OWASP Top 10

OWASP Top 10 Vulnerabilities Overview: 2021

A01. Broken Access Control

A02. Cryptographic Failures

A03. Injections

A04. Insecure Design

A05. Security Misconfigurations

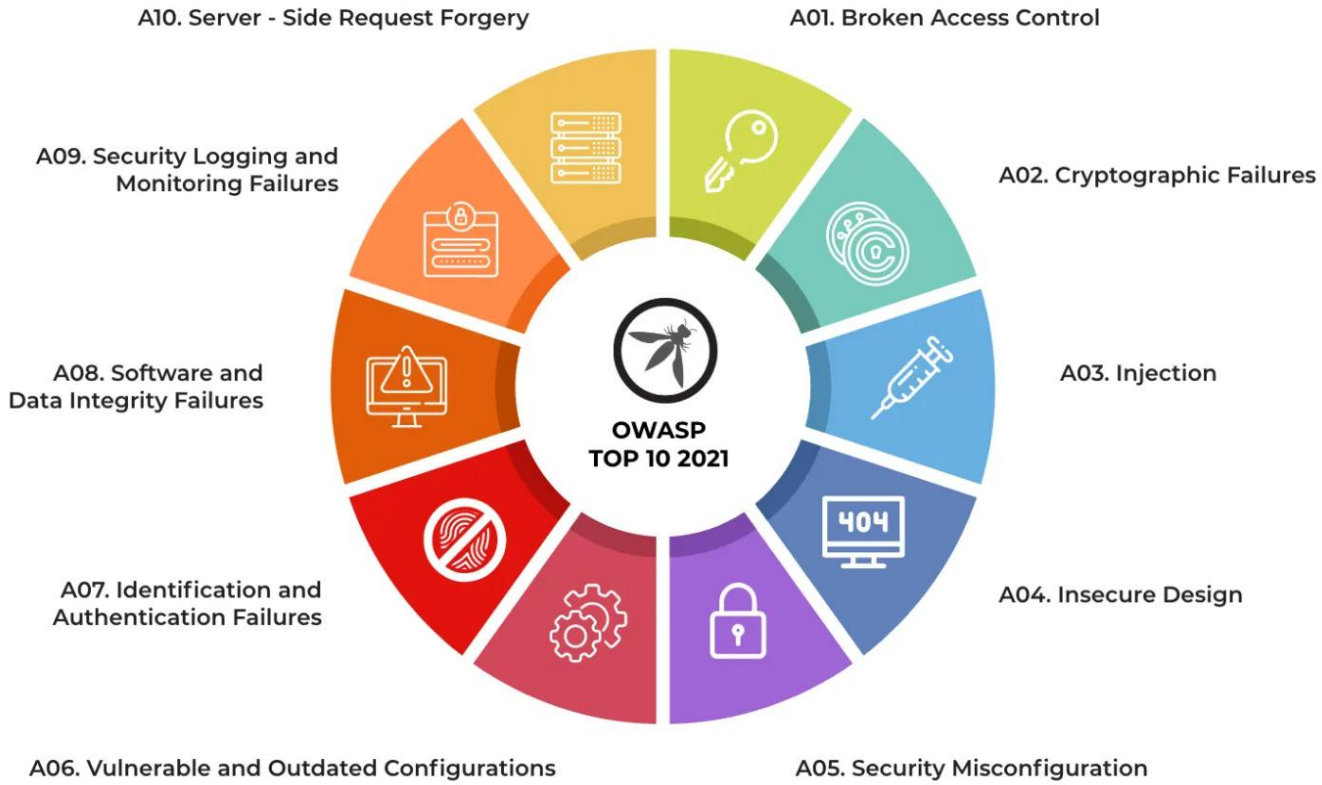
A06. Vulnerable and Outdated Components

A07. Identification and Authentication Failures

A08. Software and Data Integrity Failures

A09. Security Logging and Monitoring Failures

A10. Server-Side Request Forgery (SSRF)



Hands-On Labs: TryHackMe (tryhackme.com)

OWASP Top 10

A01. Broken Access Control: broken access control means that an attacker can access resources or data they should not have access to.

