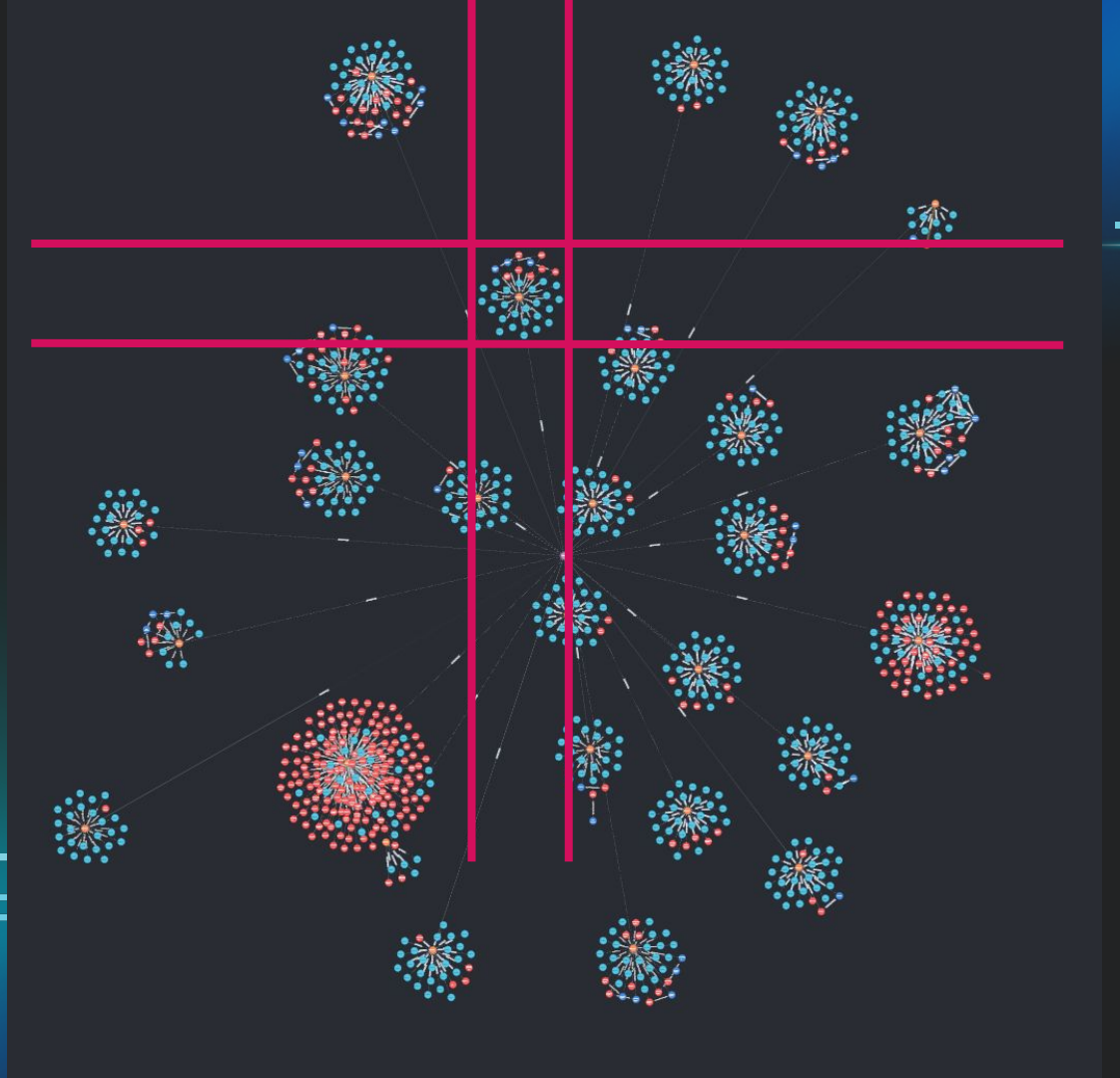
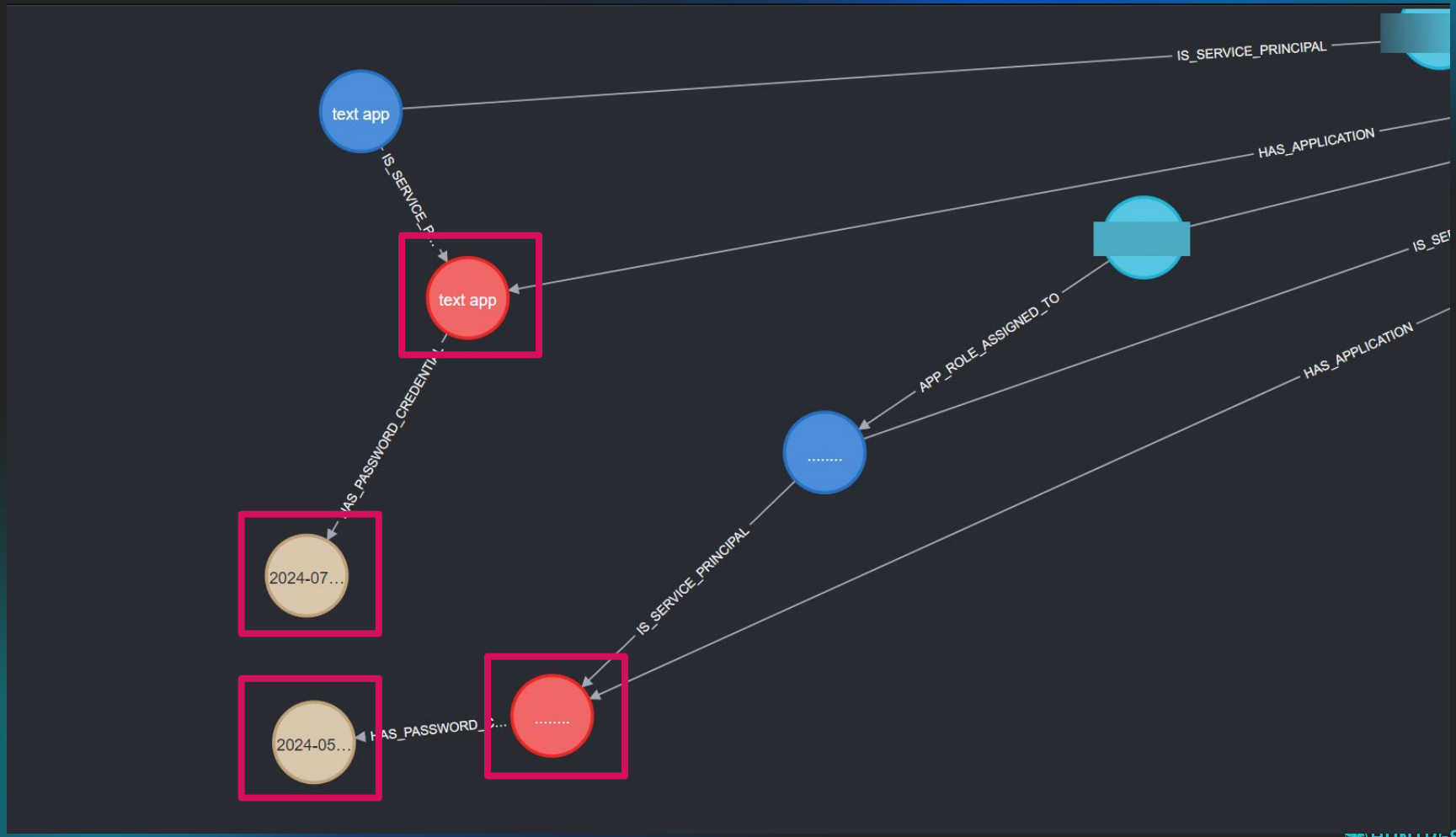
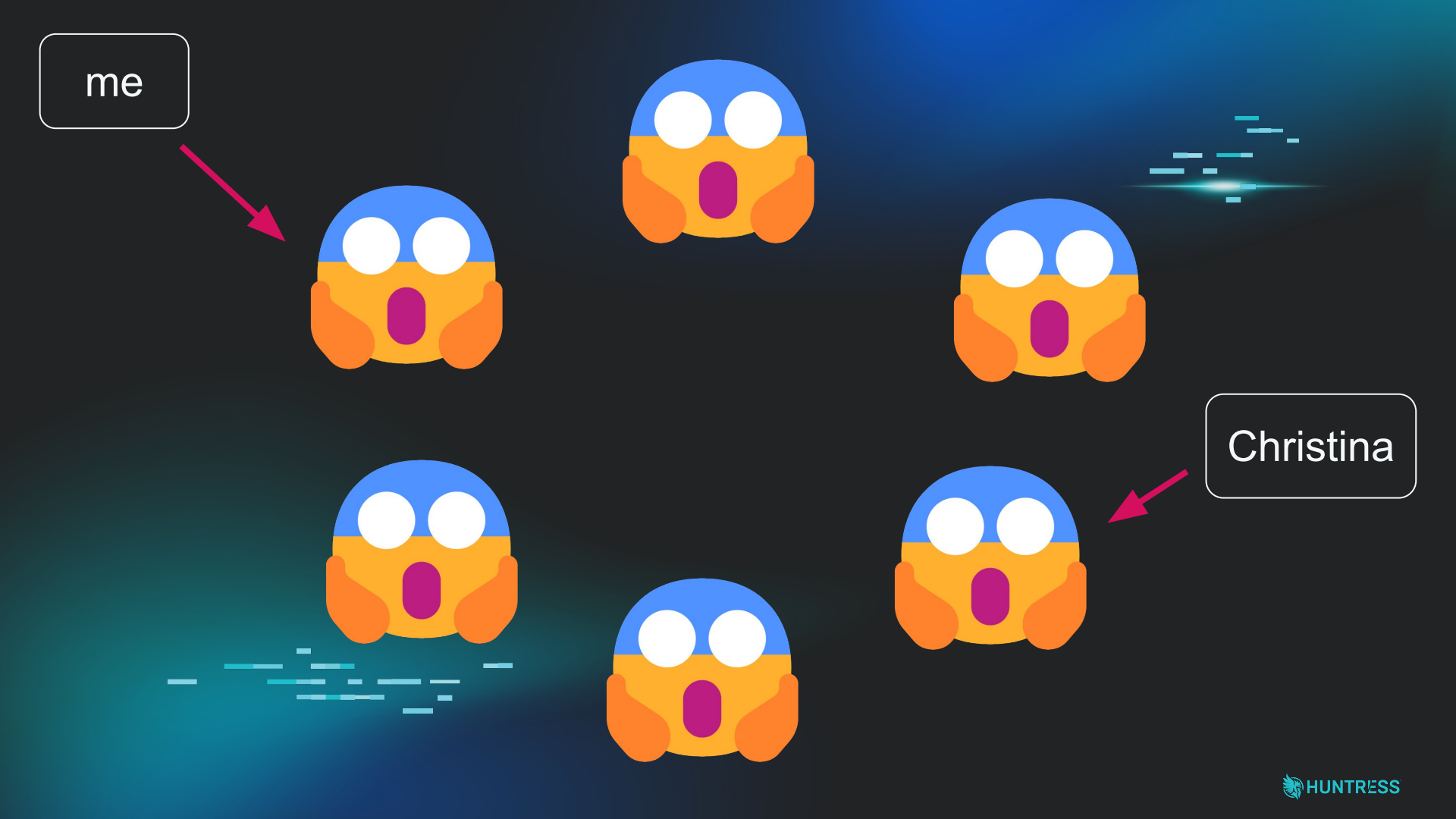
Oct 11th, 2024

