Module 4

WEB APPLICATION TOOLS

Scanning for web vulnerabilities tools

- Only a few kinds of web servers drive the Web's traffic. Apache HTTP Server is the most recognizable in the open source category, while Microsoft's Internet Information Server (IIS) is the most recognizable commercial one.
- The web server is the most obvious component of a web application plat-form; something has to deliver pages to web browsers. But the platform may also comprise data stores, load balancers, and the programming framework used to write pages.
- We can use a web vulnerability scanner to test the basic security of a web application.
- A vulnerability scanner contains a knowledge base of all vulns reported for different components of a web platform.

Nikto

- Nikto is web specific scanner.
- Developed by Chris Sullo and David Lodge.
- It is a Perl-based scanner that searches for known vulnerabilities in common web applications, looks for the presence of common files that have the potential to leak information about an application or its platform, and probes a site for indicators of common misconfigurations.
- The tool focuses on identifying vulns in commercial and open source web application frameworks.
- It won't be as helpful for assessing the security of a custom web application. For example, it may tell that a site uses an outdated (and insecure) version of WordPress, but it won't be able to tell if the blogging application we wrote from scratch is secure or not.

Contd..

Implementation

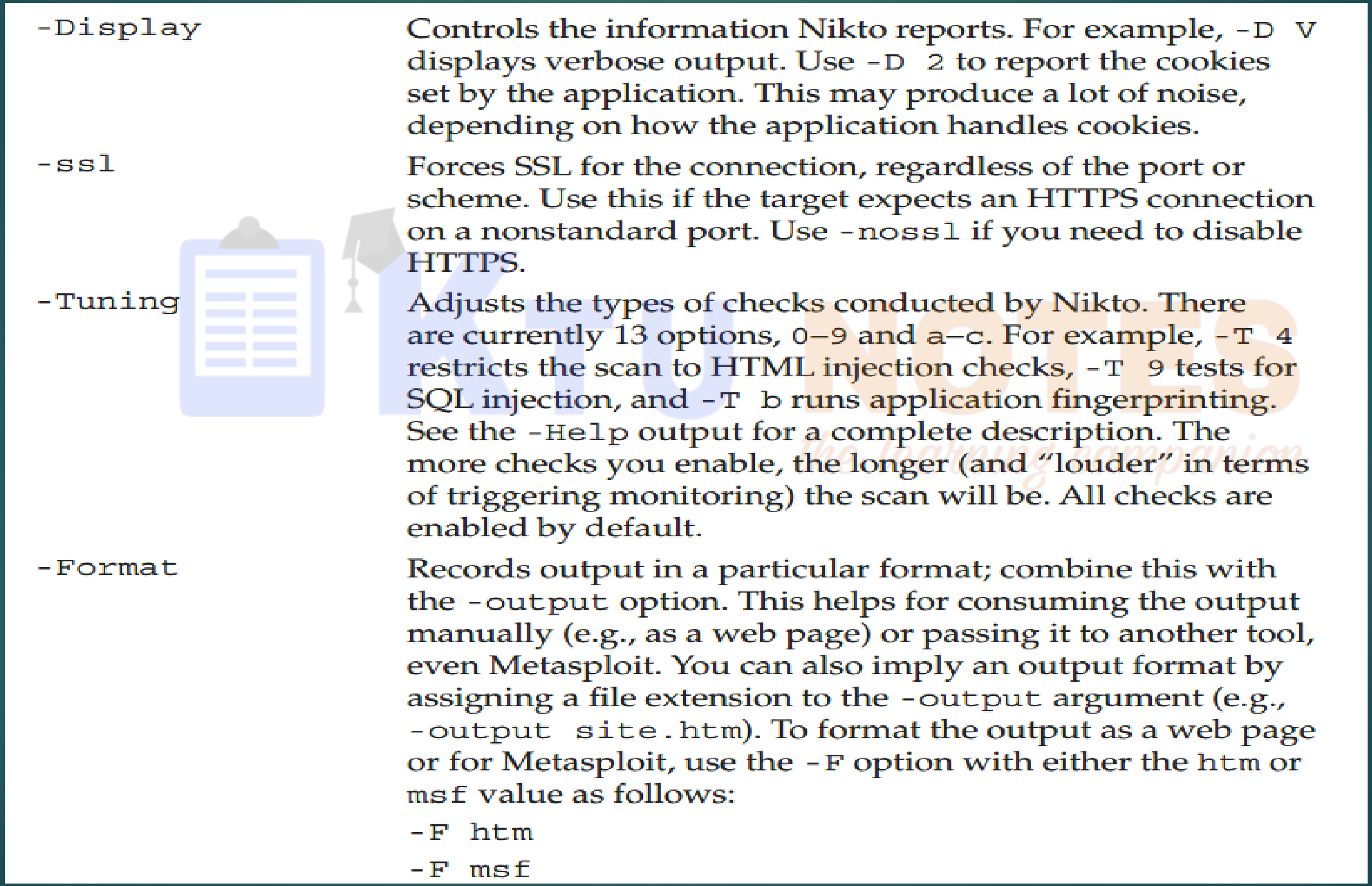
- Nikto is written in Perl, so it will run on any platform that Perl runs on.

