## 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

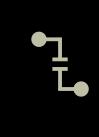
- DETECTING SSRF
  - North/south and East/West
  - LOGGING
- DEFENDING/PREVENTING SSRF
  - INPUT FILTERING
  - DOMAIN WHITELISTING
  - Additional Strategies

## 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

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



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

## SSRF101

# 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/

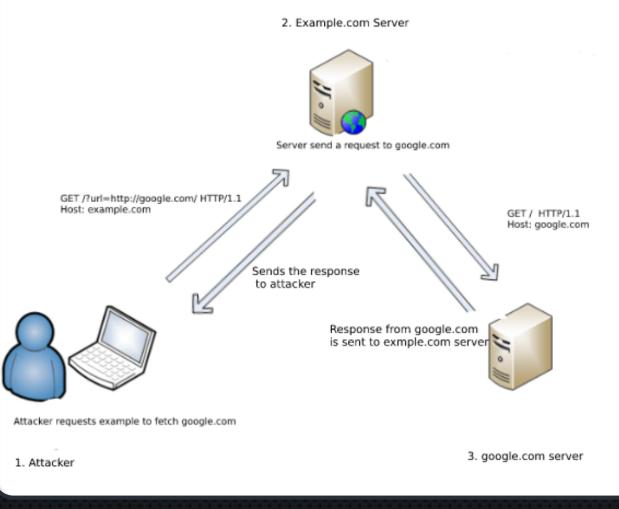
GET /getimage.php?url=file:///etc/passwd

# 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-waysto-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
  - <u>HTTPS://GITHUB.COM/S4N7H0/XVWA</u>
- OWASP NODEGOAT TOP 10
  - NODE JS/MONGODB, HEROKU APP AVAILABLE
  - <u>HTTPS://GITHUB.COM/OWASP/NODEGOAT</u>
- PORTSWIGGER
  - WEB SECURITY ACADEMY (AWESOME!)
  - <u>HTTPS://PORTSWIGGER.NET/WEB-SECURITY/SSRF</u>

#### • C1

- <u>HTTPS://APPLICATION.SECURITY/</u>
  - INTERACTIVE, RECONSTRUCT DATA BREACH

## DEMO

- XVWA
- NODEGOAT
- PORTSWIGGER WEB SECURITY
- C1

## EXISTING LABS

- EXTREME VULNERABLE WEB APPLICATION (XVWA)
  PHP/MySQL
  - <u>HTTPS://GITHUB.COM/S4N7H0/XVWA</u>
- OWASP NODEGOAT TOP 10
  - NODE JS/MONGODB, HEROKU APP AVAILABLE
  - <u>HTTPS://GITHUB.COM/OWASP/NODEGOAT</u>
- PORTSWIGGER
  - WEB SECURITY ACADEMY (AWESOME!)
  - <u>HTTPS://PORTSWIGGER.NET/WEB-SECURITY/SSRF</u>
- C1
  - <u>HTTPS://APPLICATION.SECURITY/</u>
    - 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.

#### 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

DEFENDING/ PREVENTING SSRF

#### 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



tip - Open Graph Protocol is a good case for Blind SSRF / Extract of Meta Data. My POC: SSRF in Twitter via a Tweet :) - \$5,040



7:09 PM · May 26, 2017 · Twitter Web Client

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

#### REAL WORLD EXAMPLES PT. 2

ACCESSING NIPRNET THROUGH JIRA

🕡 🖌 🛛 👘 Vplugins/servlet/oauth/users/icon-uri?consumerUn 🗸 🗐 🚥 💟 G 🏠 🖓

WARNING!!! YOU ARE ACCESSING A U.S. GOVERNMENT (USG) INFORMATION SYSTEM (IS) THAT IS PROVIDED FOR USG-AUTHORIZED USE ONLY. By using this IS (which includes any device attached to this IS), you consent to the following conditions: The USG routinely intercepts and monitors communications on this IS for purposes including, but not limited to, penetration testing, COMSEC monitoring, network operations and defense, personnel misconduct (PM), law enforcement (LE), and counterintelligence (CI) investigations. At any time, the USG may inspect and seize data stored on this IS. Communications using, or data stored on, this IS are not private, are subject to routine monitoring, interception, and search, and may be disclosed or used for any USG-authorized purpose. This IS includes security measures (e.g., authentication and access controls) to protect USG interests--not for your personal benefit or privacy. Notwithstanding the above, using this IS does not constitute consent to PM, LE or Cl investigative searching or monitoring of the content of privileged communications, or work product, related to personal representation or services by attorneys, psychotherapists, or clergy, and their assistants. Such communications and work product are private and confidential.

OK

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

## SOURCES

- <u>HTTPS://PORTSWIGGER.NET/WEB-SECURITY/SSRF</u>
- <u>HTTPS://MEDIUM.COM/SWLH/SSRF-IN-THE-WILD-E2C598900434</u>
- <u>HTTPS://TACTIFAIL.WORDPRESS.COM/2019/07/26/THREE-VULNS-FOR-THE-PRICE-OF-ONE/</u>
- <u>HTTPS://MEDIUM.COM/@LOGICBOMB\_1/THE-JOURNEY-OF-WEB-CACHE-FIREWALL-BYPASS-TO-SSRF-TO-AWS-CREDENTIALS-COMPROMISE-B250FB40AF82</u>
- <u>HTTPS://HACKERONE.COM/REPORTS/713</u>
- <u>HTTPS://WWW.HACKERONE.COM/BLOG-HOW-TO-SERVER-SIDE-REQUEST-FORGERY-SSRF</u>
- <u>HTTPS://DOCS.GOOGLE.COM/DOCUMENT/D/1v1TkWZTRHzRLyOBYXBcDLUEDXGB9NJTNIJXa3u9akHM/edit</u>
- HTTPS://WWW.ACUNETIX.COM/BLOG/ARTICLES/SERVER-SIDE-REQUEST-FORGERY-VULNERABILITY/
- <u>HTTPS://DZONE.COM/ARTICLES/THE-SERVER-SIDE-REQUEST-FORGERY-VULNERABILITY-AND</u>