



black hat[®]

USA 2019

AUGUST 3-8, 2019

MANDALAY BAY / LAS VEGAS

Preventing Authentication Bypass A Tale of Two Researchers

Terry Zhang, Tophant
Ron Chan, Independent Researcher
Ravi Jaiswal, Microsoft

Cast of characters



Terry Zhang, co-founder of Tophant and specialist in breaking identity, working independently of...



Ron Chan, an independent security researcher, who concurrently with Terry was investigating...



Outlook Web App, a mail service created by...



Microsoft, a technology company, represented here by...



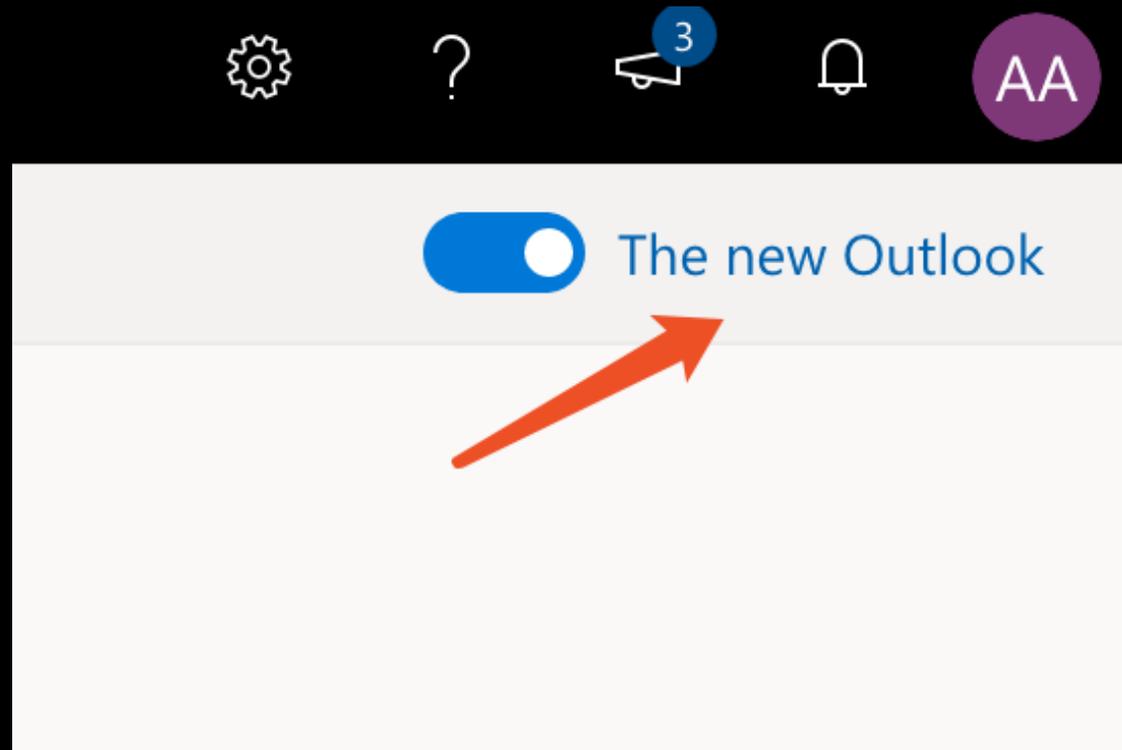
Ravi Jaiswal, a principal engineering manager for Outlook Web App

Chapter 1

the first discovery

Some time before Sunday, March 24, 2019

A new UI option? Let's check it out.



Look at that!

- Intercepting the request using Burp Suite, I discovered a new API endpoint I'd never seen before.

```
POST /owa/service.svc?action=GetAccessTokenforResource&UA=0&n=14&EP=1&app=Mail HTTP/1.1
Host: outlook.office.com

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
content-type: application/json; charset=utf-8
action: GetAccessTokenforResource
x-owa-correlationid: 55c43e39-af14-f3ce-88d5-adcea46af253
x-owa-urlpostdata: {"__type":"TokenRequest:#Exchange","Resource":"https://outlook.office.com"}
```

An unsigned token?