A02. Cryptographic Failures: unencrypted or weakly encrypted data in transit or data at rest.

A03. Injections: occur when user input is not properly sanitized before being used in an SQL query (SQL Injection) or injecting code into a web page (Cross-site scripting (XSS)).

A04. Insecure Design: refers to security flaws introduced during software development's design phase.

A05. Security Misconfigurations: leaving default credentials, not disabling directory listing etc.

A06. Vulnerable and Outdated Components: having known vulnerabilities & no longer security patches

A07. Identification and Authentication Failures: fails to correctly identify and/or authenticate a user.

A08. Software and Data Integrity Failures: when data is modified or destroyed without authorization.

A09. Security Logging and Monitoring Failures: Logging failures can lead to data loss or corruption while monitoring failures allow attackers to operate undetected.

A10. Server-Side Request Forgery (SSRF): the attacker tricks a server into requesting on their behalf.

ZAP

- **Zed Attack Proxy (ZAP)** by Checkmarx is the world's most widely used free and open source web app scanner.
  - Available at https://www.zaproxy.org/download
- It can help you automatically find security vulnerabilities in your web applications while you are developing and testing your applications. It's also a great tool for experienced pentesters to use for manual security testing.
  - ❖Automated Scan
  - ❖Directory Bruteforce
  - ❖Authenticated Scan
  - ❖Login Page Bruteforce
  - ❖Install ZAP Extensions
- This https://tryhackme.com/room/learnowaspzap will be using OWASP Zap against the DVWA machine, feel free to deploy your own instance and follow along.

Nessus

- Nessus® is as one of the **most widely deployed security technologies** on the planet - and the gold standard for vulnerability assessment.
  - Available at https://www.tenable.com/products/nessus
- Nessus® provides a penetration tester with a wealth of capabilities that will assist in the engagement, such as:
  - ❖ Identifying local and remote vulnerabilities
  - ❖ Configuration and compliance audits
  - ❖ Checking for default credentials
  - ❖ Web application scanning
- Nessus® Essentials allows you to scan your environment (up to **16** IP addresses per scanner) with the same high-speed, in-depth assessments.
- Nessus® isn't installed on Kali Linux by default, but this video will show you how to install Nessus on Kali Linux: https://www.youtube.com/watch?v=FcW2s7VpBio

Gaining Access

Gaining Access

The attack phase to gain access to a target can take many different forms and serve different purposes, such as:

- **Stealing Confidential Data,**
- **Tampering with Data**
- **Compromising the Availability of a Resource,**
- **Obtaining Unauthorized Access to a System.**

Sniffing

- Sniffing is a passive attack that attempts to compromise the confidentiality of information.
- It might be considered part of reconnaissance (e.g., sniffing to learn passwords) prefacing an attack but can just as well be argued to be an attack to gain access to information.
- Sniffer can eavesdrop on everything transmitted on the LAN including user names, passwords, DNS queries, e-mail messages, and all types of personal data.
- Some free and commercial sniffing tools:
  - Wireshark, Tcpdump, Windump,
  - Snort, Ethereal,
  - Sniffit, and dsniff.

Sniffing

- **Wireshark** is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education.
- It lets you see what's happening on your network at a microscopic level and is the de facto standard across many commercial and non-profit enterprises, government agencies, and educational institutions.
- **Wireshark** has a rich feature set which includes the following:
  - Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD
  - Live capture and offline analysis
  - Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
  - The most powerful display filters in the industry
  - Read/write many different capture file formats
  - Live data can be read from different types of networks, including Ethernet, IEEE 802.11, Bluetooth and loopback.
  - Coloring rules can be applied to the packet list for quick, intuitive analysis