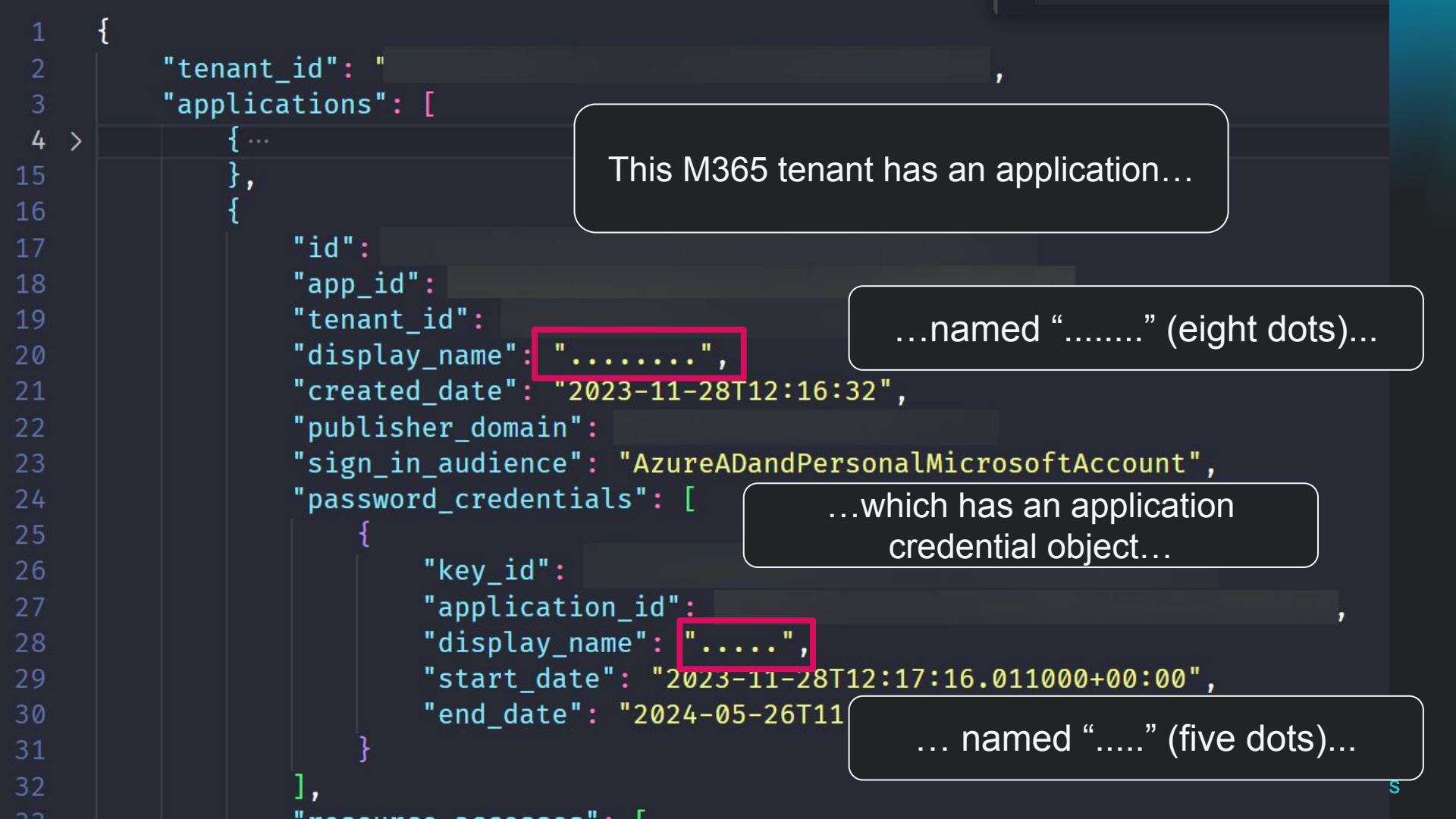
… at the Huntress Research Facility…

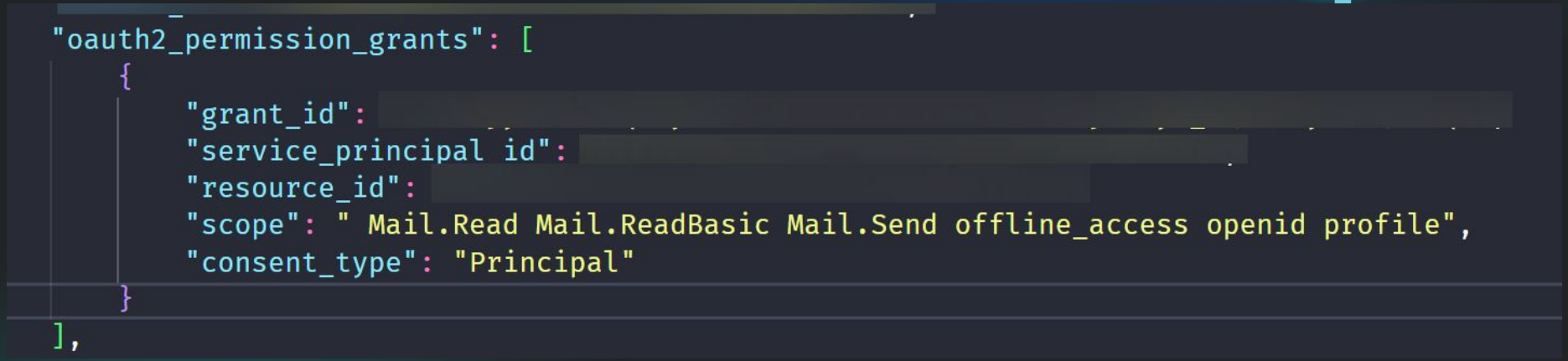


HUNTRESS

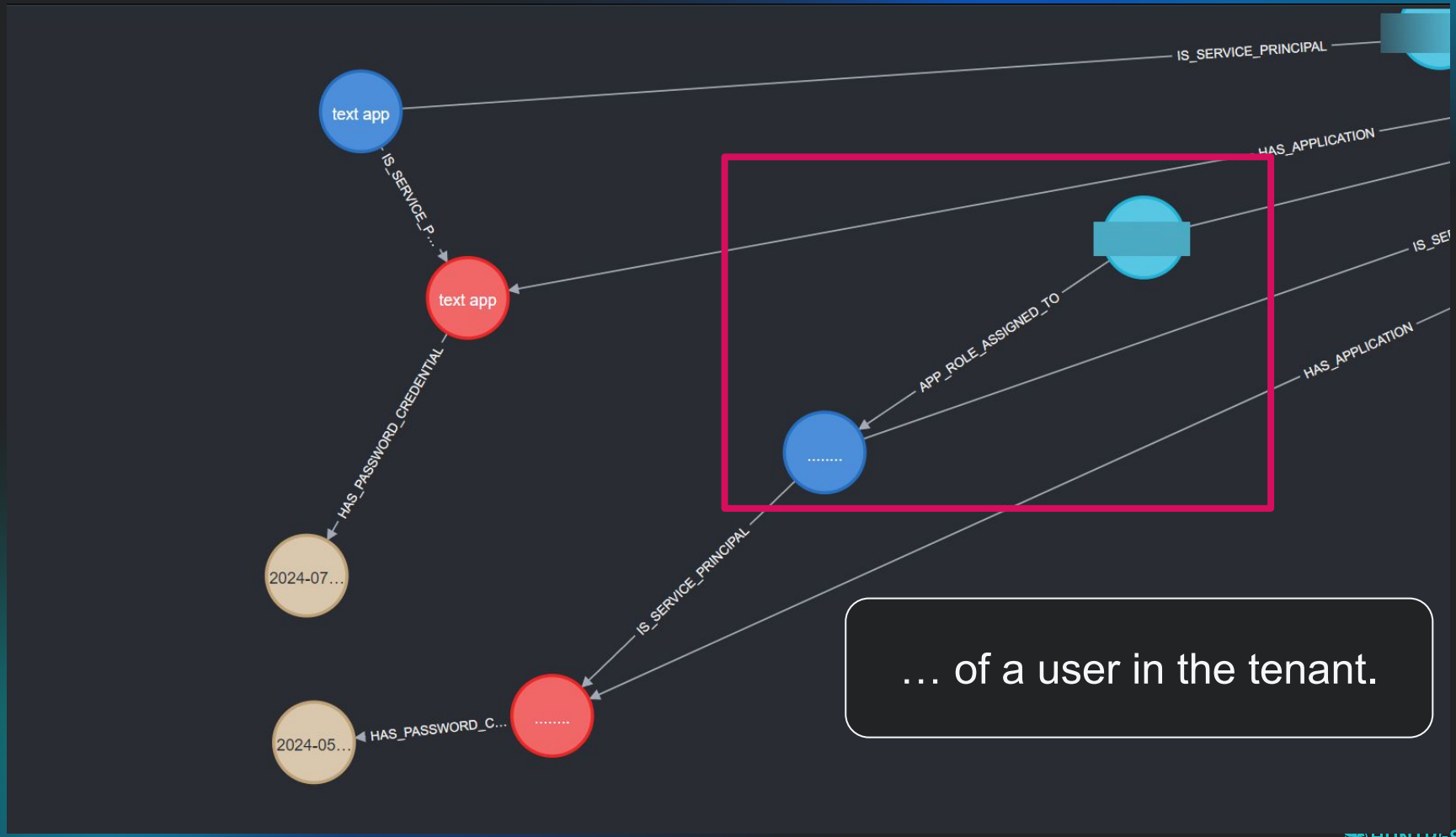

text app
IS_SERVICE_PRINCIPAL
HAS_APPLICATION
text app
APP_ROLE_ASSIGNED_TO
HAS_APPLICATION
2024-07…
IS_SERVICE_PRINCIPAL
2024-05…


me
Christina


{
"tenant_id": "
"applications": [
{…
},
{
"id":
"app_id":
"tenant_id":
"display_name": ".....",
"created_date": "2023-11-28T12:16:32",
"publisher_domain":
"sign_in_audience": "AzureADandPersonalMicrosoftAccount",
"password_credentials": [
{
"key_id":
"application_id":
"display_name": ".....",
"start_date": "2023-11-28T12:17:16.011000+00:00",
"end_date": "2024-05-26T11
}
],
This M365 tenant has an application…
…named “.....” (eight dots)...
…which has an application credential object…
… named “.....” (five dots)...



And this app has permission to access the M365 mailbox contents…


text app
IS_SERVICE_PRINCIPAL