Scanning

- Nikto is uncomplicated, but not unsophisticated. Use the -host option to start scanning a single target for the presence of default files, pages that might expose sensitive information, or pages with known vulnerabilities.
- The tool requires a target for running.
- specify the target (-host or –h: Specifies the target. Use a dash (-h -) to take the target name from stdin on the command line. This is useful for typing multiple commands together, such as nmap: nmap -p80 192.168.0.0/24 -oG - | nikto.pl –h-): ,
- specify the port (-p:Specifies an arbitrary port. Take care: specifying port 443 does not imply HTTPS. We must remember to include –ssl.),
- and record the output to a file(-output: Logs output to a file. For example: -output nikto80_website.html –F htm).

Contd..

- Some of the basic options necessary to run Nikto

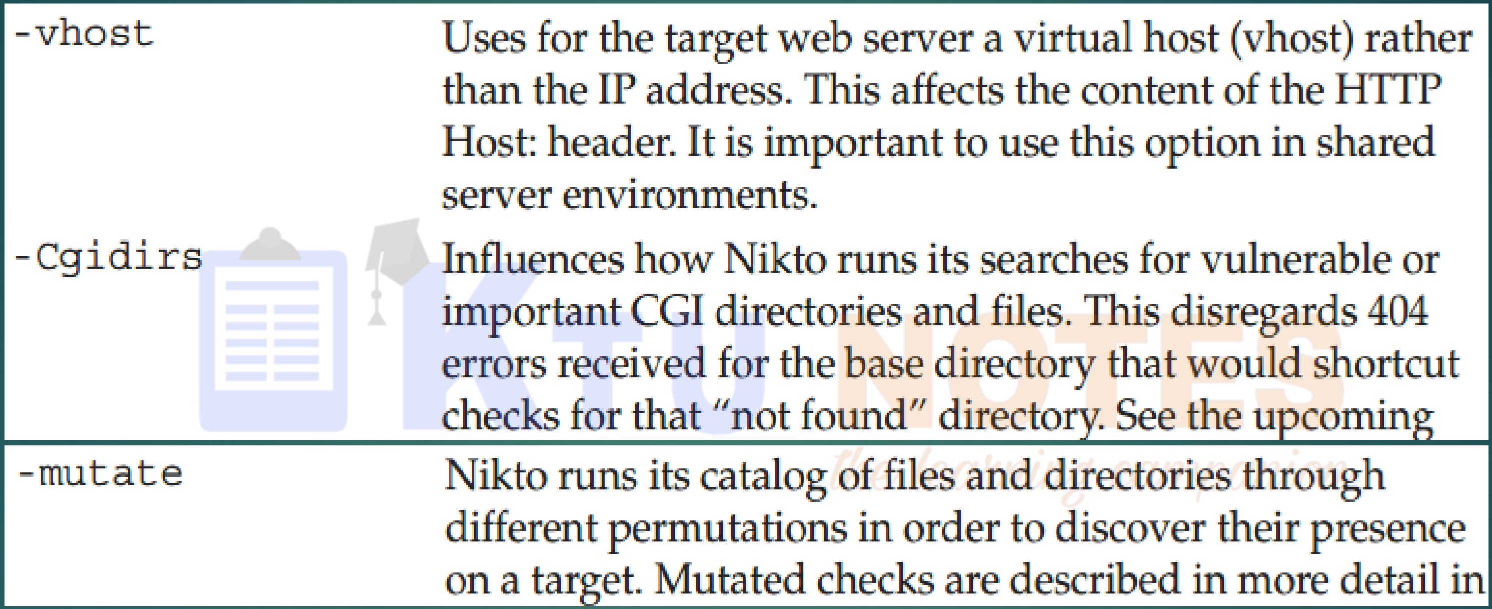


`-Display`	Controls the information Nikto reports. For example, `-D V` displays verbose output. Use `-D 2` to report the cookies set by the application. This may produce a lot of noise, depending on how the application handles cookies.
`-ssl`	Forces SSL for the connection, regardless of the port or scheme. Use this if the target expects an HTTPS connection on a nonstandard port. Use `-nossl` if you need to disable HTTPS.
`-Tuning`	Adjusts the types of checks conducted by Nikto. There are currently 13 options, `0–9` and `a–c`. For example, `-T 4` restricts the scan to HTML injection checks, `-T 9` tests for SQL injection, and `-T b` runs application fingerprinting. See the `-Help` output for a complete description. The more checks you enable, the longer (and "louder" in terms of triggering monitoring) the scan will be. All checks are enabled by default.

`-Format`	Records output in a particular format; combine this with the `-output` option. This helps for consuming the output manually (e.g., as a web page) or passing it to another tool, even Metasploit. You can also imply an output format by assigning a file extension to the `-output` argument (e.g., `-output site.htm`). To format the output as a web page or for Metasploit, use the `-F` option with either the `htm` or `msf` value as follows:
	`-F htm`
	`-F msf`

Contd..

- Some of the basic options necessary to run Nikto



-vhost	Uses for the target web server a virtual host (vhost) rather than the IP address. This affects the content of the HTTP Host: header. It is important to use this option in shared server environments.
-Cgidirs	Influences how Nikto runs its searches for vulnerable or important CGI directories and files. This disregards 404 errors received for the base directory that would shortcut checks for that "not found" directory. See the upcoming