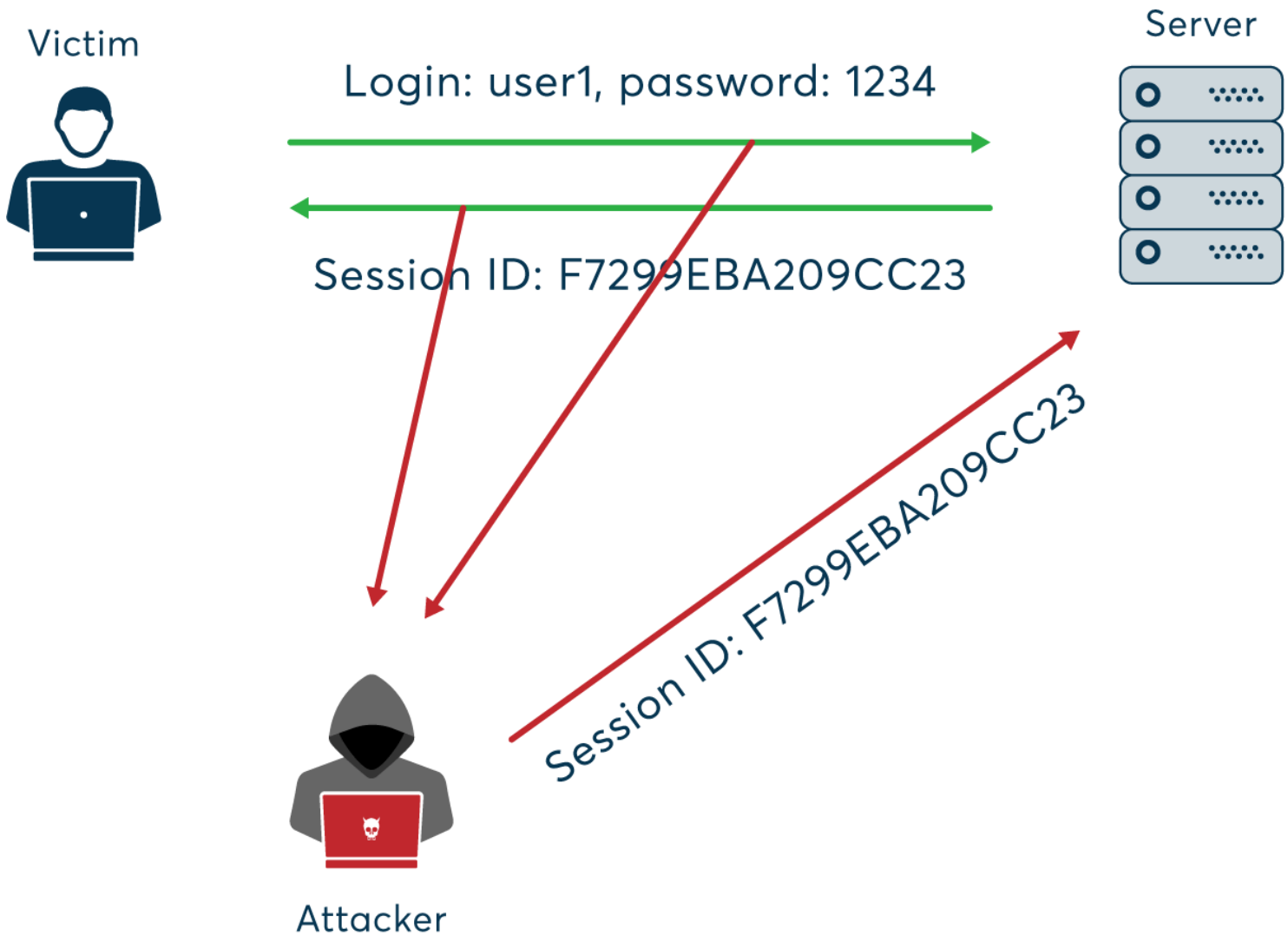
Session Hijacking

- It is a combination of sniffing and address spoofing that enables the compromise of a user's remote login session, thus providing an attacker unauthorized access to a machine with the privileges of the legitimate user.
- Address spoofing is sending a packet with a **fake source address**.
- This is quite simple because the sender of an IP packet writes the IP source address in the packet header.
- Address **spoofing** enables attackers to masquerade as another person.
- If a user is currently engaged in an interactive login session (e.g., through telnet, rlogin, FTP), a session hijacking tool *allows an attacker to steal the session*.