- The endpoint had an action called **GetAccessTokenForResource**, and returned an unsigned JSON Web Token (JWT)

```
{"TokenResultCode":0,"AccessToken":"eyJhbGciOiJIub250IiwidHlwIjoiSldUI  
n0.eyJvaWQiOiIwYmQwMmY0Yi1iNjBiLTRIODYtOWMzNi01YzE3MzExYTIh  
NzkiLCJwdWlkIjoiMTAwMzlwMDA0MTgzRjYzQSIsInNtdHAIoiJhdHRhY2tlc  
kB0b3BoYW50Lm9ubWljcm9zb2Z0LmNvbSIsInVwbiI6ImF0dGFja2VyQHR  
vcGhhbnQub25taWNyb3NvZnQuY29tliwic2NwIjoiZnVsbF9hY2Nlc3NfYXN  
fZXhjaGFuZ2UgRFdF.....5mMy0zYVFYNIJBliwidGkljoiMDAtM  
DAwMDAwMDAwMDAwQDI5NzNjZDU3LTlzN2MtNGRiYy04NTI3LTM2NG  
ZkNmY4MTQzNlslmF1ZCI6Imh0dHBzOi8vb3V0bG9vay5vZmZpY2UuY29t  
n0.","AccessTokenExpiry":"2019-03-24T04:13:55.183Z","ExpiresIn":890}
```


This didn't
look like
much.

But then I tried it out in
the Outlook REST API.

Use the Outlook REST API (version 2.0)

09/07/2018 • 10 minutes to read

Applies to: Exchange Online | Office 365 | Hotmail.com | Live.com | MSN.com | Outlook.com | Passport.com

Note

Use [Microsoft Graph](#) to build richer scenarios for [Microsoft 365 services](#) including Outlook. Find out how to [transition to Microsoft Graph-based Outlook REST API](#).

The Outlook REST API includes the following subsets to allow access to users' mailbox data in Office 365, Hotmail.com, Live.com, MSN.com, Outlook.com, and Passport.com.

The following table lists the first version that the core functionality of each subset was made available. Note that within a subset, new features and API can be subsequently added to a later version.

| API subset | Available versions |
|---|--------------------|
| Batching multiple API calls | v2.0, beta |
| Calendar API | v2.0, v1.0, beta |
| Contacts API | v2.0, v1.0, beta |
| Data Extensions API | v2.0, beta |
| Extended Properties API | v2.0, beta |
| Mail API | v2.0, v1.0, beta |
| Push Notifications API | v2.0, beta |
| Streaming Notifications API (preview) | beta |

Normally, you'd need a user's authorization to get a JWT.

Request API permissions

< All APIs

Exchange
https://outlook.office365.com/ Docs

What type of permissions does your application require?

Delegated permissions
Your application needs to access the API as the signed-in user.

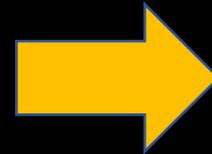
Application permissions
Your application runs as a background service or daemon signed-in user.

Select permissions

Type to search

| PERMISSION | ADMIN CONSENT REQUIRED |
|---|------------------------|
| Calendars (3) | |
| Contacts | |
| EAS | |
| EWS | |
| Exchange | |
| Group | |
| MailboxSettings | |
| Mail (5) | |
| <input checked="" type="checkbox"/> Mail.Read Read user mail | - |
| <input checked="" type="checkbox"/> Mail.Read.All Read user and shared mail | - |
| <input checked="" type="checkbox"/> Mail.Read.Shared Read user and shared mail | - |
| <input type="checkbox"/> Mail.ReadBasic Read user basic mail | - |
| <input checked="" type="checkbox"/> Mail.ReadWrite | - |

Add permissions Discard



Microsoft

permissions.microsoft.com

Permissions requested

azuredevops

This app would like to:

- Read your calendars
- Read calendars you can access
- Read and write to your calendars
- Read your mail
- Read mail you can access
- Read and write to your mail
- Read and write mail you can access
- Read your profile
- Read all users' basic profiles
- Read all users' basic profiles
- Read and write your profile
- Sign you in and read your profile

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. **The publisher has not provided links to their terms for you to review.** You can change these permissions at <https://myapps.microsoft.com>. [Show details](#)

Cancel Accept

What about consumer Outlook?

