



Common Pitfalls in Securing Web Applications

Iskandar Setiadi

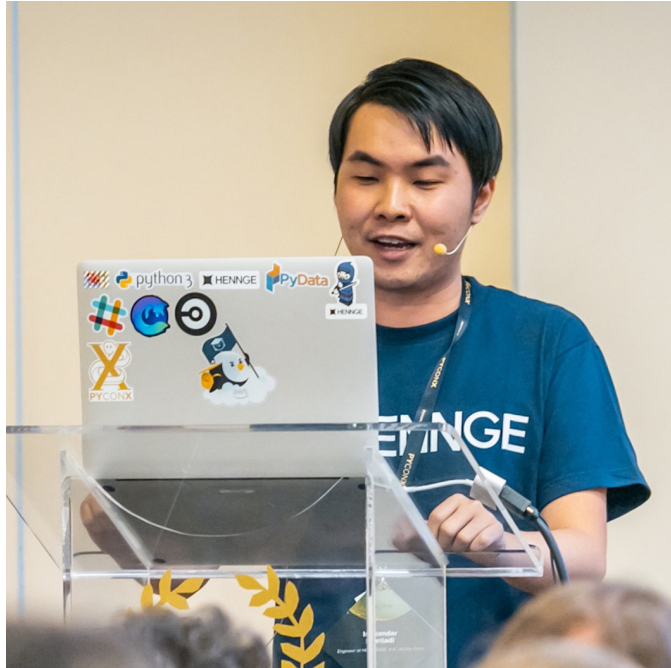
HENNGE, K.K.





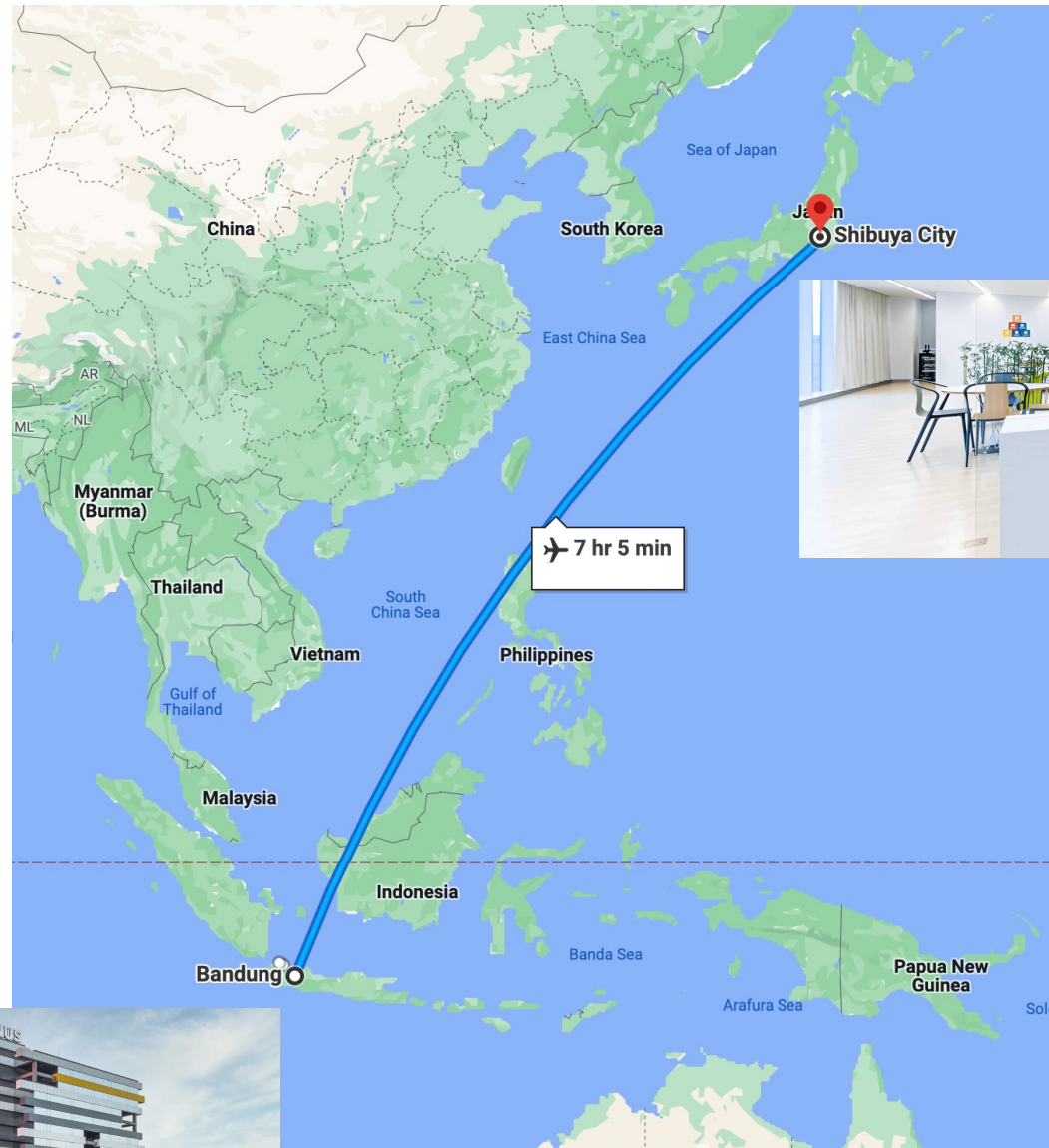
Introduction

Iskandar Setiadi



- From Jakarta, Indonesia
- Graduated from ITB in 2015 (IF' 11)
- HENNGE, K.K. based in Tokyo, Japan
- Software Engineer → Division Manager

- PyCon ID
Speaker: 2017, 2019, **2023**
- Other PyCons (as Speaker)
Japan (+ Reviewer, Sprint sponsor)
Malaysia
Hong Kong
Italy



Why PyCons?





Security Incidents

Indonesia

In Indonesia, there's much less transparency on cyber security compared to other countries. Is it caused by system error? Or human error (e.g.: social engineering)?