Session Hijacking

- When most hijack victims see their login session disappear, they usually just assume that the cause is network trouble and try to login again, unaware of the hijacking attack.
- Session hijacking tools: Burp Suite, Ettercap, OWASP ZAP, BetterCAP, netool toolkit, WebSploit Framework, sslstrip, Jhijack, Cookie Cadger, CookieCatcher, hamster, Firesheep.
- The hijacking attack begins with the attacker sniffing packets of an interactive session between two hosts, carefully noting the TCP sequence numbers of all packets.
- The proper TCP sequence numbers must be used for the attack to work, because the receiving host must be convinced to accept the faked packets from the attacker.

Session Hijacking



Password Attack

- This attack attempt to gain access to a host or service with the privileges of a current user.
- Passwords continue to be very frequently used for access control despite their major weakness: if a password is guessed or stolen, an attacker could gain complete access.
- The most well-protected systems could be compromised by **a single weak password.**
- Many attacks are often directed at guessing or bypassing passwords.
- Easy passwords to guess are the default passwords installed by many operating systems and service applications.
- They are sometimes *overlooked or ignored by system administrators*.

Password Attack: Default Password Collection

- https://www.softwaretestinghelp.com/default-router-username-and-password-list/
- https://192-168-1-1ip.mobi/default-router-passwords-list/
- https://proprivacy.com/guides/default-router-login-details
- https://datarecovery.com/rd/default-passwords/
- https://cirt.net/passwords
- https://www.defpass.com/index.php

Password Attack

- Computer systems store a list of user accounts and passwords in a password file, but the information is encrypted or hashed for protection against attackers.

- If an attacker can obtain the password file, the attacker has the advantage of time to crack the passwords by brute force.

- **Brute-force attack is method where a program attempts to try every possible combination until it cracks the password.**

- Brute-force password guessing can be very time consuming but is often not necessary.

Password Attack

- A dictionary attack takes advantage of this tendency by guessing a set of common words and names.
- Password cracking programs usually come equipped with **"dictionaries"**, or word lists, with thousands or even millions of entries of several kinds, including:

  > Words in various languages
  > names of people
  > Places
  > Commonly used passwords etc

- More sophisticated hybrid password guessing tools combine dictionary attacks with limited brute-force attacks.

- Hybrid approach begin with guesses of common words but then methodically add characters to words to form new guesses.

Password Attack

- Cryptanalysis attack method for password cracking works by comparing a precomputed encrypted password to a value in a look-up table.

- If attacker have to calculate the hash of every word in a dictionary in dictionary attack or every password combination in brute-force attack, this takes additional computing power.

- **Rainbow Table** is a database that contains pre-computed value used to more quickly break a password since values don't have to be calculated for password being guessed.

- **CrackStation** have a 190GB, 15-billion-entry lookup table for MD5 and SHA1 hashes. (https://crackstation.net/)

Salted Password Hashing

- **Lookup tables** and **rainbow tables** only work because each password is hashed the exact same way. If two users have the same password, they'll have the same password hashes. We can prevent these attacks by randomizing each hash, so that when the same password is hashed twice, the hashes are not the same.

- We can randomize the hashes **by appending or prepending a random string**, called a **salt**, to the password before hashing. To check if a password is correct, we need the salt, so it is usually stored in the user account database along with the hash, or as part of the hash string itself.

```
hash("hello")                          = 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
hash("hello" + "QxLUF1bgIAdeQX")  = 9e209040c863f84a31e719795b2577523954739fe5ed3b58a75cff2127075ed1
hash("hello" + "bv5PehSMfV11Cd")  = d1d3ec2e6f20fd420d50e2642992841d8338a314b8ea157c9e18477aaef226ab
hash("hello" + "YYLmfY6IehjZMQ")  = a49670c3c18b9e079b9cfaf51634f563dc8ae3070db2c4a8544305df1b60f007
```

- An attacker won't know in advance what the salt will be, so they can't pre-compute a lookup table or rainbow table.