- The consumer Outlook service is the same as enterprise, just running on different domains.
- Both services support **GetAccessTokenforResource**
- Interesting: all consumer mail services (Hotmail, Live, Outlook) have the same tenant ID.

```
POST
/owa/service.svc?action=GetAccessTokenforResource&UA=0&n=11&EP=1&ap
p=Mail HTTP/1.1
Host: outlook.live.com

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:65.0)
Gecko/20100101 Firefox/65.0

Accept: */*

Accept-Language: en-US,enAccept-Encoding: gzip, deflate
;q=0.5

Referer: https://outlook.live.com/
x-req-source: Mail

content-type: application/json; charset=utf-8

action: GetAccessTokenforResource

x-owa-urlpostdata:
%7B%22__type%22%3A%22TokenRequest%3A%23Exchange%22%2C%22Re
source%22%3A%22https%3A%2F%2Foutlook.live.com%22%7D

Origin: https://outlook.live.com

Connection: close

Content-Length: 0
```

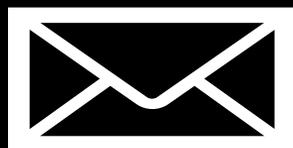

The result

In comparing the unsigned JWT from two personal accounts, I learn that the TID is always the same.

Although I can't easily get the OID for another personal account...

... I find I can substitute the email address to impersonate any user in consumer cloud.

This looks like a good candidate for bounty!



secure@microsoft.com

Chapter 2

the second discovery

Some time before Thursday, April 4, 2019

I saw a retweet.



Security Response 

@msftsecresponse

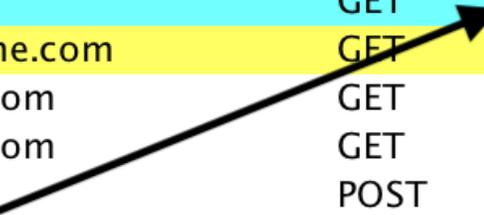


Calling all present and future bounty hunters! See our new blog post on improvements to the Microsoft vulnerability bounty program to increase some awards and pay bounties more quickly.

blogs.technet.microsoft.com/msrc/2019/04/0...

Let's try using Burp Suite on Outlook Web App.

| | | | |
|--------|-----------------------------------|------|--|
| 166... | https://outlook.office.com | GET | /search/api/v1/init?scenario=owa.react.compose&n=56&cv=eRF7... |
| 166... | https://login.microsoftonline.com | GET | /common/oauth2/authorize?response_type=token&client_id=89b... |
| 166... | https://lpcres.delve.office.com | GET | /lpc/versionless/livepersonacard_with-react_50d4b0cf59099dfe7... |
| 166... | https://lpcres.delve.office.com | GET | /lpc/versionless/livepersonacardstrings_0084387ce23bf8cf2e17e... |
| 166... | https://outlook.office.com | POST | /owa/service.svc?action=GetConversationItems&n=25&app=Mail |
| 166... | https://lpcres.delve.office.com | GET | /lpc/versionless/vendors~aria-webjs-sdk_2c2485272112800b95... |



I'd almost given up when I found the API, thanks to the JSON Web Token Attacker plugin

Just another JWT sending requests to an API endpoint.

```
GET /search/api/v1/init?scenario=owa.react.compose&n=56&cv=eRF7TBr9Qnms5BUM0uXOXV.56
HTTP/1.1
Host: outlook.office.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-OWA-CANARY: xnZa0TAnR0SILNIepuDtgpDorWW6D9cYgoa35VdboDpJF3URbAnDWVs_Rz_CakM-j9h6rSDBiQg.
X-MS-AppName: owa-react mail
MS-CV: eRF7TBr9Qnms5BUM0uXOXV.56
Authorization: Bearer
eyJhbGciOiJIJSUzI1NiIsImtpZCI6IjA2MDBGOUY2NzQ2MjA3MzdFNzM0MDRFMjg3QzQ1QTgxOENCN0NFQjgiLCJ4NXQiO
```

```
yPhv4IrbQnuGJX46yA
```

Let's use that plugin to look more closely.

Nothing special in the header...

The payload has R/W permissions—nothing special...

Now we're getting somewhere.