-mutate	Nikto runs its catalog of files and directories through different permutations in order to discover their presence on a target. Mutated checks are described in more detail in

Contd..

Nikto Components

- A line that starts with the # character is ignored. The following example shows some default settings:

```
#CLIOPTS=-g –a
SKIPPORTS=21 111
USERAGENT=Mozilla/5.00 (Nikto/@VERSION) (Evasions:@EVASIONS) (Test:@TESTID)
RFIURL=http://cirt.net/rfiinc.txt?
DEFAULTHTTPVER=1.1
#PROXYHOST=10.1.1.1
#PROXYPORT=8080
#STATIC-COOKIE=cookiename=cookievalue
@@MUTATE=dictionary;subdomains
@@DEFAULT=@@ALL;-@@MUTATE;tests(report:500)
```

- The CLIOPTS setting contains command-line options to include every time Nikto runs. This is useful for shortening the command line if we always wish to include certain options.

Contd..

- The SKIPPORTS setting determines whether Nikto will ignore a target if given one of these ports.
- Modify the USERAGENT setting to spoof the header used by a particular browser. This only spoofs the header; it doesn't affect behavior and browser- fingerprinting that a server may attempt against the client.
- Nikto uses the RFIURL to determine if a web page is vulnerable to remote file inclusion. For example, a page might expect to load HTML from a template stored on its own server and use a URL like http://web.site/index?page=contact.html. Nikto (or a hacker) could try substituting a link for the contact.html page, as in a URL like http://web.site/index?page=http://cirt.net/rfiinc.txt. If the web application retrieves and executes the PHP code from the cirt.net server, then the application is one step away from being completely compromised.

Contd..

- The catch is that every time we run a scan—and every time we find a web site that is vulnerable to an RFI attack—we're signaling its presence in the logs at cirt.net. If we change the link to point to our own page on our own web server, we can check our logs instead.
- Use the PROXY* settings to enable proxy support for Nikto.
- Although there is rarely a need to change the DEFAULTHTTPVER setting, we may find servers that support only version 1.0.
- The @@MUTATE and @@DEFAULT values affect which scan databases Nikto will use to search for vulns against the target. The @@MUTATE settings greatly increase the time it takes to scan a target because they create different combinations of files and directories in order to find vulnerable resources whose location has been slightly altered from its expected default location.
- Nikto uses the files in the database subdirectory to determine what kinds of test it performs and how it categorizes responses from a server. The most important file is the db_dictionary file that contains a manifest of common directories found on web servers.