Salted Password Hashing

- $ mkpasswd -m md5 [employs salting i.e md5-crypt]
- $ mkpasswd -m SHA-512 [employs salting i.e sha512-crypt]

- An **md5-crypt** hash string has the format ***1salt$checksum***, where:
  - *1* is the prefix used to identify md5-crypt hashes {**1** for MD5, **2a** for Blowfish (on some Linux distributions), **5** for SHA-256 , **6** for SHA-512}.
  - *salt* is **0-8 characters** drawn from the regexp range [./0-9A-Za-z]; providing a 48-bit salt (5pZSV9va in the example).
  - *checksum* is **22 characters** drawn from the same character set as the salt; encoding a 128-bit checksum (azfrPr6af3Fc7dLblQXVa0 in the example).

Password Attack Tools

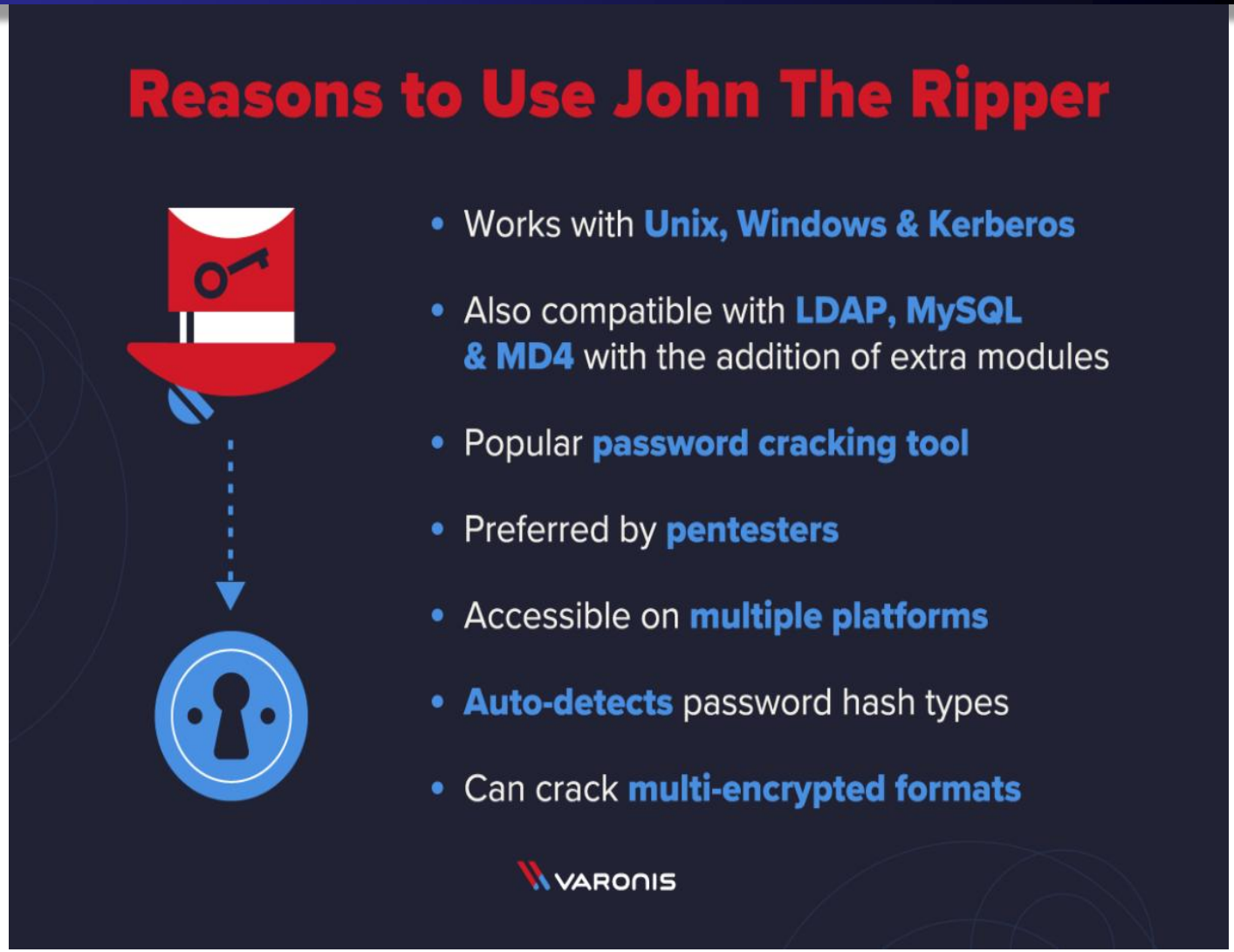
- **Hydra**
- CeWL
- **Hashcat**
- Crunch
- Medusa
- **John the Ripper**
- Ncrack
- Ophcrack
- **Wordlists**
- Mimikatz, etc.