IS_SERVICE_P...
text app
HAS_APPLICATION
APP_ROLE_ASSIGNED_TO
.....
HAS_PASSWORD_CREDENTIAL
2024-07...
IS_SERVICE_PRINCIPAL
HAS_APPLICATION
2024-05...
HAS_PASSWORD_C...
.....
... of a user in the tenant.

Evidence of Azure application attacks in the Small to Medium Business ecosystem

“... how many more are out there...?”

When Apps Attack

Hunting Traitorware and Rogue Microsoft 365 Apps at the Small to Medium Business Scale

Matt Kiely
Christina Parry

Christina Parry (she/her)
Staff ThreatOps Developer

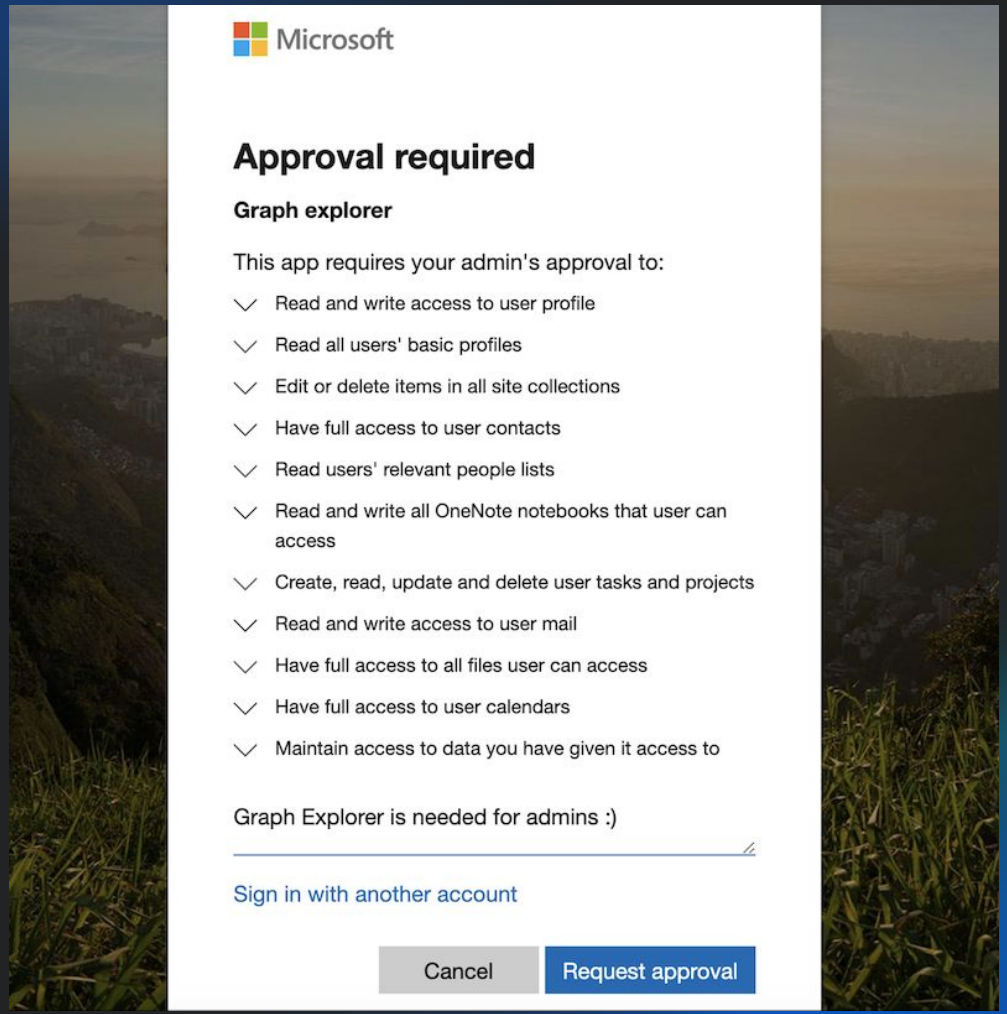
Matt Kiely (he/him)
~~Three caffeinated raccoons in a trenchcoat~~
Principal Security Researcher

Agenda

1. Azure App Crash Course
2. Tradecraft #1: Traitorware
3. Tradecraft #2: OAuth App Attacks
4. Hunting at the SMB Scale
5. Research Methods, Findings, Conclusion & Future Work
6. Questions & Acknowledgements

Azure App Crash Course

“Settle down, kids, it’s time to talk about OAuth”