w3af

- w3af (web application attack and audit framework) is an open-source web application security scanner. The project provides a vulnerability scanner and exploitation tool for Web applications. It provides information about security vulnerabilities for use in penetration testing engagements. The scanner offers a graphical user interface and a command-line interface.
- w3af is divided into two main parts, the core and the plug-ins. The core coordinates the process and provides features that are consumed by the plug-ins, which find the vulnerabilities and exploit them. The plug-ins are connected and share information with each other using a knowledge base.
- Plug-ins can be categorized as Discovery, Audit, Grep, Attack, Output, Mangle, Evasion or Bruteforce.
- w3af was started by Andres Riancho in March 2007
- It is a software that will identify vulnerabilities in web applications by sending specially crafted HTTP requests to it.
- The framework work on all Python supported platforms. It supports mainly LINUX Oses. But it can be installed on Windows OS also.

Contd..

Main plugin types

- The framework has three main plugins types: crawl, audit and attack
- Crawl plugins
  - They have only one responsibility, finding new URLs, forms, and other injection points. A classic example of a discovery plugin is the web spider. This plugin takes a URL as input and returns one or more injection points.When a user enables more than one plugin of this type, they are run in a loop: If plugin A finds a new URL in the first run, the w3af core will send that URL to plugin B. If plugin B then finds a new URL, it will be sent to plugin A. This process will go on until all plugins have run and no more information about the application can be found.
- Audit plugins
  - Take the injection points found by crawl plugins and send specially crafted data to all in order to identify vulnerabilities. A classic example of an audit plugin is one that searches for SQL injection vulnerabilities by sending a'b"c to all injection points.
- Attack plugins
  - Their objective is to exploit vulnerabilities found by audit plugins. They usually return a shell on the remote server, or a dump of remote tables in the case of SQL injection exploits.

Contd..

Other plugins

- Infrastructure
  - Identify information about the target system such as installed WAF (web application firewalls), operating system and HTTP daemon.
- Grep
  - Analyze HTTP requests and responses which are sent by other plugins and identify vulnerabilities. For example, a grep plugin will find a comment in the HTML body that has the word “password” and generate a vulnerability.
- Output
  - The way the framework and plugins communicate with the user. Output plugins save the data to a text, xml or html file. Debugging information is also sent to the output plugins and can be saved for analysis
- Mangle
  - Allow modification of requests and responses based on regular expressions, think “sed (stream editor) for the web”.

Contd..

Other plugins

- Bruteforce
  - Bruteforce logins found during the crawl phase.
- Evasion
  - Evade simple intrusion detection rules by modifying the HTTP traffic generated by other plugins.
- Scan configuration
  - After configuring the crawl and audit plugins, and setting the target URL the user starts the scan and waits for the vulnerabilities to appear in the user interface.
  - Any vulnerabilities which are found during the scan phase are stored in a knowledge base; which is used as the input for the attack plugins. Once the scan finishes the user will be able to execute the attack plugins on the identified vulnerabilities.
  - In most cases it is recommend to run w3af with the following configuration:
    - crawl: web_spider
    - audit: Enable all
    - grep: Enable all

HTTP Utilities

Stunnel

- There are situations in which the client sends out HTTPS connections and cannot be downgraded to HTTP. In these cases, you need a tool that can either decrypt SSL or sit between the client and server and watch traffic in clear text. Stunnel provides this functionality.
- Stunnel is a proxy designed to add TLS (Transport Layer Security) encryption functionality to existing clients and servers without any changes in the programs' code. Its architecture is optimized for security, portability, and scalability (including load-balancing), making it suitable for large deployments.
- Stunnel is a free software authored by Michał Trojnara.
- Stunnel uses the OpenSSL library for cryptography, so it supports whatever cryptographic algorithms are compiled into the library
- SSL communications rely on certificates. The first thing you need is a valid PEM file that contains encryption keys to use for the communications. Stunnel comes with a default file called stunnel.pem, which it lets you define at compile time.

Contd..

- One use of stunnel is to intercept traffic by downgrading client connections from HTTPS to HTTP, inspect or manipulate the traffic, and then upgrade the connection back from HTTP to HTTPS for the server. The concept is similar to using an interactive proxy to be able to view the plaintext form of HTTPS traffic.
- Run stunnel in normal daemon mode (-d). This mode accepts SSL traffic and outputs traffic in clear text. The –f option forces stunnel to remain in the foreground. This is useful for watching connection information and making sure the program is working. Stunnel is not an end-point program.
- In other words, we need to specify a port on which the program listens (-d port) and a host and port to which traffic is forwarded (-r host:port)
- Run stunnel in client mode with the -c option to accept plaintext traffic and forward it over an SSL/TLS connection to a remote (-r) host.
- Stunnel is a robust way to wrap SSL/TLS protection around an otherwise unencrypted service. Use the -l option to specify the full path to a service daemon.