The screenshot shows a network traffic analysis tool interface. It displays a request and response, with the request selected. The request is expanded to show its headers, and the payload is selected. The payload is a JWT token, which is further expanded to show its claims. The claims include:

- oid: "2a64..."
- puid: "1..."
- smtp: "..."
- com: "..."
- upn: "..."
- ver: "Exchange.Callback.V2"
- appid: "00000002-0000-0ff1-ce00-000000000000"
- deploymentid: "https://outlook.office365.com/"
- tid: "e3cf3c98-a978-465f-8254-9d541eeea73c"
- acr: "1"
- appidacr: "0"
- scp: "OWA.AccessAsUser.All EWS.AccessAsUser.All DWEEngine-Internal.Read Locations-Internal.ReadWrite SubstrateSearch-Internal.ReadWrite TailoredExperiences-Internal.ReadWrite MailboxSettings.ReadWrite Todo-Internal.ReadWrite Notes.ReadWrite Notes-Internal.ReadWrite"

Below the JWT token, there is a large black box with a white "!" and "?" symbol, indicating a point of interest or a question.

What can we do with an unsigned JWT?

- Figure out what parameter the API uses to identify me
- Change that parameter to impersonate others
- Apply the JWT to as many endpoints as possible to see what we can do.

```
{"oid":"2a64bab5-ddf4-4630-15bd-9c3e1ba094c5","puid":"20031FFFBA1D0425","smtp":"ron.chan@testing.com","upn":"ron.chan@testing.com","ver":"Exchange.Callback.V2","appid":"00000002-0000-0ff1-ce00-000000000000","deploymentid":"https://outlook.office365.com/","tid":"e3cf3c98-a978-465f-8254-9d541eaaa73c","acr":"1","appidacr":"0","scp":"OWA.AccessAsUser.All EWS.AccessAsUser.All DWEngine-Internal.Read Locations-Internal.ReadWrite SubstrateSearch-Internal.ReadWrite TailoredExperiences-Internal.ReadWrite MailboxSettings.ReadWrite Todo-Internal.ReadWrite Notes.ReadWrite Notes-Internal.ReadWrite Premium-Internal.ReadWrite Mail.ReadWrite Calendars.ReadWrite Signals.ReadWrite ConnectedAccount-Internal.ReadWrite Connectors.ReadWrite.Shared User.Read Group.ReadWrite.All Calendars-Internal.ReadWrite FocusedInbox-Internal.ReadWrite User.ReadWrite","nbf":1563921787,"exp":1563922687,"iss":"https://outlook.office365.com/","aud":"https://outlook.office.com"}
```

Next steps: RTFM to get the OID

Consulted the API documentation to learn how to find people relevant to me...

... then get their OID...

... and then get the messages in their mailbox.

```
GET  
/api/beta/me/messages?appId=Mail&userType=Business&n=43&cv=F09fy%2  
Ba1FaSk61M%2BPBWwDZ.43 HTTP/1.1
```

```
"Subject": "Re: [REDACTED]"  
"BodyPreview": "Great, thanks Ron. [REDACTED]"
```

The result

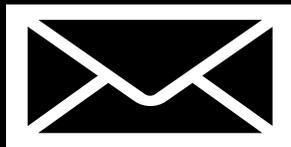
I can get an unsigned JWT.

I can get another person's OID

I can substitute that OID in the unsigned JWT and send it to the mail service.

This allows me to retrieve their email.

This looks like a good candidate for bounty!



secure@microsoft.com

Chapter 3

the response

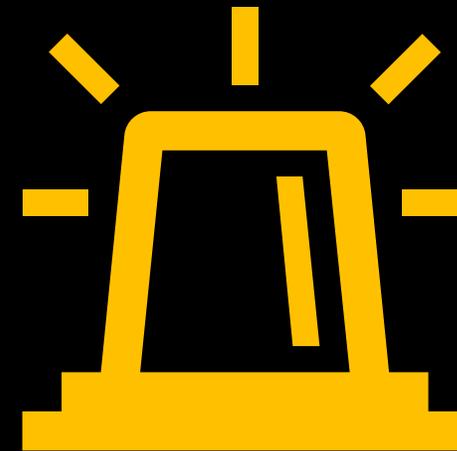
Beginning March 24, 2019

Sunday, March 24, 2019

- First report received, triaged, and cased for remediation

Thursday, April 4, 2019

- Second report received, triaged and cased.
- The Microsoft Security Response Center initiates a Software and Services Incident Response Plan (SSIRP) to coordinate response.