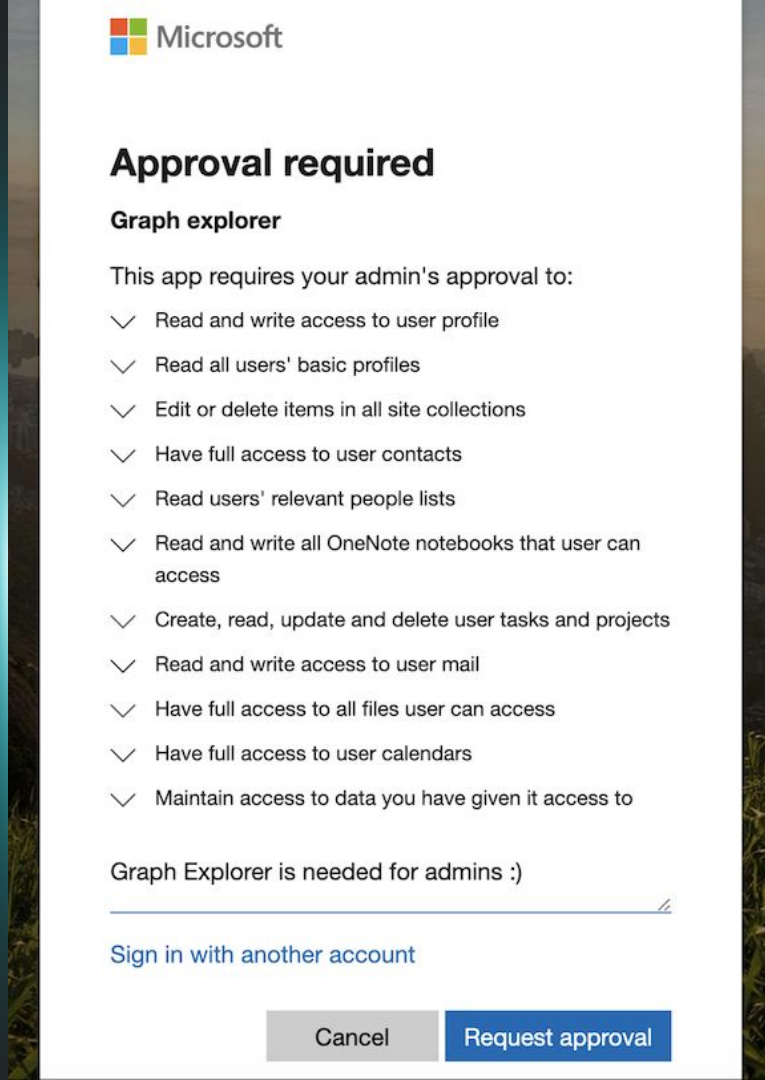

Microsoft
Approval required
Graph explorer
This app requires your admin's approval to:
Read and write access to user profile
Read all users' basic profiles
Edit or delete items in all site collections
Have full access to user contacts
Read users' relevant people lists
Read and write all OneNote notebooks that user can access
Create, read, update and delete user tasks and projects
Read and write access to user mail
Have full access to all files user can access
Have full access to user calendars
Maintain access to data you have given it access to
Graph Explorer is needed for admins :)
Sign in with another account
Cancel
Request approval

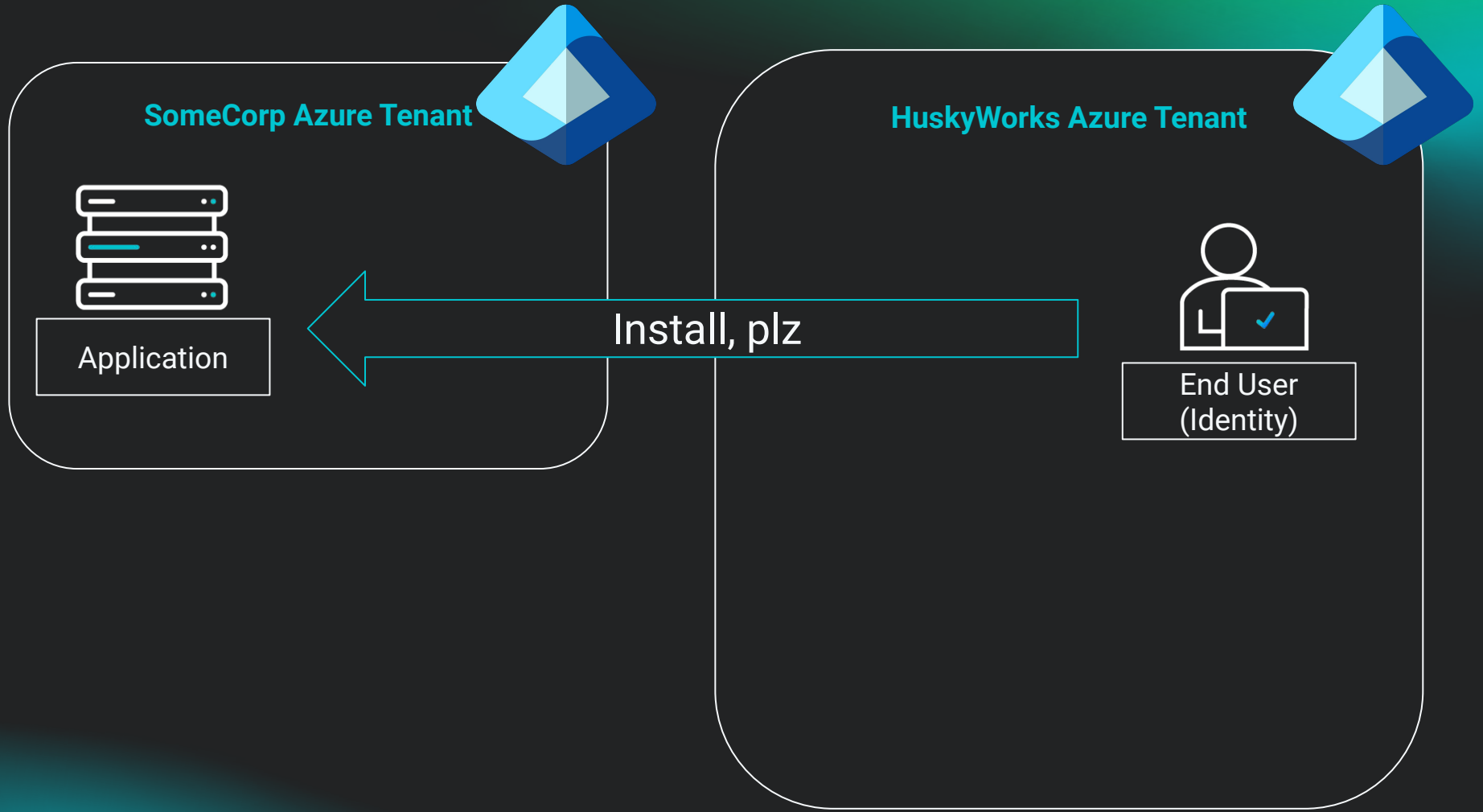
The Azure Application Ecosystem

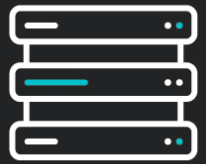
- Azure allows developers (developers, developers, developers!!!) to build apps in your tenant
  - First party apps for your own tenant
  - Globally accessible apps for everyone!

- Apps: web page, database, internal employee app, client, all kinds of things!

- Built on (opinionated and weird, tbh) OAuth2.0 implementation
  - “Kerberos, but for the cloud”




SomeCorp Azure Tenant
HuskyWorks Azure Tenant
Application
Install, plz
End User
(Identity)



Hi, I'm EmailBuddy! I can help you organize and backup your emails. To do this, you need to:

- Install me in your tenant
- Consent to the permissions that I need to access your email*
- Authenticate as your identity and let me on your behalf

*I use the Graph API to retrieve and backup your emails, so you need to allow me to access your email data ^.^