- 31 million Indonesian passport data leakage by “Bjorka” (2022) → no post-mortem analysis
- 91 million users data leakage on Tokopedia platform (2020) → no post-mortem analysis
- 13 million users data leakage on Bukalapak platform (2015-2017) → no post-mortem analysis
- Security vulnerabilities at Gojek platform (2016) which leads to users data leakage because of **unchecked Authorization header** → Post-mortem: <https://blog.gojek.io/security-vulnerabilities-lifecycle-at-gojek/>

World

In the United States, transparency is prioritized and we can learn easily from past incidents.

- 5.4 million Twitter (X) account handles and phone numbers were leaked because of **design flaw**: Submitting an email address will return associated phone number tied to the account → <https://privacy.twitter.com/en/blog/2022/an-issue-affecting-some-anonymous-accounts>
- 140+ million personal data leakage on Equifax platform (2017) because of **unpatched vulnerability CVE-2017-5638 on Apache Struts2** → <https://investor.equifax.com/news-events/press-releases/detail/237/equifax-releases-details-on-cybersecurity-incident>
- More lists: <https://haveibeenpwned.com/PwnedWebsites>

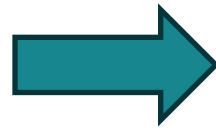
“Security is only as strong as the weakest link”



Weakest Link

From the examples on previous slides:

- Design flaw
- Unchecked authorization header
- Unpatched known vulnerability (CVE)



A lot of security incidents are caused by a pretty straightforward trick.

Is your system safe?

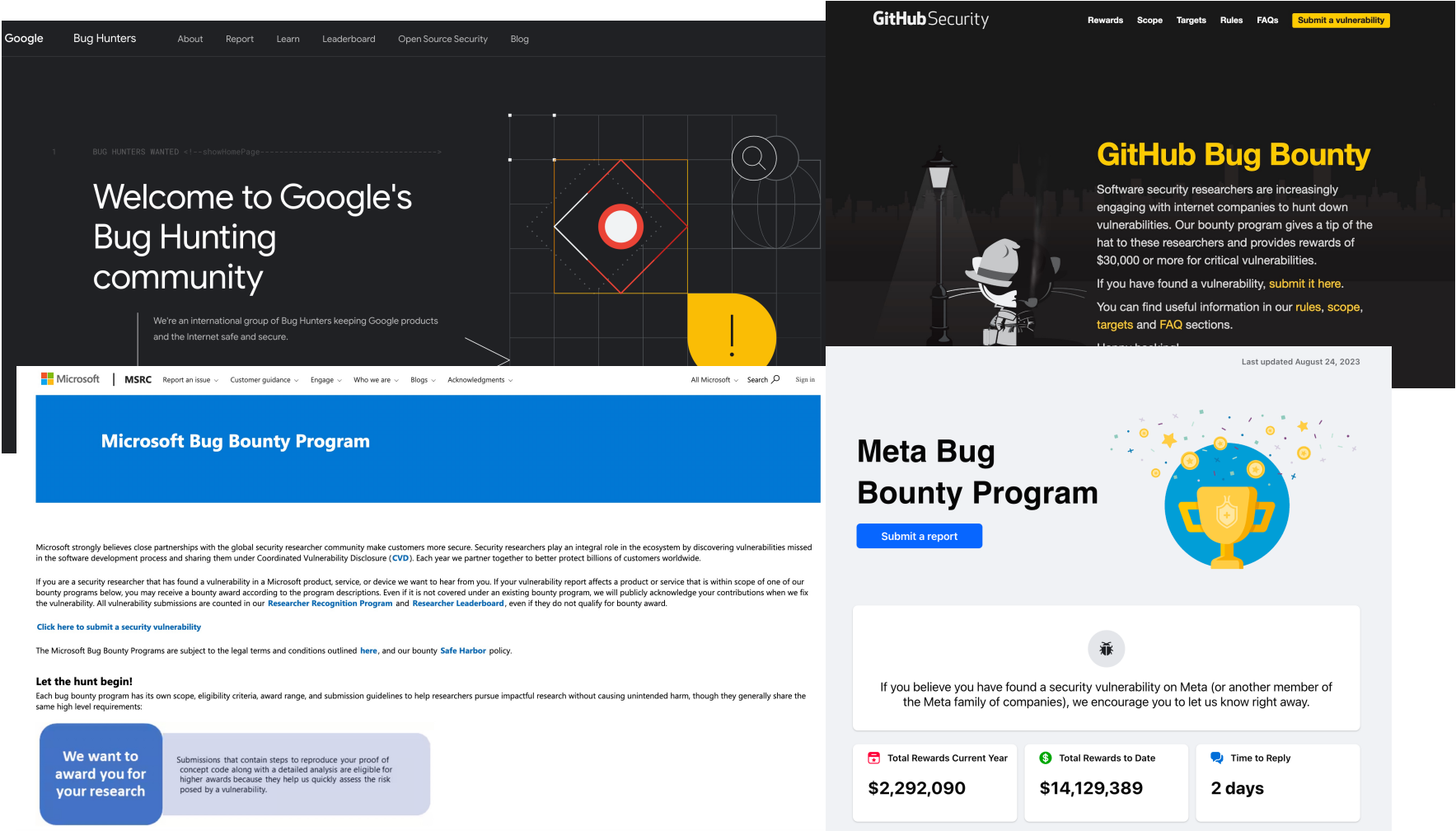
“No”

Does your system have...

- monitoring for suspicious activities?
- regular security checks?
- real-time up-to-date dependencies?
- attacker who tried to get in?

