CBTS & OWASP

SSRF Attack Scenarios & Defensive Options

ABOUT US

Ryan Hamrick

- CBTS Principal Security Engineer
- Former Software Developer (ASP.net/C#, Perl, Python)
- Professional Security Practitioner for over 12 years

Nate Fair

- CBTS Information Security Engineer
- Adjunct Professor – Penetration Testing @ UC
- Information Security & Penetration Tester for 5 years

About this talk

Attack

- SSRF101
  - Things to look for
- Types of SSRF
  - Basic/Blind/Mixed
- Abusing SSRF
  - Port Scanning
  - Bypass Firewalls
- Labs
  - Demo

Defend

- Detecting SSRF
  - North/south and East/West
  - Logging
- Defending/Preventing SSRF
  - Input filtering
  - Domain whitelisting
  - Additional Strategies

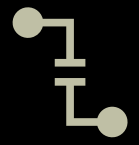
WHAT THIS TALK WON'T COVER

- (AT LEAST ON THE ATTACK SIDE)
- ADVANCED TOPICS LIKE
  - PROTOCOL SMUGGLING
    - SMTP OVER HTTPS (SNI)
    - CR-LF INJECTION
  - EXPLOITING URL PARSERS
    - CURL (LIBCURL), GO (NET/URL), PHP (PARSE_URL), RUBY (ADDRESSABLE), NODEJS (URL)

YOU CAN FIND ALL OF THAT AND MUCH MORE HERE

HTTPS://WWW.BLACKHAT.COM/DOCS/US-17/THURSDAY/US-17-TSAI-A-NEW-ERA-OF-SSRF-EXPLOITING-URL-PARSER-IN-TRENDING-PROGRAMMING-LANGUAGES.PDF

SSRF101



Vulnerability class that encompasses behavior in which a server request is initiated by an attacker

Applications will take a URL from a user perform some action

- setting your avatar via URL, Image/Link preview in chat

To exploit an SSRF vulnerability, an attacker can:

- convince server to make requests on internal resources
- bypass firewall restrictions to uncover new hosts

Things to look for and what you can do

- User supplied URLs
  - Request local/remote files
  - Initiate proxied connections
- PDF Generation
  - Credit card statements, document export functions
- Document Parsers:
  - Inject XML tags
- Link Expansion
  - Image/link preview in chats
- File Uploads
  - Avatar/profile photo from URL
  - Attempt different URI resources

WHAT IT LOOKS LIKE IN CODE

```
//getimage.php

$content = file_get_contents($_GET['url']);

file_put_contents('image.jpg', $content);
```

GET /getimage.php?url=https://website.com/images/cat.jpg

GET /getimage.php?url=http://127.0.0.1/api/v1/getuser/id/1

GET /getimage.php?url=http://169.254.169.254/latest/meta-data/

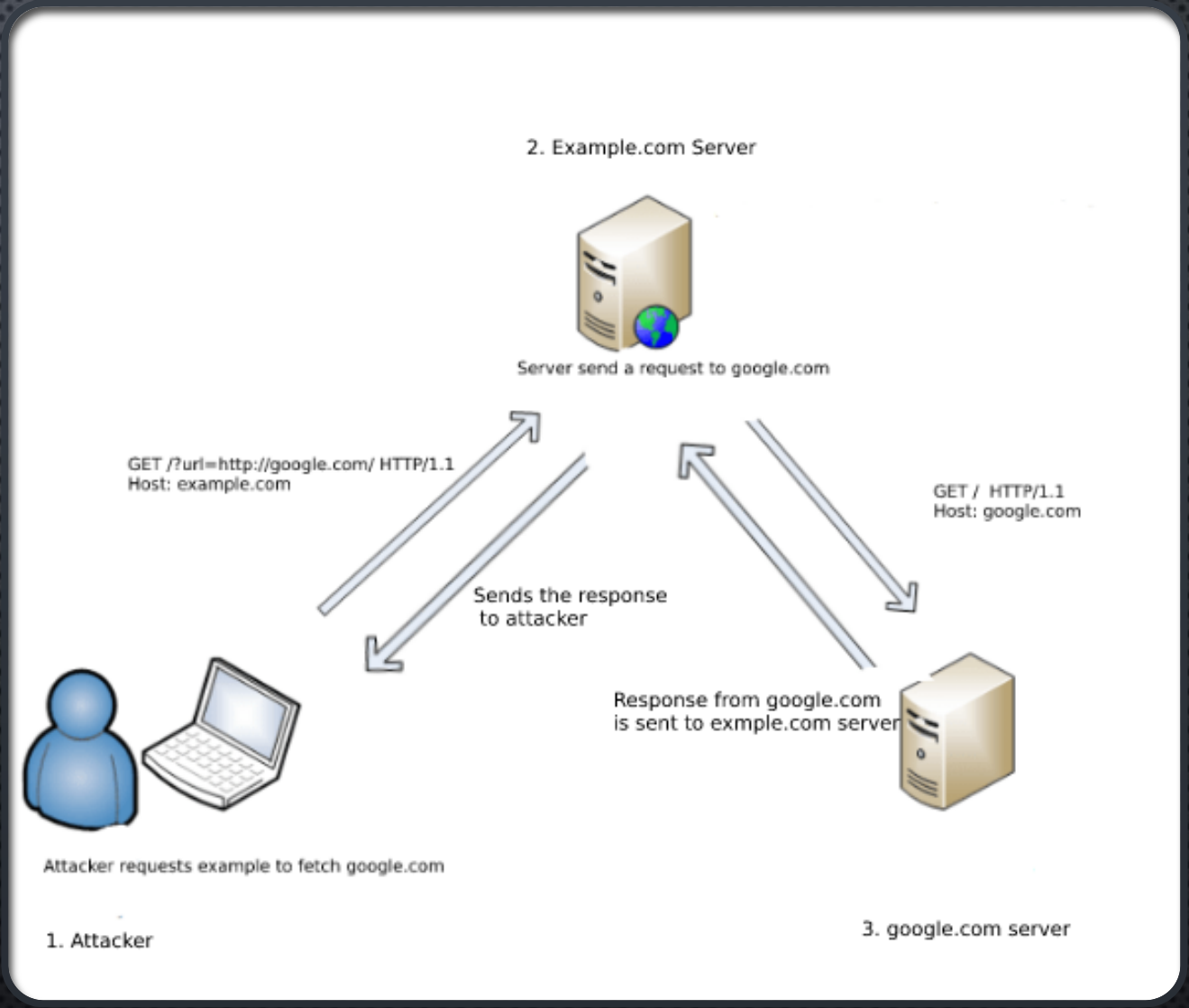
GET /getimage.php?url=file:///etc/passwd