John the Ripper (JtR)

- **John the Ripper** is an open source password cracker used for Dictionary-style attacks.

- **JtR** also includes its own wordlists of common passwords for **20+ languages**.

- **JtR** supports several common encryption technologies out-of-the-box for UNIX and Windows-based systems.



Usage of JtR

- The */etc/passwd* file is a text-based database of information about users that may log into the system. The */etc/shadow* is used to increase the security level of passwords by restricting all but highly privileged users' access to hashed password data.

- The **JtR** requires whatever it wants to crack to be in a specific format. To convert the *passwd*, and *shadow* files, we need to leverage the ***/usr/sbin/unshadow*** executable.

```
$ sudo /usr/sbin/unshadow /etc/passwd /etc/shadow > passwords.txt
```

- And the command to crack your Linux passwords is simple enough. To perform the crack execute the following:

```
$ sudo gzip -d /usr/share/wordlists/rockyou.txt.gz
$ /usr/sbin/john --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
```

Hashcat

- **Hashcat** claims to be the world's fastest CPU and GPU-based password **recovery** tool.

- It is cross-platform, and available on Windows, macOS and Linux.

- Currently, it supports 237 different hash types including LM Hash, NT hash, MD4, MD5, SHA-1, and many more.

- Usage: $ hashcat –a 0 –m 1000 ./hashes/ntlm.txt ./wordlists/rockyou.txt

Hashcat

- **Argument 1 of 4**:
  - **-a (or --attack mode)**
    - Tells the method to crack the password, e.g., using a *dictionary*, *combinatory*, or *brute-force*
- **Argument 2 of 4:**
  - **-m (or --hash-type)**
    - Type of hash, e.g., MD4, SHA1, SHA512, NTLM etc.
- **Argument 3 of 4:**
  - [**filename** | **hash**]
    - Specifies a file containing the hash(es) you intend to crack.
- **Argument 4 of 4:**
  - [**dictionary**| **mask** | **directory**]
    - Specifies a dictionary (wordlist), *mask*, or *directory* to be used.
- Usage: $ hashcat –a 0 –m 1000 ./hashes/ntlm.txt ./wordlists/rockyou.txt

Hashcat: Power of Mask

- Dictionaries can be a great tool in your password cracking arsenal, but they can hit some very real limits quickly. By using a **mask**, you can specify what combinations of characters to run through during your attack.
- For some reason, you have additional information that the password is **only** made with *lowercase letters,* and ends with *two numbers* and has exactly *6 characters*.