The absence of evidence
is not the evidence of absence

Bug Bounty Programs





Common Pitfalls

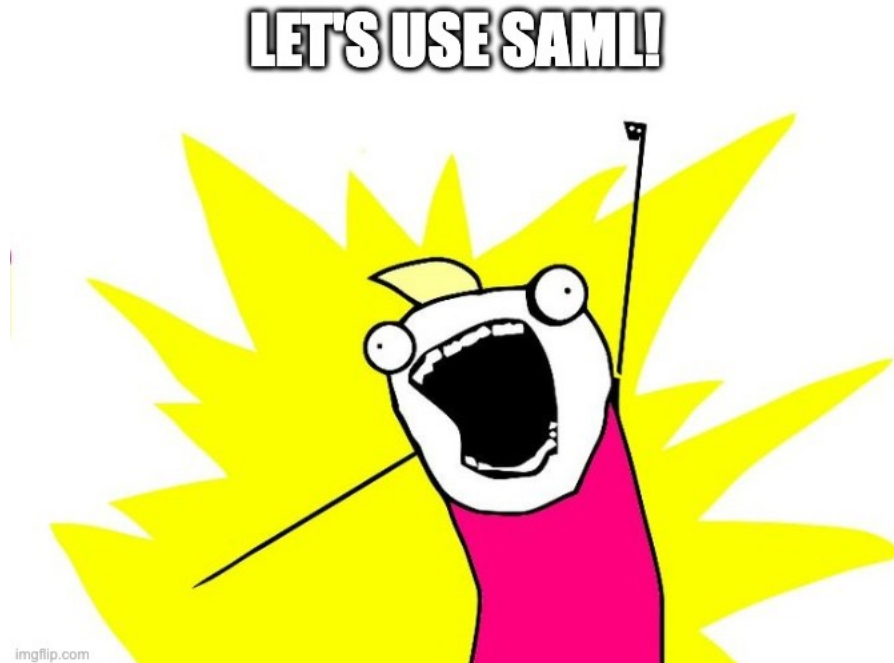
Let's build a security wall!

```
def idp_initiated(request):
    saml_client = app.saml_client
    authn_response = saml_client.parse_authn_request_response(
        request.form.get('SAMLResponse'),
        entity.BINDING_HTTP_POST)
    if authn_response is None:
        return text('Not authorized')
    authn_response.get_identity()
    user_info = authn_response.get_subject()
    username = user_info.text

    if request.form.get('RelayState'):
        response = redirect(request.form.get('RelayState'))
    else:
        response = redirect('/')

    response.cookies['user_id'] = username
    response.cookies['user_id']['httponly'] = True
    if not app.config_object.is_local:
        response.cookies['user_id']['secure'] = True

    return response
```



Wait a minute...

I have followed the framework documentation to read & write cookies ✓

I have set "httponly" and "secure" attributes ✓

It should be safe, right?

Name	Value
user_id	iskandar.setiadi@hennge.com

← → ↺ 🏠 📄 sanic.readthedocs.io/en/v19.12.0/sanic/cookies.html

Logging
Request Data
Response

☐ Cookies

- Reading cookies
- Writing cookies
- Deleting cookies

Routing
Blueprints
Static Files
Versioning
Exceptions
Middleware And Listeners
WebSocket
Handler Decorators
Streaming
Class-Based Views
Custom Protocols
Sockets
SSL Example
Debug Mode
Testing
Deploying
Extensions
Examples
Changelog
Contributing
API Reference

Cookies

Cookies are pieces of data which persist inside a user's browser. Sanic can both read and write cookies, which are stored as key-value pairs.

ⓘ Warning

Cookies can be freely altered by the client. Therefore you cannot just store data such as login information in cookies as-is, as they can be freely altered by the client. To ensure data you store in cookies is not forged or tampered with by the client, use something like [itsdangerous](#) to cryptographically sign the data.

Reading cookies

A user's cookies can be accessed via the `Request` object's `cookies` dictionary.

```
from sanic.response import text

@app.route("/cookie")
async def test(request):
    test_cookie = request.cookies.get('test')
    return text("Test cookie set to: {}".format(test_cookie))
```

Writing cookies

When returning a response, cookies can be set on the `Response` object.

```
from sanic.response import text

@app.route("/cookie")
async def test(request):
    response = text("There's a cookie up in this response")
    response.cookies['test'] = 'It worked!'
    response.cookies['test']['domain'] = '.gotta-go-fast.com'
    response.cookies['test']['httponly'] = True
    return response
```

← But we rarely read this part

← We often immediately read this part

Fortunately, I store users' info in Redis.
I'm safe!

Let's analyze...

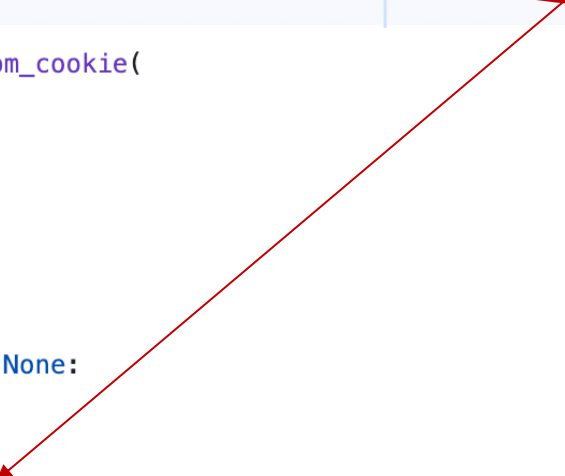
Name	Value
verified_email	71eb9e22-3325-482b-ae8a-b8ef477ea241

```
async def get_valid_email_address_from_cookie(
    link: models.Link,
    cookie_id: Optional[str] = None,
) -> Optional[str]:
    if cookie_id is None:
        return None

    if link.valid_email_addresses is None:
        return None

    email_addresses = await reader_aredis_client.smembers(
        f"verified_email_{cookie_id}_{link.domain_id}"
    )
    for email_address in email_addresses:
        email_address = email_address.decode()
        for pattern in link.valid_email_addresses:
            if check_valid_email(pattern, email_address):
                return email_address

    return None
```



No one can bruteforce UUID! Unless ...