OTHER LESS OBVIOUS THINGS TO LOOK FOR

- URL's embedded in file upload functionality
  - SVG, JPG, XML, JSON
- Hidden API endpoints accepting URL inputs
- HTML tag injection (commonly seen in XSS attacks)

TYPICAL SCENARIO

- NOT JUST HTTP!
- VALID PROTOCOL HANDLERS:
  - FILE:// -- FETCH LOCAL FILES
  - GOPHER:// -- FETCH REMOTE FILES
  - DICT:// -- LOOKUP ENTRIES ACROSS NETWORK
  - TFTP:// -- INSECURE FILE TRANSFER
  - SFTP:// -- SECURE FILE TRANSFER
  - LDAP:// -- INTERACT WITH DIRECTORY SERVICES



Source: https://medium.com/@madrobot/ssrf-server-side-request-forgery-types-and-ways-to-exploit-it-part-1-29d034c27978

TYPES OF SSRF

Basic

- Target application provides a response back to attacker
- Often in the form of HTTP response codes, application errors, other salient behavior
- High degree of confidence vulnerability is present, exploitability likely possible

Blind

- Target application does not provide response back to attacker
- Vulnerability presence is unknown/uncertain, exploitability more difficult
- Often requires more analysis & testing to confirm/deny

Mixed

- Largely application specific
- Time Based – inverse mapping through time/responses variations
- Error Based – “access denied” combined with inverse mapping

Port scanning the internal network

- Inverse mapping/time based/error based
  - Port scanning Intranet Resources
    - Inverse mapping (Time & Error Based), Discover new hosts
  - Bypass typical firewall restrictions/boundaries
    - Uncover hosts unreachable from the web (ie, Non-routeable DMZ hosts a la RFC1918)

Existing Labs

- Extreme Vulnerable Web Application (XVWA)
  - PHP/MySQL
  - https://github.com/s4n7h0/xvwa
- OWASP NodeGoat – Top 10
  - Node JS/MongoDB, Heroku App available
  - https://github.com/OWASP/NodeGoat
- PortSwigger
  - Web Security Academy (Awesome!)
  - https://portswigger.net/web-security/ssrf
- C1
  - https://application.security/
    - Interactive, reconstruct Data Breach

DEMO

- XVWA
- NodeGoat
- PortSwigger Web Security

- C1

Existing Labs

- Extreme Vulnerable Web Application (XVWA)
  - PHP/MySQL
  - https://github.com/s4n7h0/xvwa
- OWASP NodeGoat – Top 10
  - Node JS/MongoDB, Heroku App available
  - https://github.com/OWASP/NodeGoat
- PortSwigger
  - Web Security Academy (Awesome!)
  - https://portswigger.net/web-security/ssrf
- C1
  - https://application.security/
    - Interactive, reconstruct Data Breach

DETECTING SSRF

North/South and East/West

- WAF for North/South
- East/West firewalling is critical for this sort of attack detection
- Position to inspect traffic between web server and back end infrastructure/data sources

Logging!!!

- Centralized logging from WAF, additional firewalls, web server, other infrastructure systems
- Due to the nature of the exploit, there will be many failed requests, watch for scanning type activity
- Ensure that logging levels are correct to capture all the potential events. Debugging not necessary, but INFO level should be collected and reviewed.