SomeCorp Azure Tenant

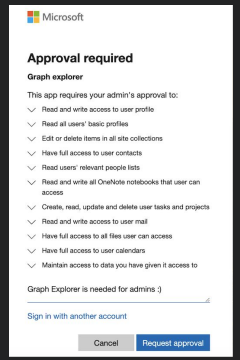
HuskyWorks Azure Tenant

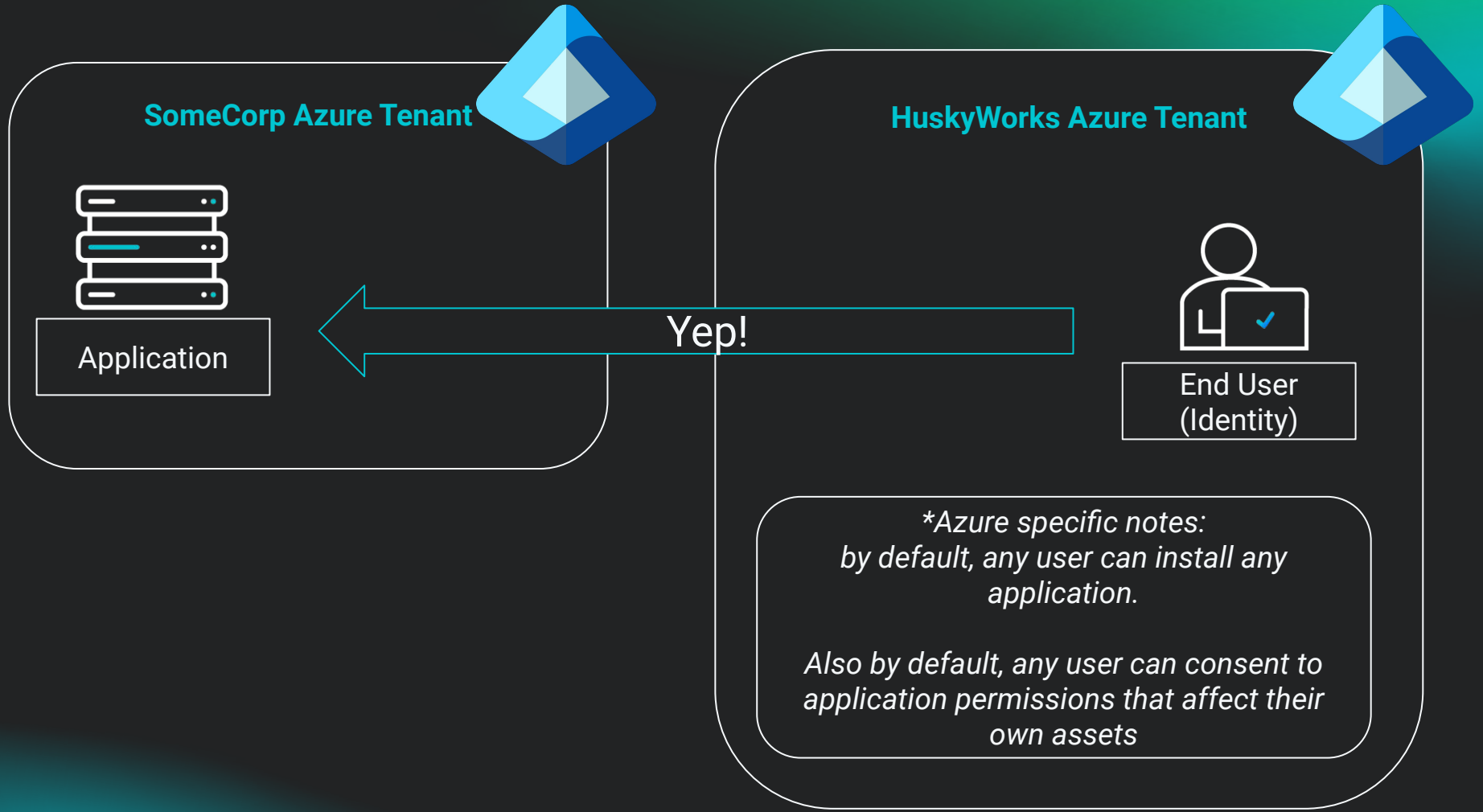
Application

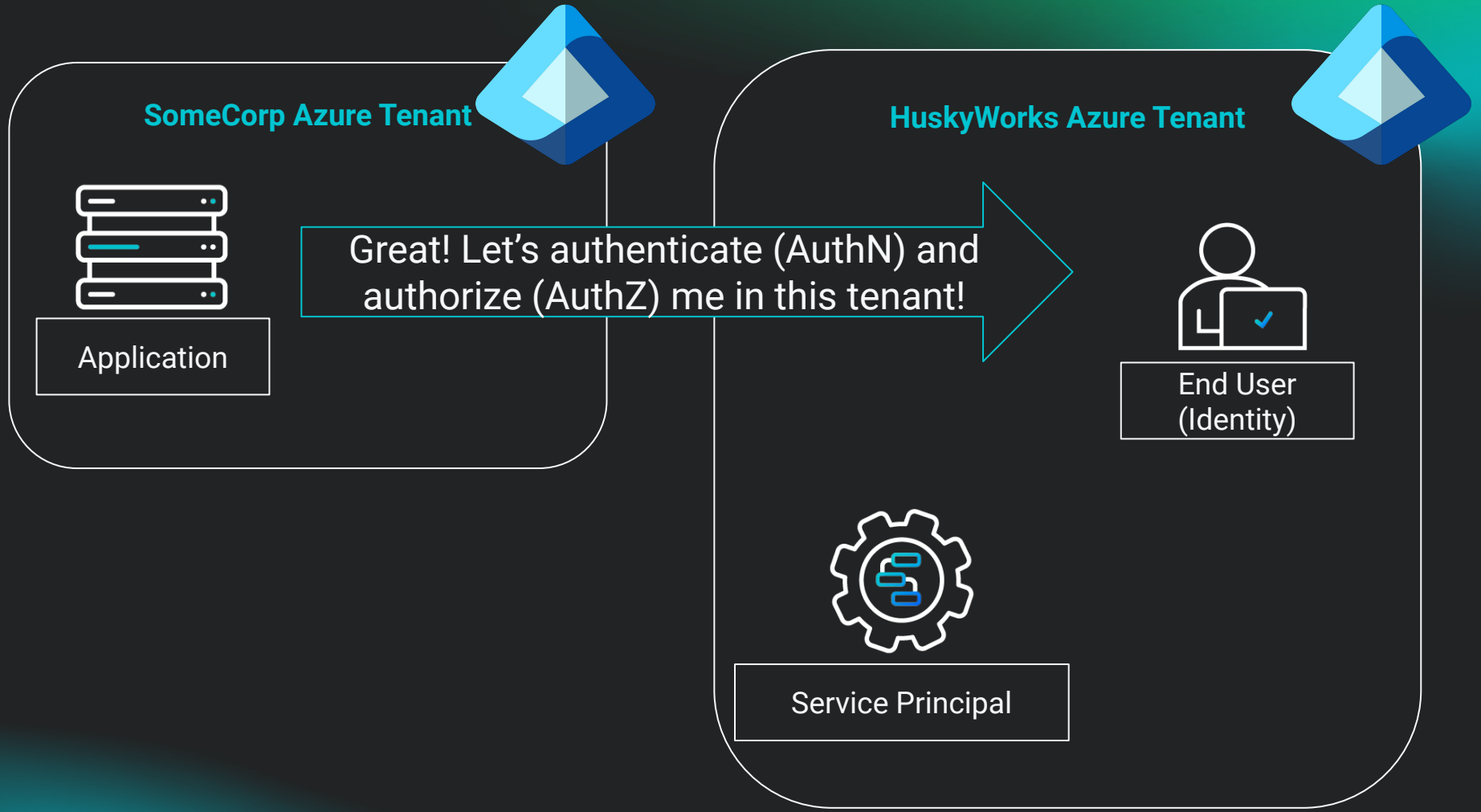
Do you consent to the following?

End User (Identity)

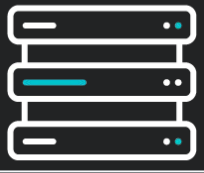
Mail.ReadWrite
Mail.Send
MailboxSettings.ReadWrite
Calendars.Read




SomeCorp Azure Tenant
HuskyWorks Azure Tenant
Application
Yep!
End User
(Identity)
*Azure specific notes:
by default, any user can install any application.
Also by default, any user can consent to application permissions that affect their own assets


SomeCorp Azure Tenant
HuskyWorks Azure Tenant
Application
Great! Let's authenticate (AuthN) and authorize (AuthZ) me in this tenant!
End User (Identity)
Service Principal

SomeCorp Azure Tenant



Application

HuskyWorks Azure Tenant

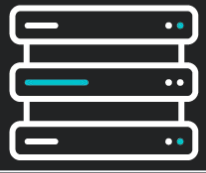
Greetings 🤖

I am an instance of EmailBuddy.