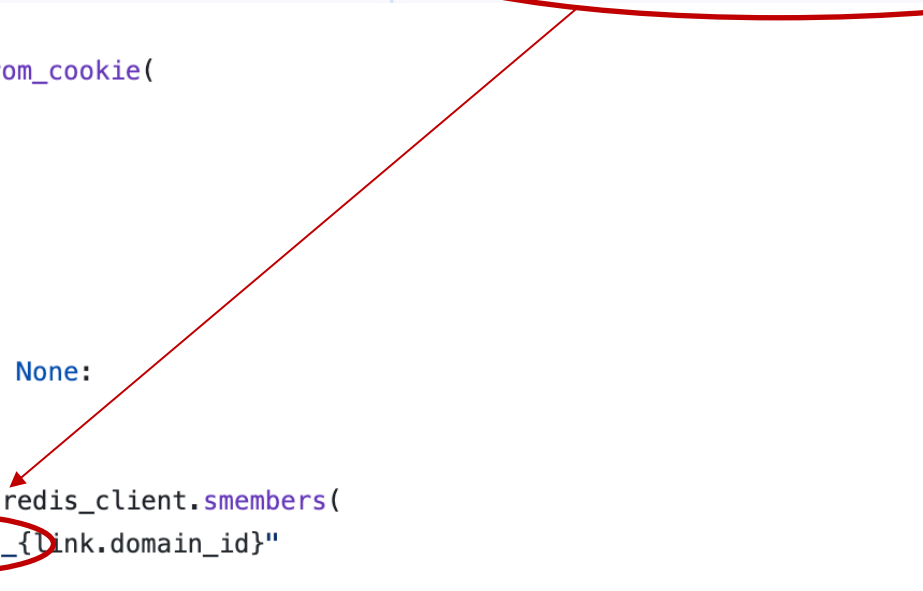
Name	Value
verified_email	*

```
async def get_valid_email_address_from_cookie(
    link: models.Link,
    cookie_id: Optional[str] = None,
) -> Optional[str]:
    if cookie_id is None:
        return None

    if link.valid_email_addresses is None:
        return None

    email_addresses = await reader_aredis_client.smembers(
        f"verified_email_{cookie_id}_{link.domain_id}"
    )
    for email_address in email_addresses:
        email_address = email_address.decode()
        for pattern in link.valid_email_addresses:
            if check_valid_email(pattern, email_address):
                return email_address

    return None
```



Let's sanitize cookie value and store it as JWT.
I'm safe!

Common Mistakes

pyjwt.readthedocs.io/en/latest/

PyJWT

latest

Search docs

Installation

Usage Examples

Frequently Asked Questions

Digital Signature Algorithms

API Reference

Changelog

Installation

You can install `pyjwt` with `pip`:

```
$ pip install pyjwt
```

See [Installation](#) for more information.

Example Usage

```
>>> import jwt
>>> encoded_jwt = jwt.encode({"some": "payload"}, "secret", algorithm="HS256")
>>> print(encoded_jwt)
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzb211IjoicGF5bG9hZCJ9.4twFt5NiznN84AWoo1d7K01T_yoc0Z6X0
>>> jwt.decode(encoded_jwt, "secret", algorithms=["HS256"])
{'some': 'payload'}
```

Common Mistakes

- Weak signing key such as “secret”
 - For public Github projects, common issue is pushing this secret to the codebase
- JWT is decoded but signature is not validated (PyJWT enforces validation by default but there are a few libraries out there which don't follow proper standards)
- Payload contains private / sensitive information
- JWT is issued but other cookies are issued for simplifying implementations and these other cookies are not signed

Examples

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Imlza2FuZGFyLnNldGhZGlAaGVubmdlLmNvbSIsInN5c3R1bV9zZW50ZXQ1OjE2OTk3OTM1NTV9.Xi3RZgvURLBmef2DEs_p0U0toCU-1Hlpah8gBrWcBYU
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "email": "iskandar.setiadi@hennge.com",
  "system_secret": "secret",
  "iat": 1699793555
}
```

Name	Value
user_id	iskandar.setiadi@hennge.com
session	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Imlza2FuZGFyLnNldGhZGlAaGVubmdlLmNvbSIsInN5c3R1bV9zZW50ZXQ1OjE2OTk3OTM1NTV9.Xi3RZgvURLBmef2DEs_p0U0toCU-1Hlpah8gBrWcBYU

That's all, right?
I'm safe...?

So far, we only discussed about the session cookie (/ in local storage) issuance part.
What's about the process in-between?

SAML assertion signature is not validated

A single line is all that matters.

```
try:
    saml_response = response_factory(xmlstr,
                                     client.config,
                                     service_url,
                                     outstanding_queries_cache,
                                     timeslack=300,
                                     decode=False,
                                     asynchop=True,
                                     allow_unsolicited=True)
```