Contd..

- Most services natively support SSL/TLS connections. This is more useful for setting up redirects in order to inspect traffic between a client and server.
- For example: Some clients either don’t provide HTTP proxy settings (otherwise you could use a tool like the Zed Attack Proxy discussed a bit later) or run some protocol other than HTTP over the SSL/TLS connection. In these cases, it’s necessary to use host spoofing tricks and redirection so that you can “downgrade” the client’s connection from SSL/TLS in order to manipulate it, then “upgrade” the connection back to SSL/TLS when sending traffic on to the server.

Password Cracking and Brute-Force Tools

John the Ripper

- John the Ripper remains one of the fastest, most versatile, and most popular password crackers available. It supports password hashing schemes used by many systems, including most Unix-based systems (like OpenBSD and various Linux distributions) and the various Windows hashes, as well as proprietary password hashing functions used by several database and software packages for user account management. John's cracking modes include specialized wordlists, the ability to customize the generation of guesses based on character type and placement (useful when targeting a specific password policy), raw brute force, and statistically guided brute force that uses successfully cracked passwords to influence future guesses.
- John runs on just about any operating system.

Contd..

Implementation

- Obtain and Compile John
- John has hard-coded many compilation flags and optimization settings for dozens of specific operating systems and CPU architectures.
- The following commands would compile John under OS X, Cygwin, and FreeBSD:
  - $ make macosx-x86-64
  - $ make win32-cygwin-x86-sse2
  - $ make freebsd-x86-64
- The make step configures and compiles John for our platform. When this step has finished, the binaries and configuration files will be placed in the ./run directory relative to the ./src directory in which you executed the make command.
- If it has installed correctly, then we can run John.
- Now verify that John works by generating a baseline cracking speed for our

Contd..

Cracking Passwords

- John automatically recognizes common password formats extracted from operating system files like /etc/ shadow or dumped by tools like pwdump. In practice, John supports close to 150 different hashing algorithms
- The following example shows John's ability to guess the correct format for password entries.
  - First, create a text file named windows.txt with the following two lines containing an entry for "Ged" and "Arha." They represent passwords taken from a Windows system.

```
Ged:1006:NO PASSWORD**********************:FB9C381BD729E7A93C14EBAFBA9B78DE:::
Arha:1007:NO PASSWORD**********************:2C5F5597333BD214B5BEA2C01C591BC9:::
```

  - Next, run John against the windows.txt file:

```
$ ./john windows.txt
Warning: detected hash type "nt", but the string is also recognized as "nt2"
Use the "--format=nt2" option to force loading these as that type instead
Loaded 2 password hashes with no different salts (NT MD4 [128/128 X2 SSE2-16])
Tenar (Arha)
```

Contd..

- The brute-force attack should very quickly discover that "Tenar" is the password for the Arha account. It will take much longer to guess the Ged account's password unless we try some refinements to the brute-force approach.
- In the example, John recommended that we use the --format=nt2 option to explicitly define which hash algorithm to target with the cracker. If the format isn't evident, or John misinterprets the format of the target file, use that option to correct it. We can obtain all formats supported by John with the --list option, as follows:

```
$ ./john --list=formats
...
$ ./john --list=format-all-details
```

- To make effective and to reduce time consuming in password cracking using John, expend the resources effectively
- We can try to improve the power of the brute-force attack by optimizing the implementation of algorithms, using faster CPUs, using customized processors, distributing the work, etc. to attain higher cracks per second

Contd..

- We can try to improve the efficiency of the attack by guiding the sequence of guesses or choosing dictionaries that are statistically more likely to match the kinds of passwords humans create.
- One password should have been cracked so far. We use the --show option to list it:

```
$ ./john --show windows.txt
Arha:Tenar:NO PASSWORD*********************:2C5F5597333BD214B5BEA2C01C591BC9:::
1 password hash cracked, 1 left
```

- John keeps track of all passwords it has ever cracked in a john.pot file by default. For example, here's what ours currently looks like:

```
$ cat john.pot
$NT$2c5f5597333bd214b5bea2c01c591bc9:Tenar
```

- Use the --pot option to specify alternate files to store (or read) cracked passwords from.