DEFENDING/ PREVENTING SSRF

Input Filtering

- Sanitize and filter user input, limiting to known good data inputs
- Potential for regex-style data matching for validation

Domain Whitelisting

- Restrict access to internal resources using a specific whitelist of organizational domains
- Log ALL requests, highlight improper requests and alert

Additional Strategies

- User and group access review and validation, especially important in cloud environments (Capital One)
- Proper error and response handling!!! (Again, Capital One)

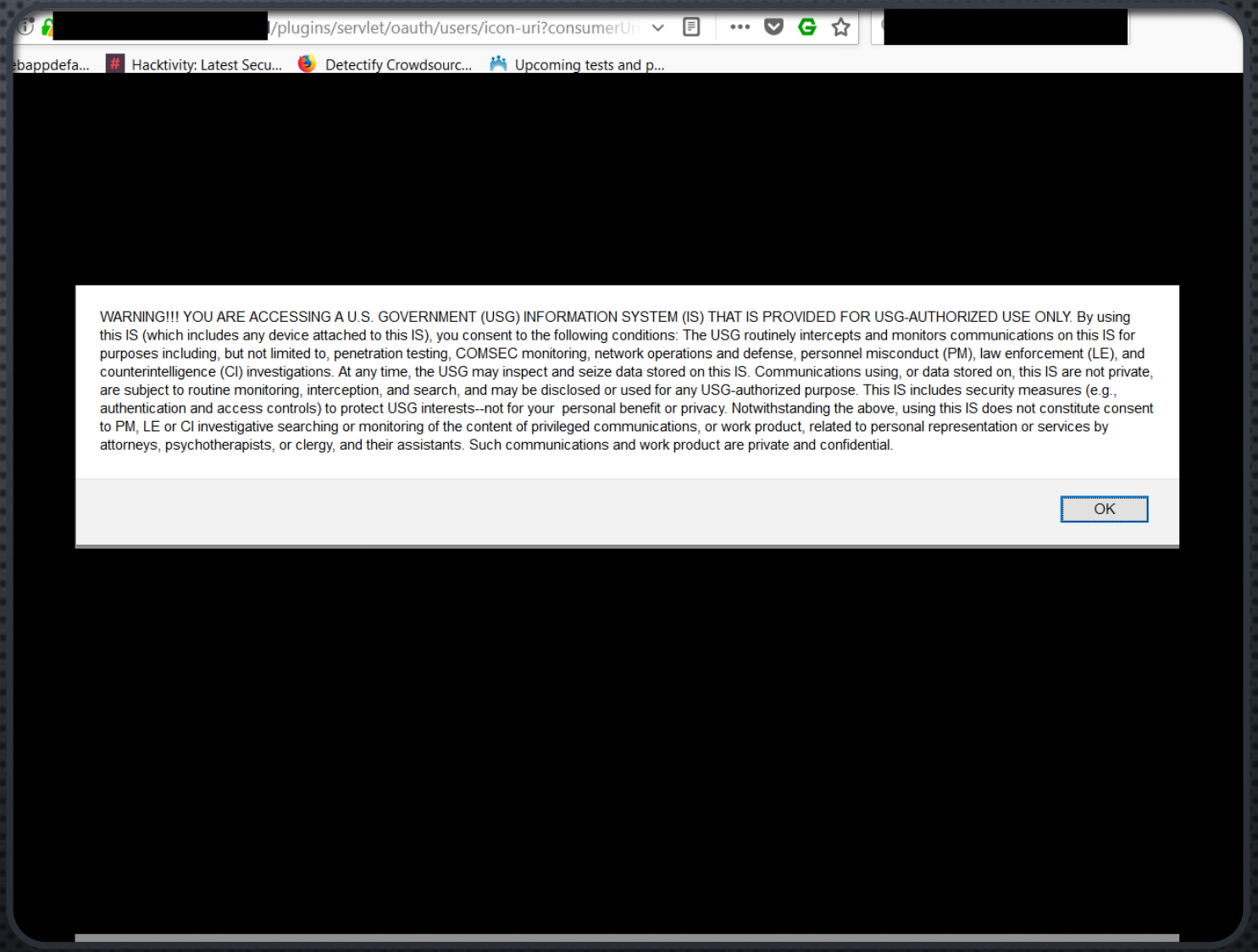
OTHER REAL WORLD EXAMPLES

- TWITTER – LINK EXPANSION

https://twitter.com/BugBountyHQ/status/868242771617792000

REAL WORLD EXAMPLES PT. 2

- ACCESSING NIPRNET THROUGH JIRA



https://medium.com/bugbountywriteup/piercing-the-veil-server-side-request-forgery-to-niprnet-access-c358fd5e249a

SOURCES

- https://portswigger.net/web-security/ssrf
- https://medium.com/swlh/ssrf-in-the-wild-e2c598900434
- https://tactifail.wordpress.com/2019/07/26/three-vulns-for-the-price-of-one/
- https://medium.com/@logicbomb_1/the-journey-of-web-cache-firewall-bypass-to-ssrf-to-aws-credentials-compromise-b250fb40af82
- https://hackerone.com/reports/713
- https://www.hackerone.com/blog-How-To-Server-Side-Request-Forgery-SSRF
- https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit
- https://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability/
- https://dzone.com/articles/the-server-side-request-forgery-vulnerability-and