```
0/3
674 try:
675     saml_response = response_factory(xmlstr,
676                                     client.config,
677                                     service_url,
678                                     outstanding_queries_cache,
679                                     timeslack=300,
680                                     decode=False,
681                                     asynchop=True,
682 + want_assertions_signed=True,
683                                     allow_unsolicited=True)
```

SAML is hard, don't reinvent the wheel when it's possible

Disclosure Github

I reported this to Github security via Hackerone on 16th of January. It was acknowledged 31st with GHE version 2.8.7

Timeline

- 2017-01-10: Incorrect XML Signature validation vulnerability discovered and reported
- 2017-01-10: Report acknowledged
- 2017-01-11: Report triaged
- 2017-01-12: Mitigation released with v. 2.8.6 and bounty awarded
- 2017-01-16: XSW vulnerability discovered and reported
- 2017-01-16: Report acknowledged and triaged
- 2017-01-27: Asked for update on mitigation/release
- 2017-01-31: Mitigation released with v. 2.8.7 and bounty awarded

Full SAML Implementation Assessment

Following the above reports I received a research grant in order to continue looking into (and extend that the agreed timeframe and my off-work availability allowed) security audit suggestions/recommendations about the implementation in order to minimize the p

Outro

I enjoyed finding and writing these so I hope if you made it through to the end, you did too. Working with the Github Security guys was a bliss and I can verify first hand that their approach towards their bounty program is as serious and as cool as they describe it on their recent [blog post](#)

SAML authentication XSW

On Breaking SAML: Be Whoever You Want to Be

Juraj Somorovsky¹, Andreas Mayer², Jörg Schwenk¹, Marco Kampmann¹, and Meiko Jensen¹

¹Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany

²Adolf Würth GmbH & Co. KG, Künzelsau-Gaisbach, Germany

{Juraj.Somorovsky, Joerg.Schwenk, Marco.Kampmann, Meiko.Jensen}@rub.de,
Andreas.Mayer@wuertth.com

Abstract

The Security Assertion Markup Language (SAML) is a widely adopted language for making security statements about subjects. It is a critical component for the development of federated identity deployments and Single Sign-On scenarios. In order to protect integrity and authenticity of the exchanged SAML assertions, the XML Signature standard is applied. However, the signature verification algorithm is much more complex than in traditional signature formats like PKCS#7. The integrity protection can thus be successfully circumvented by application of different XML Signature specific attacks, under a weak adversarial model.

In this paper we describe an in-depth analysis of 14 major SAML frameworks and show that 11 of them, including Salesforce, Shibboleth, and IBM XSW, have critical XML Signature wrapping (XSW) vulnerabilities. Based on our analysis, we developed an automated penetration testing tool for XSW in SAML frameworks. Its feasibility was proven by additional discovery of a new XSW variant. We propose the first framework to analyze such attacks, which is based on the information flow between two components of the Relying Party. Surprisingly, this analysis also yields efficient and practical countermeasures.

1 Introduction

The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security information

and, this shall be achieved by using XML Signatures, which should either cover the complete SAML assertion, or an XML document containing it (e.g. a SAML Authentication response).

However, roughly 80% of the SAML frameworks that we evaluated could be broken by circumventing integrity protection with novel XML Signature wrapping (XSW) attacks. This surprising result is mainly due to two facts:

- **Complex Signing Algorithm:** Previous digital signature data formats like PKCS#7 and OpenPGP compute a single hash of the whole document, and signatures are simply appended to the document. The XML Signature standard is much more complex. Especially, the position of the signature and the signed content is variable. Therefore, many permutations of the same XML document exist.
- **Unspecified internal interface:** Most SAML frameworks treat the Relying Party (i.e. the Web Service or website consuming SAML assertions) as a single block, assuming a joint common state for all tasks. However, logically this block must be subdivided into the signature verification module (later called RP_{sig}) which performs a cryptographic operation, and the SAML processing module (later called RP_{claim}) which processes the claims contained in the SAML assertion. Both modules have different views on the assertion, and they typically only exchange a Boolean value about the validity of the signature.

Paper that explains a lot of attack vectors

ig manually for a couple of the aforementioned companies reveals that

- telefonika
- Caltex Australia
- Georgia State University
- Japan Airlines
- Santa Clara County
- City of Chicago, IL
- British Airways

others have their domains set as federated and thus were vulnerable. Additionally, some random checks against high profile targets showed organizations across various sectors use federated SSO with their Office365 subscriptions. Some prominent examples are in the list below

- Microsoft (well, duh)
- Vodafone
- British Telecommunications plc
- Verizon
- International Monetary Fund
- Royal Dutch Shell
- The Daily Mail
- Novartis Pharma AG
- Pfizer
- Toyota Motor North America
- Cisco
- IBM
- Intel
- Pricewaterhouse Coopers (PwC)
- KPMG
- Scania AB
- Vinci

retty easy to automate this and check against company domain name lists to identify potential targets, but we did not have the time nor the ion to do so.

ro

The aforementioned issue fell within the scope of the Online Service bug bounty program and as such has been rewarded and acknowledged by Microsoft on

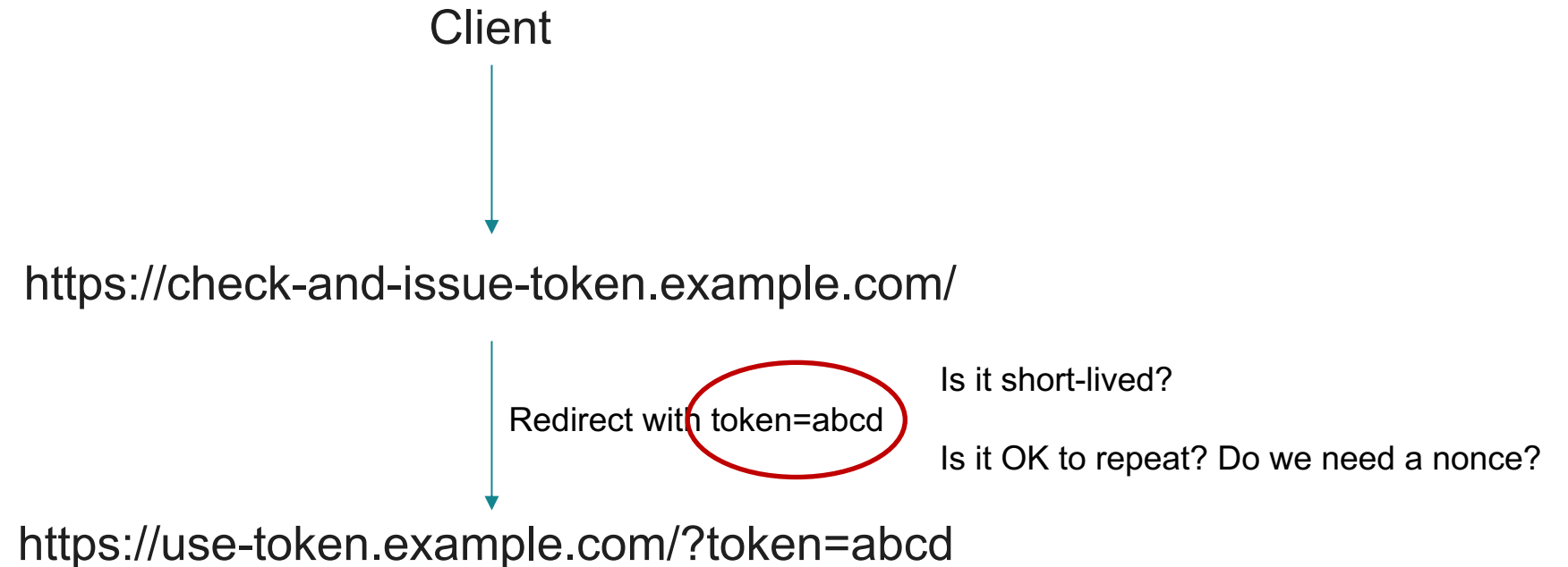
- <https://technet.microsoft.com/en-us/security/dn469163>
- <https://technet.microsoft.com/en-us/security/cc308589.aspx>

Microsoft

Developers: “Let’s make our system as generic as possible by supporting A, B, C, etc!”



Replay attack in microservices



OK, I understand common pitfalls in AuthN/Z process. What's next?

Broken Access Control (Object Level Authorization) → #1 in OWASP

- Validation is only implemented on the front-end component