5 predefined character sets in Hashcat

```
?l = abcdefghijklmnopqrstuvwxyz
?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d = 0123456789
?s =  !"#$%&'()*+,-./:;<=>?@[]^_`{|}~
?a = ?l?u?d?s
```

- For this, standard brute-force would require running through all combinations from one letter up, and using uppercase and special characters that we know will never match. Using a **mask**, we can simply use the following to skip all of the extra combinations we know will fail.

$ hashcat -a 3 -m 0 example0.hash ?l?l?l?l?d?d

- It's a lot easier than it looks, and it's **ridiculously powerful** and is a definite game-changer.

Hashcat: The Art of Word Mangling

- The **rule-based attack** is like a **programming language** designed for password candidate generation. Hashcat's rules are located in the ***rules*** directory of your Hashcat installation.
- Rules used in conjunction with wordlists are generally faster than using a large wordlist that contained all of the candidates that would be produced by a rule based attack.

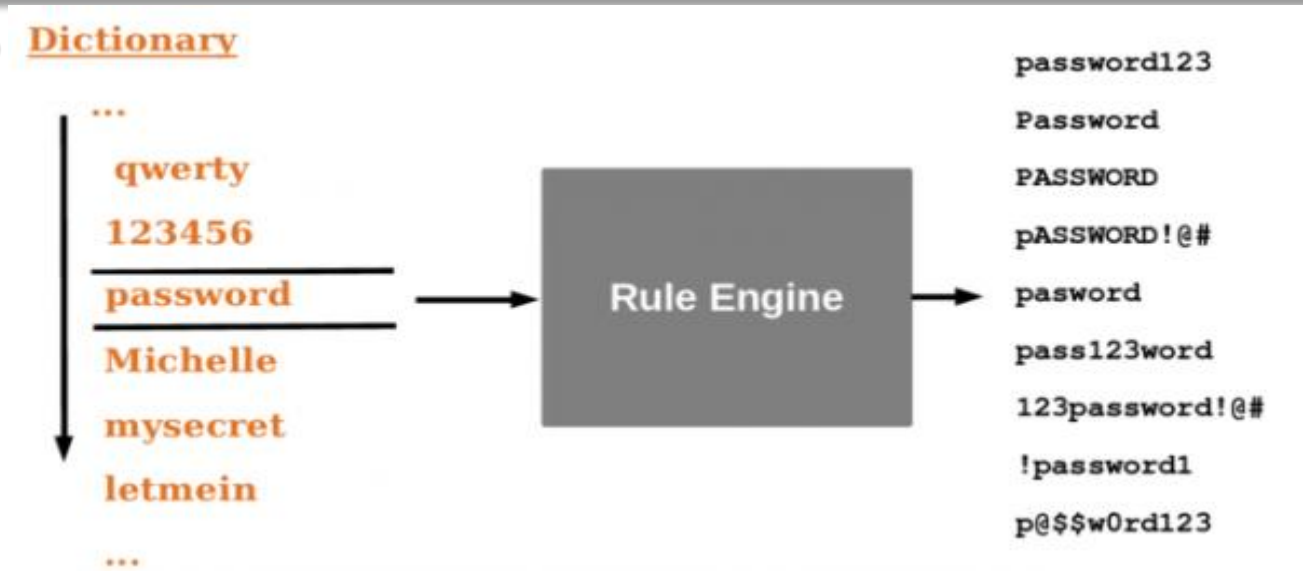
Name	Function	Description	Example Rule	Input Word	Output Word	Note
Nothing	:	Do nothing (passthrough)	:	p@ssW0rd	p@ssW0rd	
Lowercase	l	Lowercase all letters	l	p@ssW0rd	p@ssw0rd	
Uppercase	u	Uppercase all letters	u	p@ssW0rd	P@SSW0RD	
Capitalize	c	Capitalize the first letter and lower the rest	c	p@ssW0rd	P@ssw0rd	
Invert Capitalize	C	Lowercase first found character, uppercase the rest	C	p@ssW0rd	p@SSW0RD	
Toggle Case	t	Toggle the case of all characters in word.	t	p@ssW0rd	P@SSw0RD	
Toggle @	TN	Toggle the case of characters at position N	T3	p@ssW0rd	p@sSW0rd	*
Reverse	r	Reverse the entire word	r	p@ssW0rd	dr0Wss@p	
Duplicate	d	Duplicate entire word	d	p@ssW0rd	p@ssW0rdp@ssW0rd	
Duplicate N	pN	Append duplicated word N times	p2	p@ssW0rd	p@ssW0rdp@ssW0rdp@ssW0rd	
Reflect	f	Duplicate word reversed	f	p@ssW0rd	p@ssW0rddr0Wss@p	
Rotate Left	{	Rotate the word left.	{	p@ssW0rd	@ssW0rdp	
Rotate Right	}	Rotate the word right	}	p@ssW0rd	dp@ssW0r	
Append Character	$X	Append character X to end	$1	p@ssW0rd	p@ssW0rd1	