You, [USER], authenticated me into this tenant, so now I can access resources on your behalf

Service Principal

SomeCorp Azure Tenant



Application

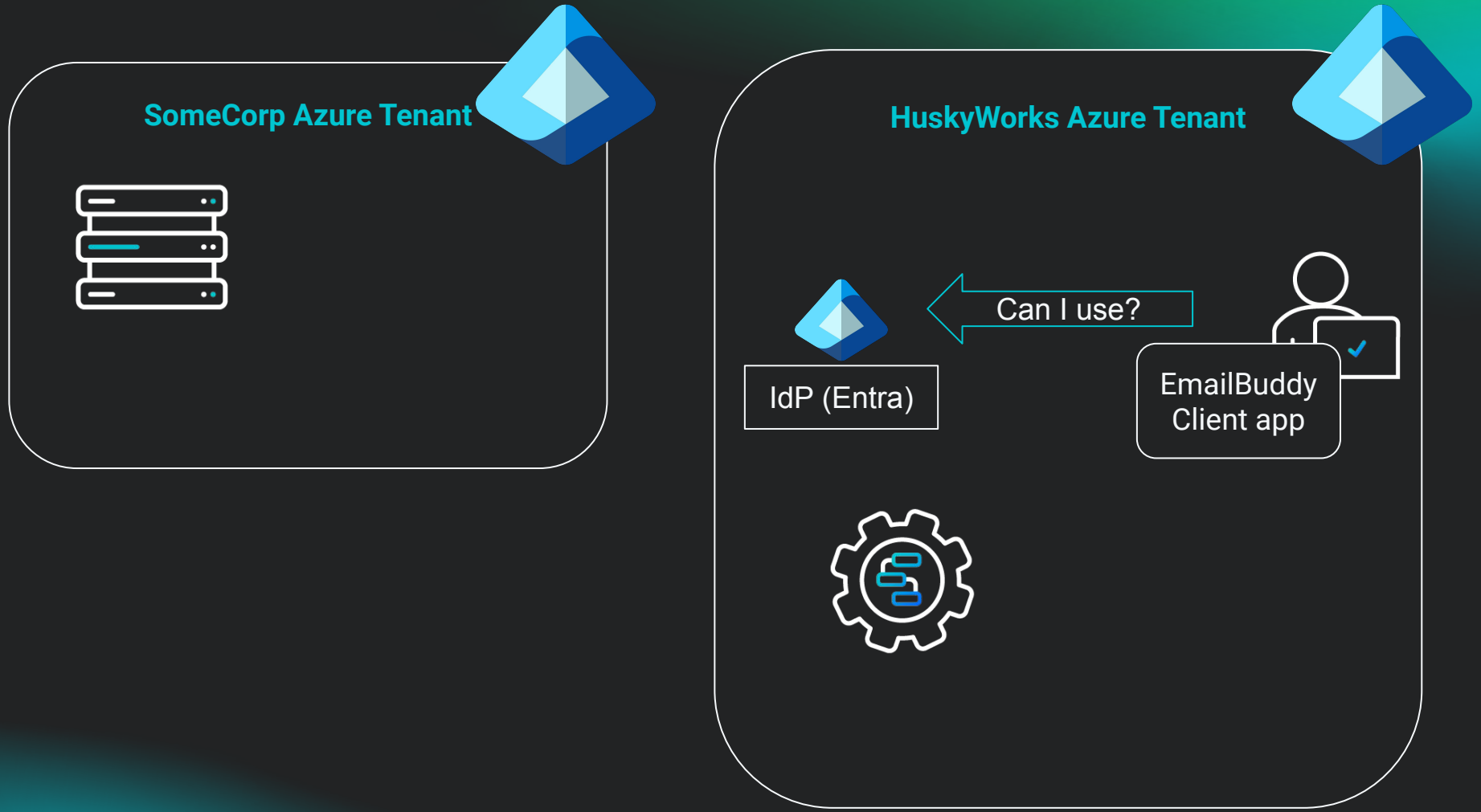
HuskyWorks Azure Tenant

End User (Identity)

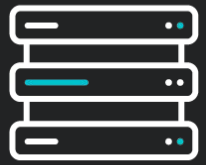
Service Principal

This app is now:

Authenticated ✅
Authorized ✅
Able to access resources on the end user's behalf ✅


SomeCorp Azure Tenant
HuskyWorks Azure Tenant
Can I use?
IdP (Entra)
EmailBuddy
Client app

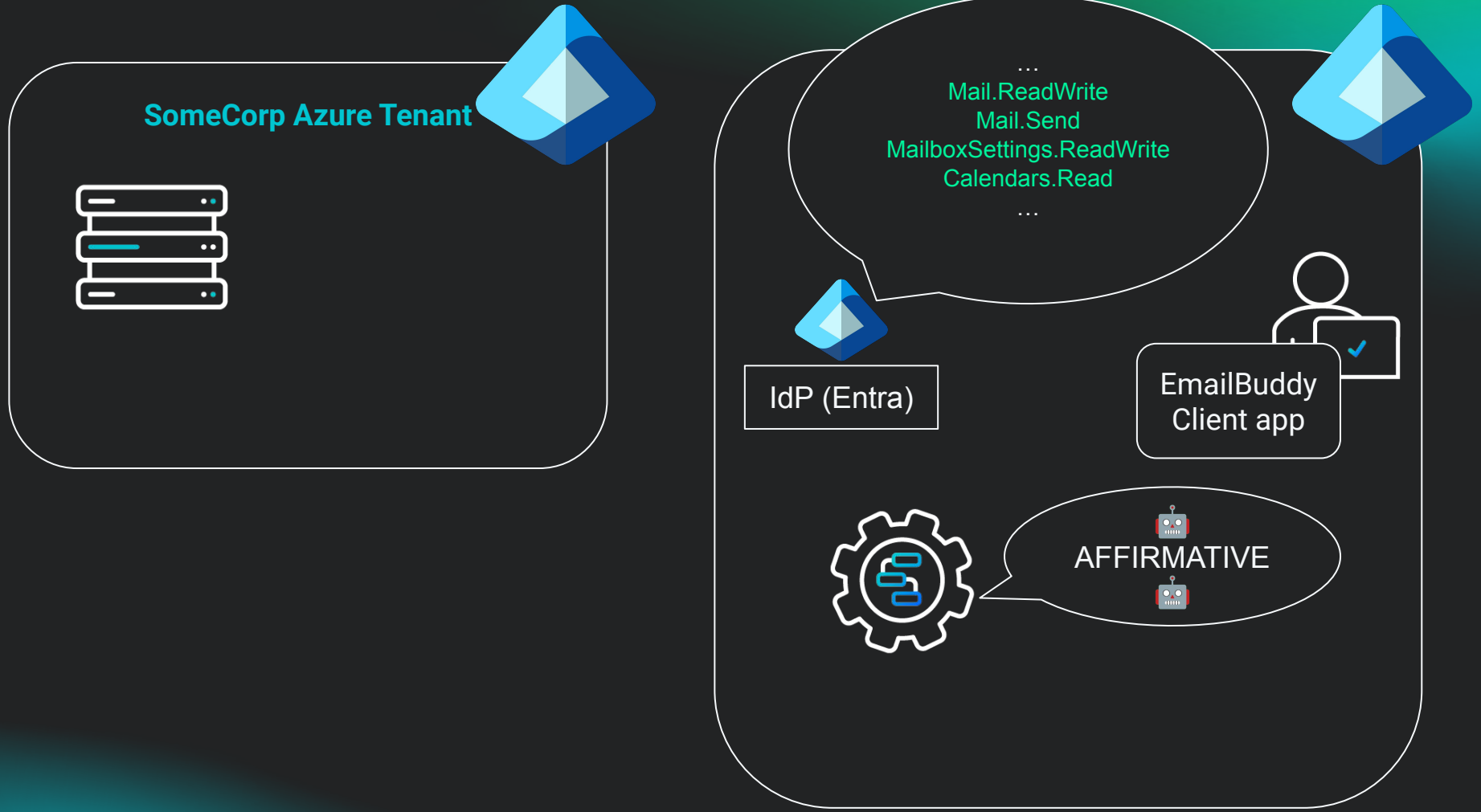
SomeCorp Azure Tenant



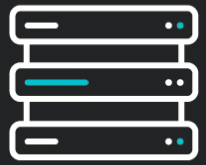
You are an authorized user of EmailBuddy in this tenant, which has been granted the following permissions…

IdP (Entra)

EmailBuddy Client app


SomeCorp Azure Tenant
...
Mail.ReadWrite
Mail.Send
MailboxSettings.ReadWrite
Calendars.Read
...
IdP (Entra)
EmailBuddy
Client app
AFFIRMATIVE