```
function EmployeeExportCsvController($rootScope, $scope, $uibModal, $location) {  
  
    if ($rootScope.user.admin == true) {  
        // Code to export all users to CSV file here  
    } else {  
        $location.path('/');  
    }  
};
```


Broken Access Control (Object Level Authorization) → #1 in OWASP

- User role is not validated properly
 - User can access admin's route
 - **Unauthenticated user can access endpoints that should be protected →**

Vulnerability #1 Ride History Data Leak

You can get a list of all rides taken of any user using this API endpoint including the exact GPS co-ordinates. The Authorization token is present but is not being used for validation.

<https://api.gojekapi.com/gojek/v2/customer/v2/history/551925748>

```
Work manish $ curl -H 'Host:api.gojekapi.com' -H 'Accept:/*/*' -H 'X-Location:' -H 'Authorization:Bearer [REDACTED]' -H 'X-User-Locale:en-ID' -H 'Accept-Encoding:gzip;q=1.0, compress;q=0.5' -H 'Accept-Language:en-ID' -H 'X-AppId:com.go-jek.ios' -H 'User-Agent:GO-JEK/2.17.1 (com.go-jek.ios; build:2.17.2.0; iOS 10.0.2) Alamofire/3.5.1' -H 'Connection:keep-alive' -H 'X-UniqueId:[REDACTED]' -H 'X-AppVersion:2.17.1' 'https://api.gojekapi.com/gojek/v2/customer/v2/history/551925748?limit=20' | gunzip | jq '.'
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
100	1095	0	1095	0	0	177	0
						0:00:06	0:00:06

```
[
  {
    "name": "Jalan Durian Raya, Rawa Buaya, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta, Indonesia, Jalan Durian Raya, Rawa Buaya, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta, Indonesia",
    "latLong": "-6.1636313,106.7379723",
    "note": "",
    "address": "Jalan Durian Raya, Rawa Buaya, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta, Indonesia, Jalan Durian Raya, Rawa Buaya, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta, Indonesia",
    "distance": 0
  },
  {
    "name": "Jalan Haji Cokong, Kecamatan Setiabudi, Kota Jakarta Selatan",
    "latLong": "-6.216352800000001,106.8321648",
    "note": "",
    "address": "Jalan Haji Cokong, Kecamatan Setiabudi, Kota Jakarta Selatan",
    "distance": 0
  },
  {
    "name": "Jl. H. Cokong, Karet, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta, Indonesia, Jl. H. Cokong, Karet, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta, Indonesia",
    "latLong": "-6.2147895,106.8328642",
    "note": "",
    "address": "Jl. H. Cokong, Karet, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta, Indonesia, Jl. H. Cokong, Karet, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta, Indonesia"
  }
]
```

Broken Access Control (Object Level Authorization) → #1 in OWASP

- Multi-tenancy is not implemented properly
 - User can access resources from other tenants (*a.com* user can access *b.com* resources)
- Resource object is not tied to a user properly

Transaction ID: 1234
User: A
Tenant: example.com

Transaction ID: 2345
User: B
Tenant: example.com

Allowed
Not allowed

: /api/example.com/transaction/1234
: /api/example.com/transaction/2345 → not checked

Malicious user input

Image upload vulnerability (ImageMagick) affecting Bukalapak, Tokopedia, and many other vendors in 2016

Write-up by Herdian Nugraha →

<https://id.techinasia.com/talk/bagaimana-saya-hack-situs-besar-indonesia-dengan-gambar>
(CVE-2016-3714)

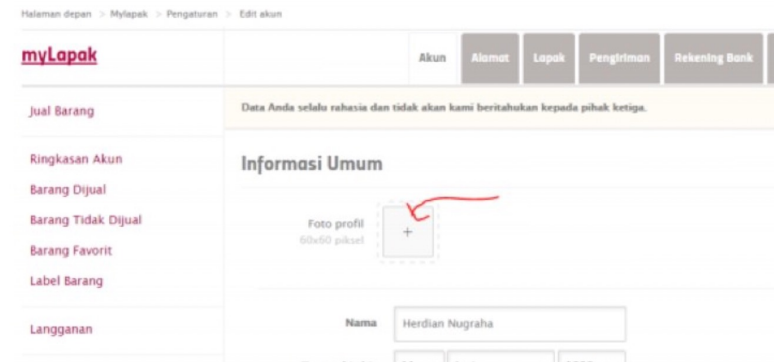
Lesson learned:

- Never trust user's input
- Keep up-to-date with zero-day vulnerabilities

Bagaimana Saya Meretas Bukalapak, Tokopedia, dan Sribu dengan Gambar

Pada suatu hari, ada seorang pemuda yang sedang *browsing* sana-sini mencari barang di Bukalapak. Rasa iseng mulai menggelitik pemuda tersebut setelah melihat fitur File Upload di laman profil. Pikiran pemuda tersebut langsung mengarah pada celah yang baru sebulan ditemukan oleh Nikolay Ermishkin, yaitu celah ImageTragick, sesuatu yang pernah pemuda tersebut bahas di Botani CTF.

Kunjungi [https://www.bukalapak.com/users/\(USER ID KAMU\)/edit?section=general](https://www.bukalapak.com/users/(USER ID KAMU)/edit?section=general) untuk mengunggah *payload*.