- The few rules that I used are ***base64.rule***, ***passwordspro.rule***, ***T0XlC.rule*** and in some cases ***d3ad0ne.rule***.

Hashcat: The Art of Word Mangling

- *By using smaller wordlists and rules, it is possible to generate a significant number of password candidates more efficiently.*

- *This word in the wordlist simply needs the right manipulation performed on it in order to produce the correct password candidate.*



- *Using a single rule:*

```
C:\Users\HP\Desktop\Information Systems Security\hashcat-6.2.4>hashcat.exe -a 0 -m 0 example0.hash example.dict -r rules\best64.rule
```

- *Combination attack*

```
C:\Users\HP\Desktop\Information Systems Security\hashcat-6.2.4>hashcat.exe -a 1 -m 0 example0.hash example.dict rockyou.txt --remove
```

- *Hybrid wordlist + mask attack*

```
C:\Users\HP\Desktop\Information Systems Security\hashcat-6.2.4>hashcat.exe -a 6 -m 0 example0.hash example.dict ?l?l?l?l?d?d --remove
```

- *Hybrid mask + wordlist attack*

```
C:\Users\HP\Desktop\Information Systems Security\hashcat-6.2.4>hashcat.exe -a 7 -m 0 example0.hash ?l?l?l?d?d example.dict --remove
```

Hashcat for Windows Forensics

- Security Account Manager (**SAM**) is a **database** file in **Windows XP/Vista/7/8.1/10** that stores users' passwords. This file is located under *C:\Windows\System32\config\.*
- Passwords are, of course, not stored in clear text but rather in "**hashed**" form and for all recent Windows versions, using the **NTLM** proprietary (but known) hashing algorithm.
- Even the hashes are not stored "as is", they are actually found **Double Encrypted** within the SAM Registry Hive, with parts of the encryption keys in the **SYSTEM** Registry Hive.
- **Double Encrypted => DES(AES(NTLMHASH))**
- The SAM file cannot be moved or copied while Windows is running, since the Windows kernel obtains and keeps an exclusive filesystem lock on the SAM file.
- We can retrieve plain NTLM hash using **impacket** which is a collection of Python classes for working with network protocols. Available at https://github.com/SecureAuthCorp/impacket.
- *$ python3 secretsdumpy.py –sam SAM –system SYSTEM local*
- *$ hashcat -a 0 -m 1000 ntlm.txt rockyou.txt -r ./rules/best64.rule*

Document (PDF/DOCX) Password Cracking

- **To extract the plain hash from PDF file:**

$ git clone https://github.com/openwall/john.git
$ cd john
$ cd run
$ python pdf2john.py crackpdf.pdf > hash.txt (Note: remove the pdf name from hash.txt)
$ john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
$ john --show hash.txt

- **To extract the plain hash from Microsoft Office file:**

>> python office2john.py file.docx > wordfile_hash.txt

Available at https://github.com/openwall/john/blob/bleeding-jumbo/run/office2john.py

Note: remove file name from wordfile_hash.txt

>> hashcat.exe -a 0 -m 9600 wordfile_hash.txt rockyou.txt --show