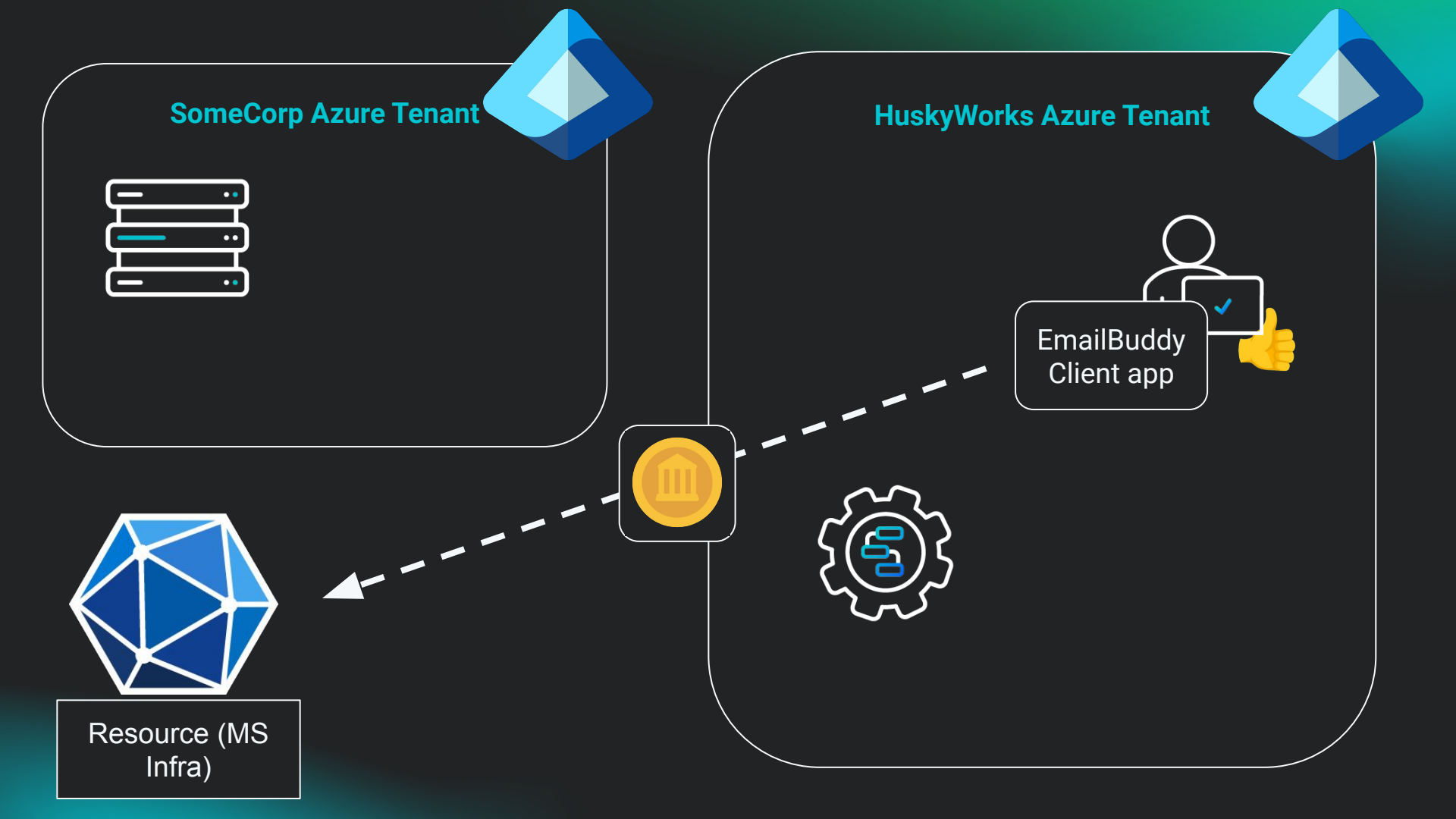
SomeCorp Azure Tenant



Looks good! Here's a token for this app to use with the Graph API.

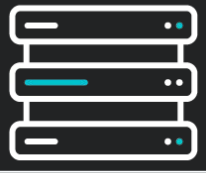
IdP (Entra)

EmailBuddy Client app


SomeCorp Azure Tenant
HuskyWorks Azure Tenant
EmailBuddy
Client app
Resource (MS
Infra)

What if the app is built in-house?

HuskyWorks Azure Tenant



Application

Install

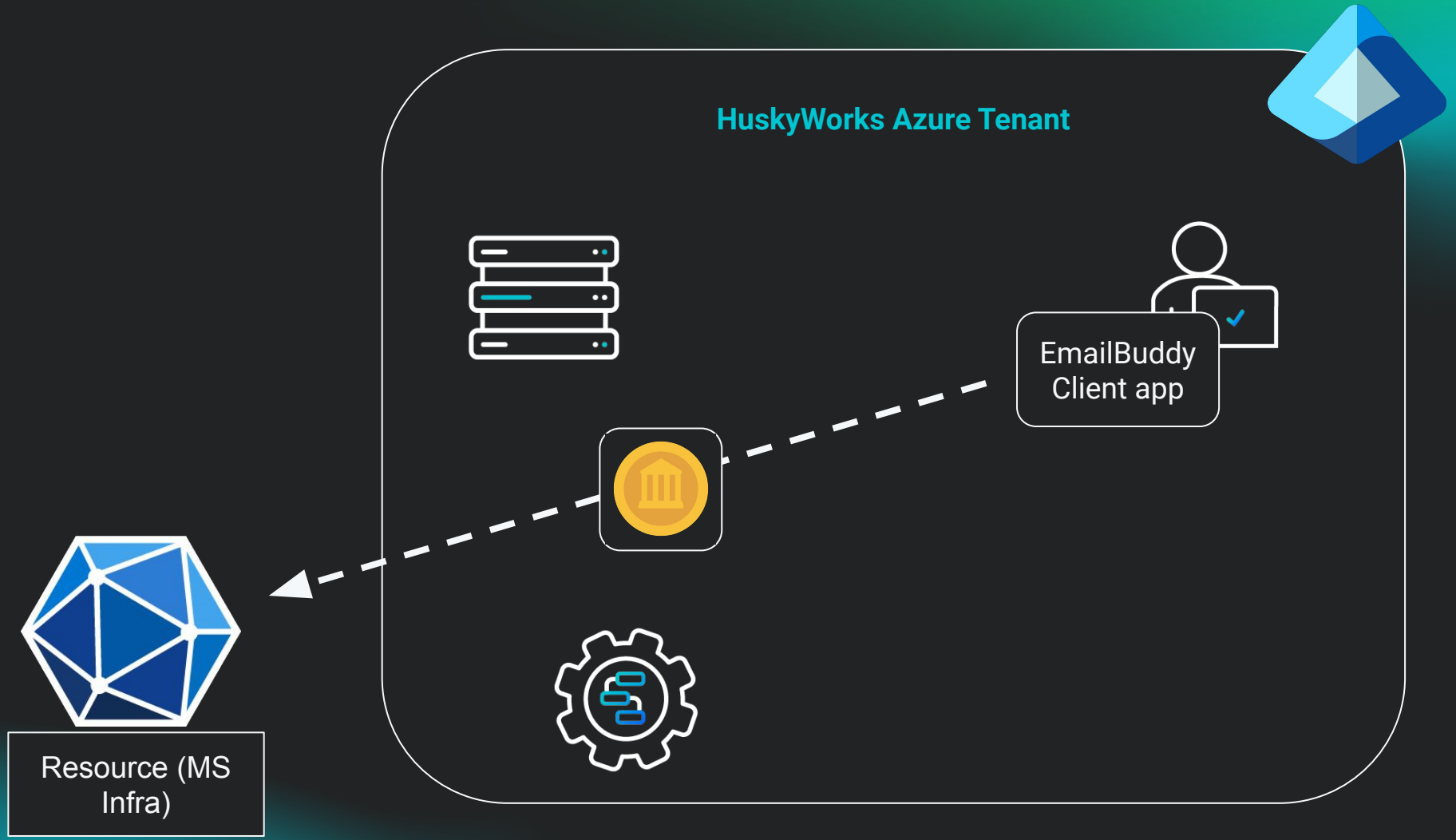
Consent?

AuthN / AuthZ

End User (Identity)

Service Principal

Resource (MS Infra)


HuskyWorks Azure Tenant
EmailBuddy
Client app
Resource (MS
Infra)

Notice how the application…

… can be built in-house or installed from another tenant…

… uses OAuth for AuthN and AuthZ (which tenant can I work in and what do I have access to?)...

… and has delegated access on behalf of one (or more) users.

Attacker can socially engineer a user into installing an app OR install one after initial access

Uses the built in schema of authentication and authorization (best exploits are FNAB)

Can be scoped to single users, multiple users, narrow permissions, broad permissions, anything an attacker wants!