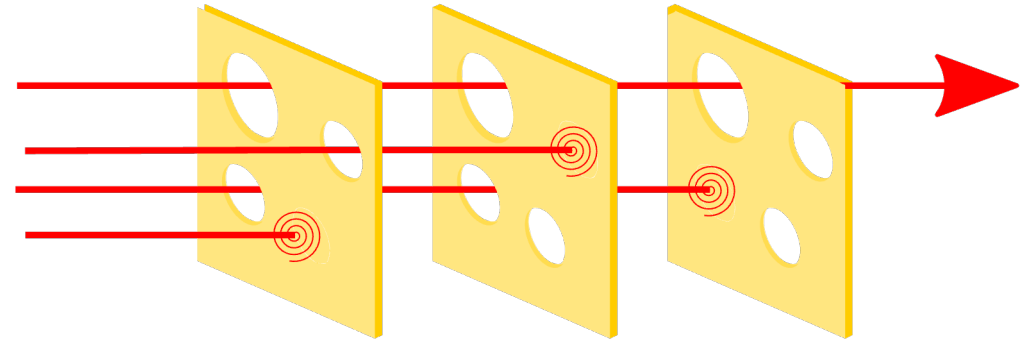
Upload *payload* yang telah dibuat dan ... *voila!*

Update: Saat ini saya telah direkrut Bukalapak sebagai Security Engineer.

There are a lot of possible cases to check!

Swiss cheese model in PPT

- **People**
 - **Awareness of best security practices (this part)**
- **Process**
 - QA and regular security checks
- **Technology**
 - Automating tests (will be explained later)





Tools

OWASP ZAP

The screenshot displays the OWASP ZAP 2.4.3 interface. The main window shows a 'Welcome to the OWASP Zed Attack Proxy (ZAP)' message. Below the message, there is a 'URL to attack' field containing 'http://www.owasp.org' and buttons for 'Attack' and 'Stop'. The 'Progress' bar indicates 'Spidering the URL to discover the content' at 4%.

The left sidebar shows a tree view of the site structure, including folders like 'GET:Category:SAMM-ST-2', 'GET:Category:OWASP_Tool', and 'GET:Category:OWASP_Release_Quality_Tool'. The bottom section shows a table of processed URLs.

Processed	Method	URI	Flags
●	GET	https://www.owasp.org/index.php/Mohd_Fazli_Azran	
●	GET	https://www.owasp.org/index.php/John_Vargas	
●	GET	https://docs.google.com/spreadsheets/d/1m00e983giNVwhbuQ96cmb79t...	OUT_OF_SCOPE
●	GET	https://www.owasp.org/images/d/da/WASPY_2015_Sponsorship_Docume...	
●	GET	http://owasp.blogspot.com/2015/06/2015-waspy-award-nominations.html	OUT_OF_SCOPE
●	GET	https://mail.google.com/mail/u/0/?ik=f64bf2af68&th=14e21153b3fe690d&ui...	OUT_OF_SCOPE
●	GET	https://mail.google.com/mail/u/0/?ik=f64bf2af68&th=14e211c0353117cd&ui...	OUT_OF_SCOPE
●	GET	https://twitter.com/owasp/status/613372502698532864	OUT_OF_SCOPE
●	GET	https://www.facebook.com/groups/owaspfoundation/permalink/798497196...	OUT_OF_SCOPE
●	GET	https://plus.google.com/116933056486234813396/posts/EFTauUZjyuE	OUT_OF_SCOPE
●	GET	https://mail.google.com/mail/u/0/?ik=f64bf2af68&th=14e3182abfd79146&ui...	OUT_OF_SCOPE
●	GET	https://twitter.com/owasp/status/614528448325771266	OUT_OF_SCOPE
●	GET	https://plus.google.com/116933056486234813396/posts/1JcQN92vZHA	OUT_OF_SCOPE
●	GET	http://owasp.blogspot.com/2015/06/nominate-your-waspy-candidates-today...	OUT_OF_SCOPE
●	GET	https://www.facebook.com/groups/owaspfoundation/permalink/800341350...	OUT_OF_SCOPE
●	GET	https://twitter.com/owasp/status/624676858127368194	OUT_OF_SCOPE

The bottom status bar shows 'Alerts 0 1 6 0' and 'Current Scans 0 0 0 0 1 0 0'.

Burp Suite

The screenshot displays the Burp Suite Community Edition v1.7.34 interface. The top menu bar includes options like Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, User options, and Alerts. Below this is a filter bar with 'Site map' and 'Scope' tabs. The main area shows a list of HTTP requests and responses. The selected request is a GET request to /bWAPP/login.php, which is highlighted in orange. The details pane below shows the raw request and response data.

Host	Method	URL	Params	Sta...	Length	MIME type	Title	Comment
http://localhost	GET	/bWAPP/csrf_2.php		200	13665	HTML	bWAPP - CSRF	
http://localhost	GET	/bWAPP/csrf_2.php?...	✓	200	13668	HTML	bWAPP - CSRF	
http://localhost	GET	/bWAPP/csrf_2.php?...	✓	200	13665	HTML	bWAPP - CSRF	
http://localhost	GET	/bWAPP/js/html5.js		200	2684	script		
http://localhost	GET	/bWAPP/login.php		200	4321	HTML	bWAPP - Login	
http://localhost	GET	/bWAPP/portal.php		200	23676	HTML	bWAPP - Portal	
http://localhost	GET	/bWAPP/reset.php		200	13598	HTML	bWAPP - Reset	
http://localhost	GET	/bWAPP/sqli_7.php		200	13553	HTML	bWAPP - SQL Injection	
http://localhost	POST	/bWAPP/sqli_7.php	✓	200	13847	HTML	bWAPP - SQL Injection	
http://localhost	POST	/bWAPP/sqli_7.php	✓	200	13814	HTML	bWAPP - SQL Injection	

Request /bWAPP/login.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: security_level=0; PHPSESSID=i6q80lthkjc13l1dnl3743n3q7
Connection: close
Upgrade-Insecure-Requests: 1

SAMLRaider (Burp Extension)