Contd..

- From an efficiency perspective, we can try different wordlists (aka dictionaries) of common passwords against our unknown hashes. The measure of "common" may be based on past successful cracks, actual dictionaries, or popular terms from media. Use the --wordlist option to try a (relatively) quick pass against the hashes. John provides a single dictionary, password.lst, with its distribution. We can find more, larger dictionaries on the John the Ripper web site.

Incremental Mode Cracking

- John's incremental mode uses "charset" files and john.conf directives to control what kinds of guesses it performs and therefore how many guesses and how long the guesses will take to complete.
- John comes with several predefined incremental modes.
- John's incremental mode tries all eventual permutations of a charset file
- Incremental mode is guaranteed to guess every combination at the expense of taking a very, very long time to complete.

Contd..

- By default, the mode tries all combinations between one and eight characters long.
- If we want to target a specific length, we can edit the john.conf file to add a new incremental mode.
- John builds the charset file with statistical properties from an input file that contains the target characters.

Markov Mode Cracking

- One of John's improvements over time is its adoption of cracking techniques that rely on the statistical composition of cracked passwords to guide the generation of new guesses.
- Its Markov mode tries a limited set of permutations based on a "stats" file.
- Markov mode trades completeness for speed; it tries guesses that are very close to known passwords under the assumption that humans choose passwords based on habit or identifiable patterns.
- Use the --markov option to start this mode against a password file.

Contd..

- Markov mode is most useful when targeting long passwords. For example, trying to brute force a 19-character password composed from a pool of 96 characters is roughly equivalent to brute-forcing a 125-bit encryption algorithm.
- In order to use Markov mode against long passwords, you need to provide the calc_stat command with a source of words of the same size.

Contd..

Pwdump-Grabbing Windows Password Hashes

- The original pwdump program was written by Jeremy Allison in 1997 to demonstrate how to extract password hashes from the Windows Registry.
- Till then there are a number of versions availabale.But they all rely on extracting hashes from the Registry, SAM file, or the lsass.exe process's memory space. The lsass.exe process handles the Local Security Subsystem Service; it's essentially responsible for authentication, which is why its memory contains the system's password hashes.

Pwdump6

- The pwdump tools are simple to use. They require Administrator privileges, so wewill need to start the cmd.exe shell with Run As Administrator.
- The following example demonstrates pwdump6 on a 64-bit Windows system. The -x option is necessary to let pwdump6 know the target system is 64-bit. Otherwise, the process will hang without returning results. The -n option instructs pwdump6 to forego the search for password histories. The output may be passed to John the Ripper in order to start cracking hashes.

Contd..

```
C:\pwdump6\PwDumpRelease> PwDump.exe -n -x localhost
Administrator:500:NO PASSWORD*********************:NO
PASSWORD*********************:::
Arha:1007:NO  PASSWORD*********************:2C5F5597333BD214B5BEA2C01C591BC9:::
Ged:1006:NO PASSWORD*********************:FB9C381BD729E7A93C14EBAFBA9B78DE:::
Guest:501:NO PASSWORD*********************:NO PASSWORD*********************:::
Completed.
```

- Pwdump6 supports remote enumeration provided you have Administrator access to the target's network shares

Pwdump7

- Pwdump7 is hardly any different from pwdump6 in terms of execution. Its command- line options enable us to specify specific source files from which to extract hashes.
- It does not support remote access to a target.

Contd..