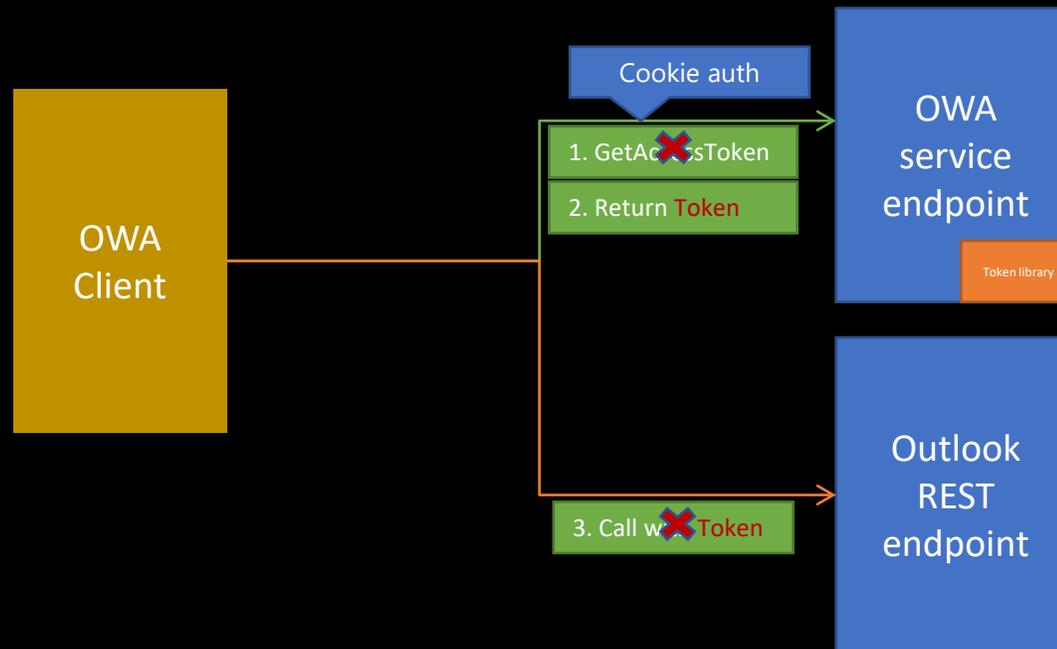


Saturday, April 6, 2019

The OWA team mitigates the vulnerability by shutting off the API path.



