - Network protocol "fuzzers"
  - Brute Forcing
- Examples
  - discovery/udp_sweep UDP Service Sweeper
  - http/http_version HTTP Version Detection
  - http/http_put HTTP Writable Path PUT/DELETE
  - mssql/mssql_login MSSQL Login Utility
  - mssql/mssql_ping MSSQL Ping Utility



Post Exploitation Modules - Categories

• gather	Modules that involve data gathering/collecting/enumeration.
• gather/credentials	Modules that steal credentials.
• gather/forensics	Modules that involve forensics data gathering.
• manage	Modules that modifies/operates/manipulates something on the system. Session management related tasks such as migration, injection
• recon	Modules to learn more about the system in terms of reconnaissance. No data stealing.
• escalate	Privilege escalation modules (not exploit-based => exploits)
• wlan	Modules that are for WLAN related tasks.
• capture	Modules that involve monitoring something for data collection. For example: key logging.

https://github.com/rapid7/metasploit-framework/wiki/How-to-get-started-with-writing-a-post-module

- Be careful when installing backdoors during penetration tests
- Someone else might connect to this "service" as well!
- To maintain access and avoid detection:
  - Deploy well-known remote management tools
  - Use built-in tools/mechanisms
  - Use custom (fileless) backdoors
  - Apply obfuscation/morphing to known tools
  - …

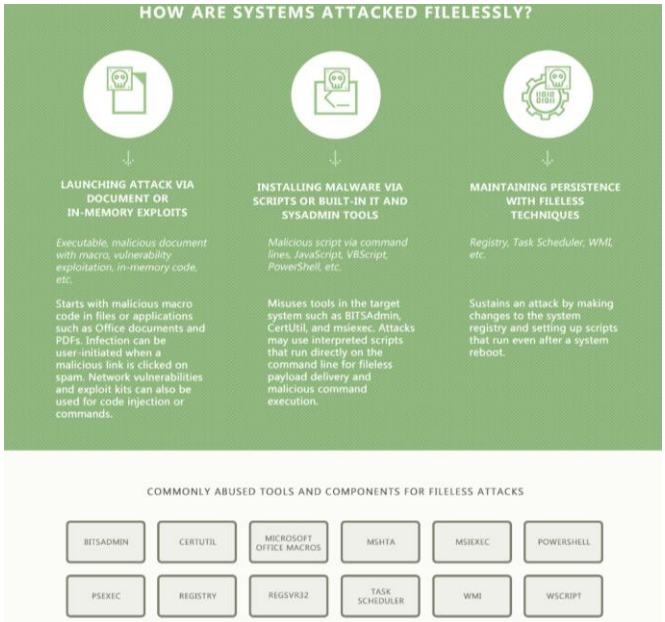


Image source: https://documents.trendmicro.com/assets/infographics/Fileless-Threats-101-infographic.jpg

Summary

- The phases threat modeling, vulnerability analysis, exploitation and post exploitations involve many activities, skills and challenges
  - We scratched only the surface
  - We gave a basic idea and pointed out some methods and (types of) tools
- Vulnerability scanners do not work perfectly, there are legitimate reasons why they have false positives and false negatives
- The Metasploit framework is a tool that makes exploiting well-known vulnerabilities (and more) easy