THC-Hydra

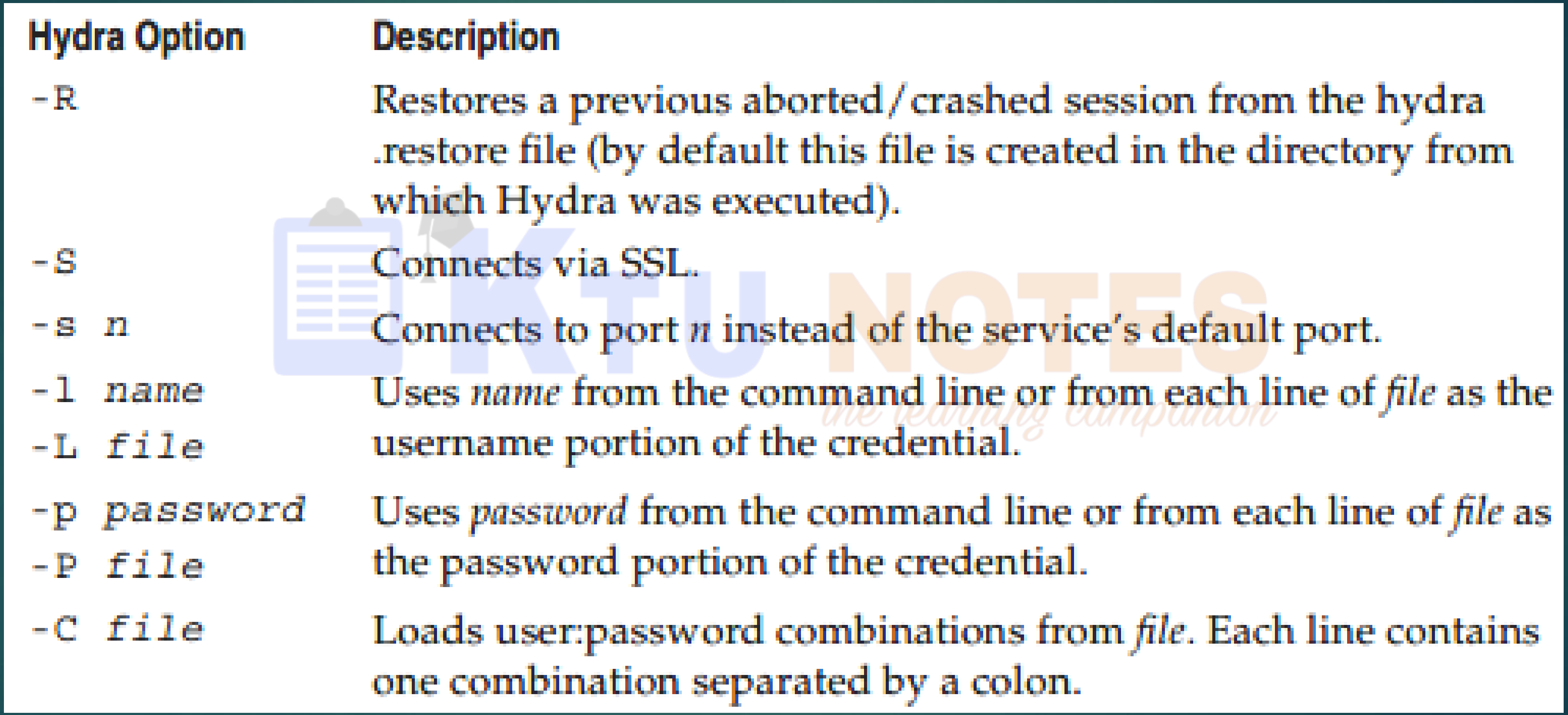
- THC-Hydra (aka simply Hydra) easily surpasses the majority of brute-force tools available on the Internet for two reasons: it is fast, and it targets authentication mechanisms for several dozen protocols.
- When we need to brute force crack a remote authentication service, Hydra is often the tool of choice. It can perform rapid dictionary attacks against more than 50 protocols, including telnet, ftp, http, https, smb, several databases, and much more

Implementation

- Hydra compiles on BSD and Linux systems without a problem; The software can be used under Windows through the Cygwin environment. Follow the usual ./configure, make, make install method for compiling source code.