 CompassSecurity / SAMLRaider

Code

Issues19

Pull requests1

Actions

Projects

Wiki

Security

Insights

 SAMLRaider Public

Watch21

Fork76

Starred371

master2 branches13 tags

Go to fileAdd fileCode

emanuelduss Version 1.4.13cbbd8d on May 12, 2022152 commits

doc	Explanation on how to debug extension in IntelliJ	last year
lib	Development	8 years ago
scripts	Test script in python, more stable	3 years ago
src	Style fix	last year
.classpath	Use Java 14	3 years ago
.gitignore	Include javax.xml so extension works on OpenJDK 11	4 years ago
.project	Initial commit	8 years ago
BappDescription.html	Updated BApp description	3 years ago
BappManifest.bmf	Version 1.4.1	last year
LICENSE	Initial commit	8 years ago
README.md	Version 1.4.1	last year
build.gradle	BApp update	3 years ago
pom.xml	Version 1.4.1	last year

About

SAML2 Burp Extension

samlburp

Readme

MIT license

Activity

371 stars

21 watching

76 forks

Report repository

Releases12

Release v1.4.1Latest

on May 12, 2022

+ 11 releases

Packages

No packages published

Core Feature	Burp Suite	OWASP ZAP
Automation	Offers both automated and semi-automated processes	Uses an automation framework to give users more flexibility
Coverage	An impressive range of built-in features ensures your system has comprehensive coverage	Testers need to use add-ons to increase coverage
Scalability	Designed to scale up to large-scale testing engagements	Very scalable but has some limitations
Accuracy	Highly accurate and minimal false positives produced	Very accurate but there are limitations to the scope of the scanner
Cost	Three available editions: Community (Free), Professional (\$449/year), and Enterprise (starts at \$8,395/year)	Free

openapi-fuzzer

If you're using frameworks that support OpenAPI (e.g.: FastAPI), you can also use Fuzzer tools to test the behavior of your application.

While it's mainly used for checking functionalities correctness, it could also potentially detect incorrect logic that might lead to security holes.

OpenAPI fuzzer [↗](#)

CI passing

Black-box fuzzer that fuzzes APIs based on [OpenAPI specification](#). All you need to do is to supply URL of the API and its specification. Find bugs for free!

METHOD	PATH	STATUS	MEAN (µs)	STD.DEV.	MIN (µs)	MAX (µs)
GET	auth/token/accessors/	ok	230	76	163	1037
POST	auth/token/create	ok	435	78	316	941
POST	auth/token/create-orphan	ok	476	94	366	1175
POST	auth/token/create/{role_name}	ok	446	250	156	3037
GET	auth/token/lookup	ok	331	67	260	900
POST	auth/token/lookup	failed	341	38	286	444
POST	auth/token/lookup-accessor	ok	395	122	300	1840
GET	auth/token/lookup-self	ok	346	54	284	643
POST	auth/token/lookup-self	ok	388	186	301	2970
POST	auth/token/renew	failed	375	96	279	831
POST	auth/token/renew-accessor	ok	417	167	321	2744
POST	auth/token/renew-self	ok	383	99	280	1275
POST	auth/token/revoke	failed	322	21	287	367
POST	auth/token/revoke-accessor	ok	389	85	310	963
POST	auth/token/revoke-orphan	failed	369	69	286	592
POST	auth/token/revoke-self	ok	357	85	262	1053
GET	auth/token/roles	ok	225	41	180	456
GET	auth/token/roles/{role_name}	ok	308	176	128	2338
POST	auth/token/roles/{role_name}	ok	466	245	135	1845
DELETE	auth/token/roles/{role_name}	ok	308	163	123	1788
POST	auth/token/tidy	ok	311	38	258	487
GET	cubbyhole/{path}	ok	208	57	131	650
POST	cubbyhole/{path}	ok	274	214	122	3227
DELETE	cubbyhole/{path}	ok	295	124	126	716
POST	identity/alias	ok	366	96	261	1085
GET	identity/alias/id	ok	209	34	163	388
GET	identity/alias/id/{id}	ok	320	144	123	810
POST	identity/alias/id/{id}	ok	360	198	121	2119
DELETE	identity/alias/id/{id}	ok	329	171	120	1252
POST	identity/entity	ok	388	134	312	2316
POST	identity/entity-alias	ok	363	33	314	508
GET	identity/entity-alias/id	ok	200	29	166	417
GET	identity/entity-alias/id/{id}	ok	324	184	129	2263



Lesson Learned

Try to break your own system
before someone else breaks it

Try to minimize reinventing the wheel
Focus on delivering value & solving real problems

Apply principle of least privilege
Minimize attack surface when it's possible

No monitoring ≠ no security incident
Always be ready for incident (runbook, incident
response team, etc)

Keep up-to-date with security news
(Zero-day vulnerability)

Last but not least,
Human is often the weakest link
Raise awareness, reduce social engineering attack

AI: Threat? Opportunity? Both?

- Threat

- Don't trust AI-generated code (e.g.: Copilot) blindly
- New attack vectors generation
 - Phishing
 - GAN
 - and many more

- Opportunity

- Automate security checks (e.g.: Firewall, threat analysis)
- Automate code tests generation



Twitter

[freedom_holicx](https://twitter.com/freedom_holicx)



Instagram

[freedomofkeima](https://www.instagram.com/freedomofkeima)



Facebook

[iskandarsetiadi](https://www.facebook.com/iskandarsetiadi)



LinkedIn

[iskandarsetiadi](https://www.linkedin.com/in/iskandarsetiadi)



Portfolio

freedomofkeima.com



Github

[freedomofkeima](https://github.com/freedomofkeima)

HENNGE is hiring!

Mid-career



Internship



This slide is also available at
<https://freedomofkeima.com/pyconid2023.pdf>



Upcoming Next

Parallel Session 7

19 November 2023 • 14:10 – 14:50

Room 4

Beginnter

EN

Automating Victory: Beating Browser Games with Accessible Python

Jon Gaul

• Game • Automation



Terima kasih!

Hatur nuhun!

Thank you!

ありがとうございます！