The mitigation



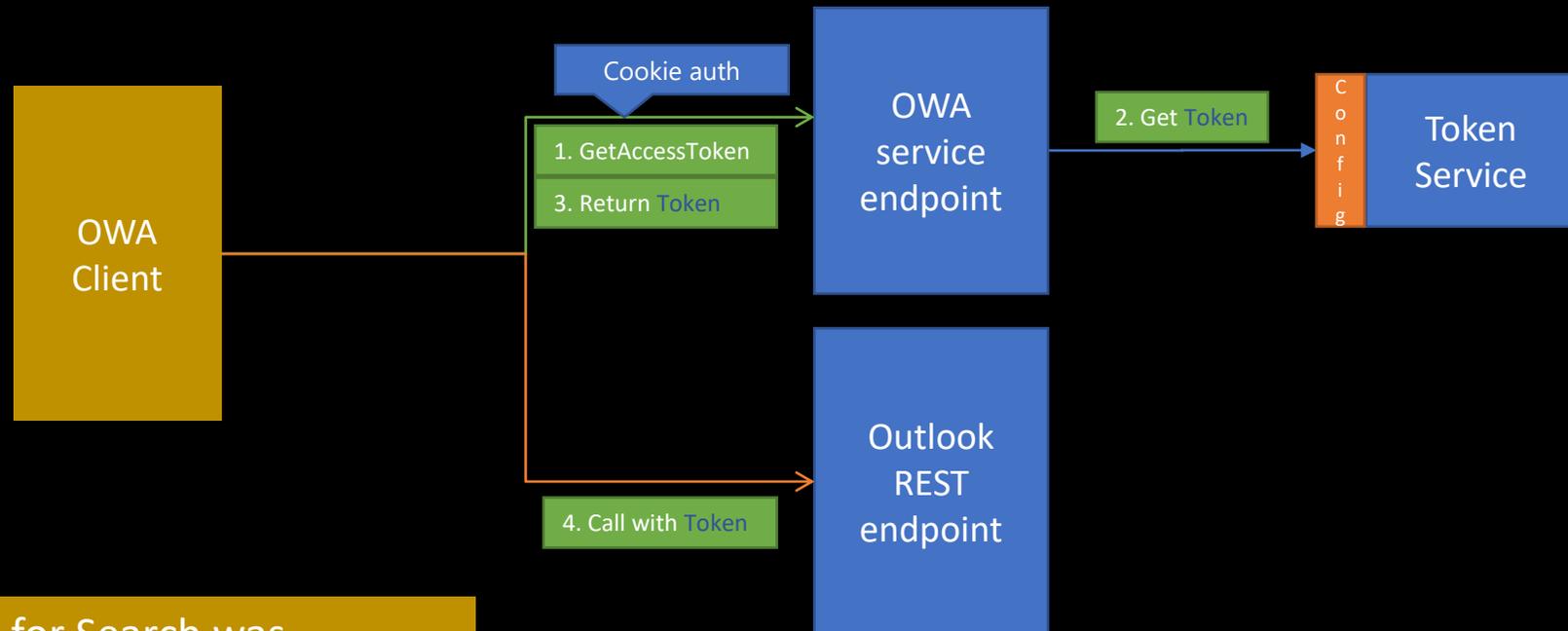
Search depended on the API, so the OWA team temporarily rerouted search to use a new path until the fix was ready.

Wednesday, April 24,
2019

The OWA team finished rolling out a fix that prevented clients from being able to acquire the editable token.



The fix



As part of the fix the flow for Search was restored.

Wednesday, April 24, 2019

Reply Reply All Forward ...
Wed 4/24/2019 5:41 PM

 Kimberlee Price
To  Tianqi Zhang
Cc  Jarek Stanley;  Microsoft Security Response Center

Hi Terry,

Thank you for your recent vulnerability report, MSRC case 51087. This is a great discovery, and we appreciate your partnership to help us protect Microsoft customers.

I am writing for two reasons. First, I wanted to let you know that we are preparing to award you with a \$40,000 bounty for this report. As you know, the [standard award](#) for a high quality, critical EoP vulnerability is \$8,000. We are increasing the award to better reflect the quality and impact of your vulnerability submission.

Thank you for your recent vulnerability report, MSRC case 51256. This is a great discovery, and we appreciate your partnership to help us protect Microsoft customers.

I am writing for two reasons. First, I wanted to let you know that we are preparing to award you with a \$20,000 bounty for this report. As you know, the [standard award](#) for a medium quality, critical EoP vulnerability is \$4,000. We are increasing the award to better reflect the quality and impact of your vulnerability submission.

Forward ...
Wed 4/24/2019 5:41 PM

Thank you for your partnership!

The outcome

Researchers reported the issue to Microsoft

Microsoft quickly mitigated the issue to protect customers

Comprehensive fix was rolled out shortly afterwards

The researchers received bounty rewards

Customers were protected

Epilogue

avoiding repeats and key learnings

Key learnings for developers

- Continue promoting resource-scoped and one-time tokens
- Drive to end support for legacy token types
- Set security policies for token allocation

Key learnings for service providers

- Ready your incident response process
- Review scenario security
- Keep improving flow security
- Partner with security community to find flaws early

Key learnings for researchers

- Change is an opportunity to find new attack surface
- Keep improving your toolbox
- Make sure to document and screenshot your work

Resources

Download Burp Suite

<https://portswigger.net/>

Read the Outlook Mail API

<https://docs.microsoft.com/en-us/graph/outlook-mail-concept-overview>

Develop a security incident response plan

<https://msrc-blog.microsoft.com/2019/06/25/inside-the-msrc-customer-centric-incident-response/> (1st in a series)

Follow Terry and Ron on Twitter

@pnig0s and @ngalongc

Thank you!

Terry, Ron, and Ravi