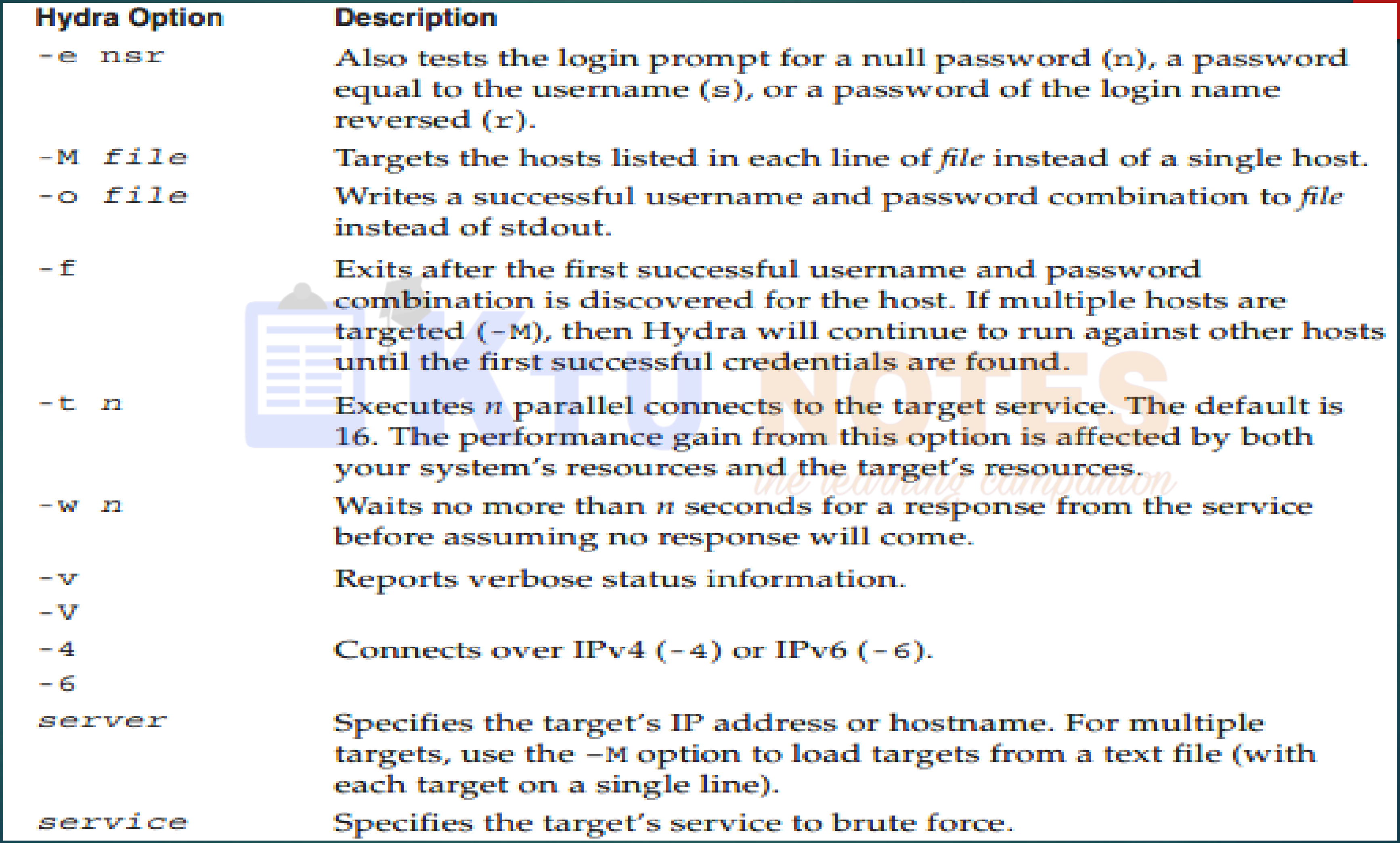
Contd..

- The command-line arguments



Hydra Option	Description
`-R`	Restores a previous aborted/crashed session from the hydra .restore file (by default this file is created in the directory from which Hydra was executed).
`-S`	Connects via SSL.
`-s` *n*	Connects to port *n* instead of the service's default port.
`-l` *name* `-L` *file*	Uses *name* from the command line or from each line of *file* as the username portion of the credential.
`-p` *password* `-P` *file*	Uses *password* from the command line or from each line of *file* as the password portion of the credential.
`-C` *file*	Loads user:password combinations from *file*. Each line contains one combination separated by a colon.

Contd..



Hydra Option	Description
`-e nsr`	Also tests the login prompt for a null password (`n`), a password equal to the username (`s`), or a password of the login name reversed (`r`).
`-M` *`file`*	Targets the hosts listed in each line of *file* instead of a single host.
`-o` *`file`*	Writes a successful username and password combination to *file* instead of stdout.
`-f`	Exits after the first successful username and password combination is discovered for the host. If multiple hosts are targeted (`-M`), then Hydra will continue to run against other hosts until the first successful credentials are found.
`-t` *`n`*	Executes *n* parallel connects to the target service. The default is 16. The performance gain from this option is affected by both your system's resources and the target's resources.
`-w` *`n`*	Waits no more than *n* seconds for a response from the service before assuming no response will come.
`-v` `-V`	Reports verbose status information.

`-4`
`-6`
	Connects over IPv4 (`-4`) or IPv6 (`-6`).

`server`	Specifies the target's IP address or hostname. For multiple targets, use the `–M` option to load targets from a text file (with each target on a single line).
`service`	Specifies the target's service to brute force.