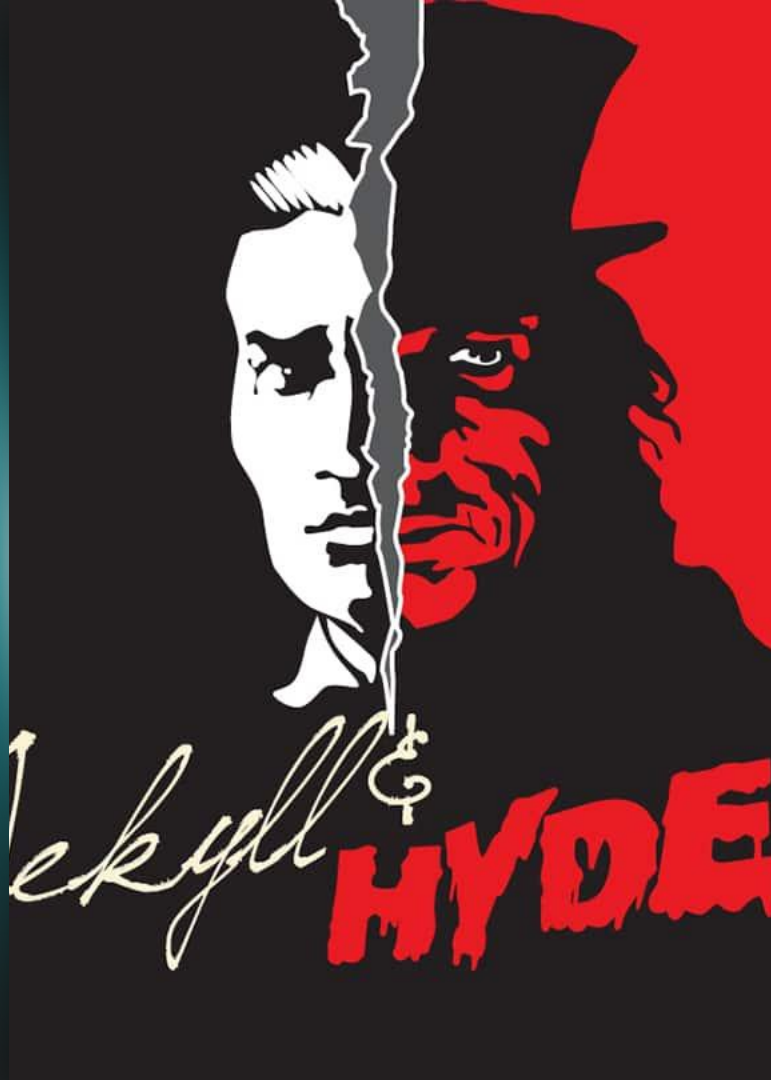
…which makes a fantastic system for exploitation.

Tradecraft #1: Traitorware

“Every App has a Dark Side”

Traitorware

- Application abuse where the app was not specifically built to be evil
- Closest approximation: living off RMM, BYO RMM, LOLBINS (kinda…)
- Commonly seen in BEC and financial attacks



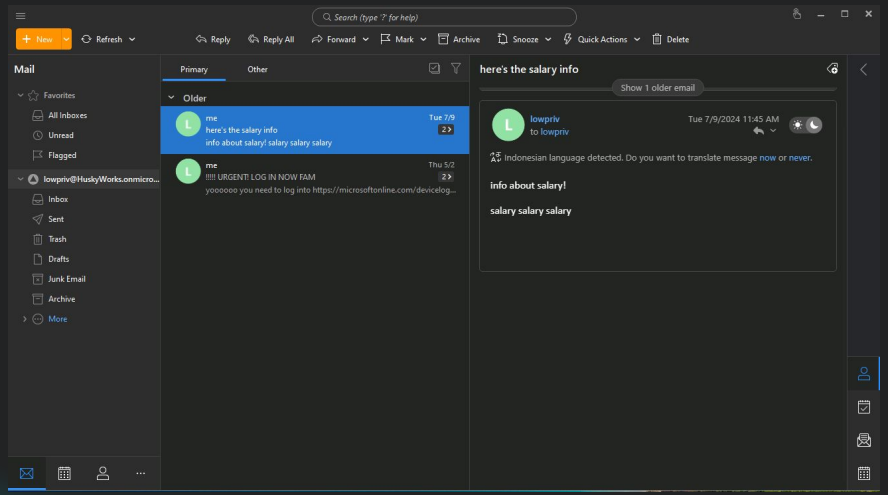
eM Client Features Mobile app For Companies Support Resources Pricing English Free download

Boost your email
Skyrocket your productivity

Get the best email client for Windows and macOS,
for professionals and home users alike.

Download for free

Available for: Windows, macOS, iOS and Android



New Account

1. Account details
2. Encryption
3. Finish

Almost there!

When you're all set, click the Finish button to create the account.

Account avatar

Change...

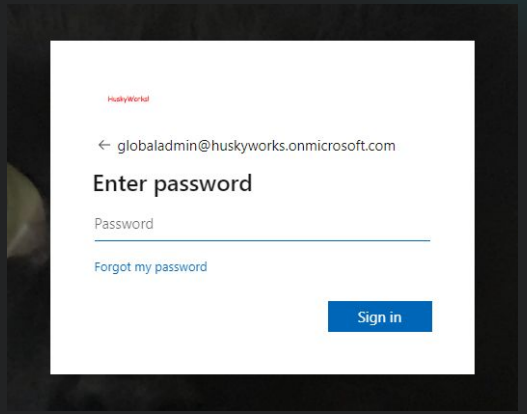
Sync Options

Password required for globaladmin@huskywork...

This operation may take a while to complete.
Please be patient.

Cancel

Back | Finish | Cancel



Microsoft

globaladmin@huskyworks.onmicrosoft.com

Permissions requested

eM Client

eM Client s.r.o.

This app would like to:

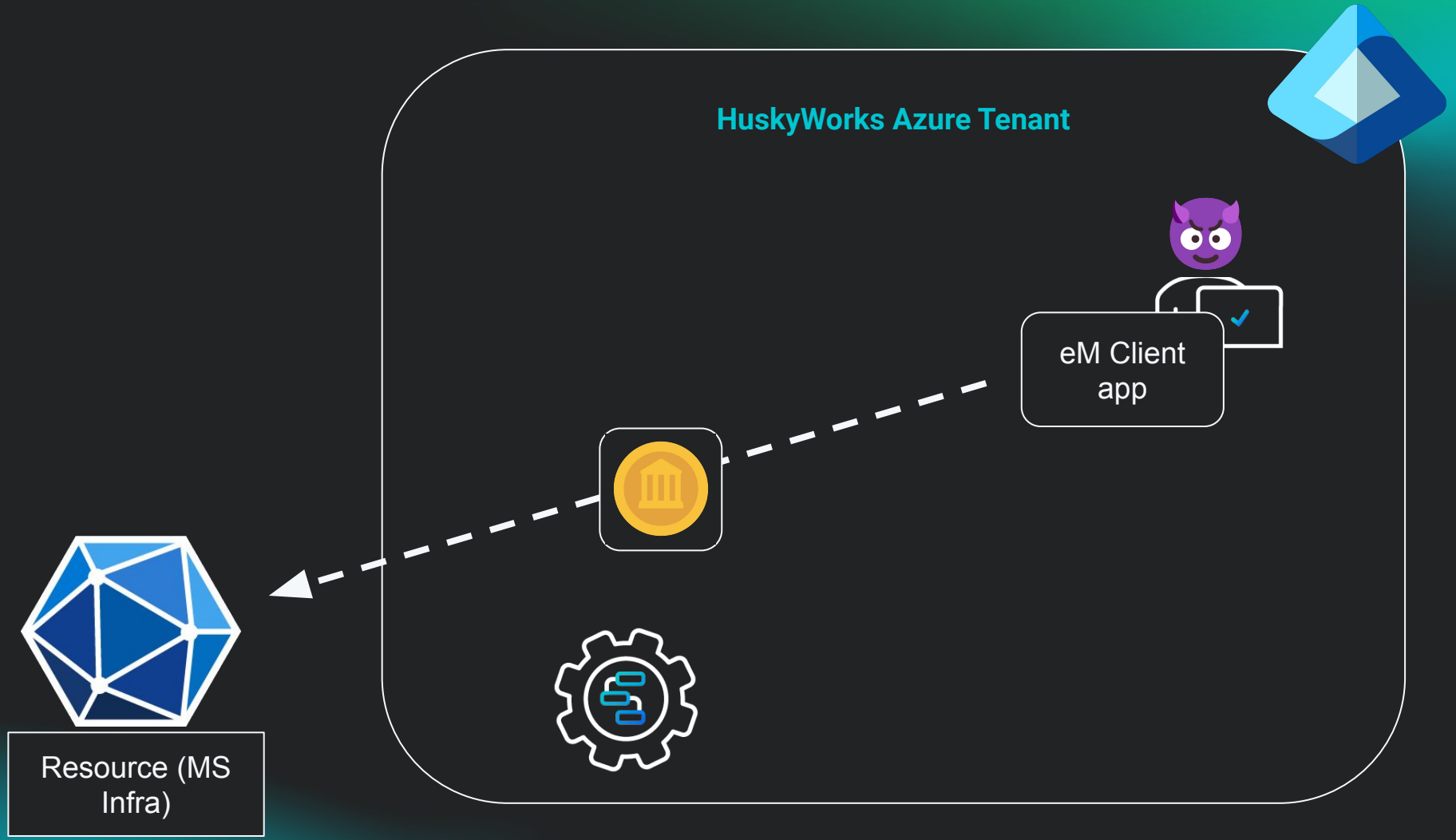
- Access your mailboxes
- Maintain access to data you have given it access to
- View your basic profile

☐ Consent on behalf of your organization

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

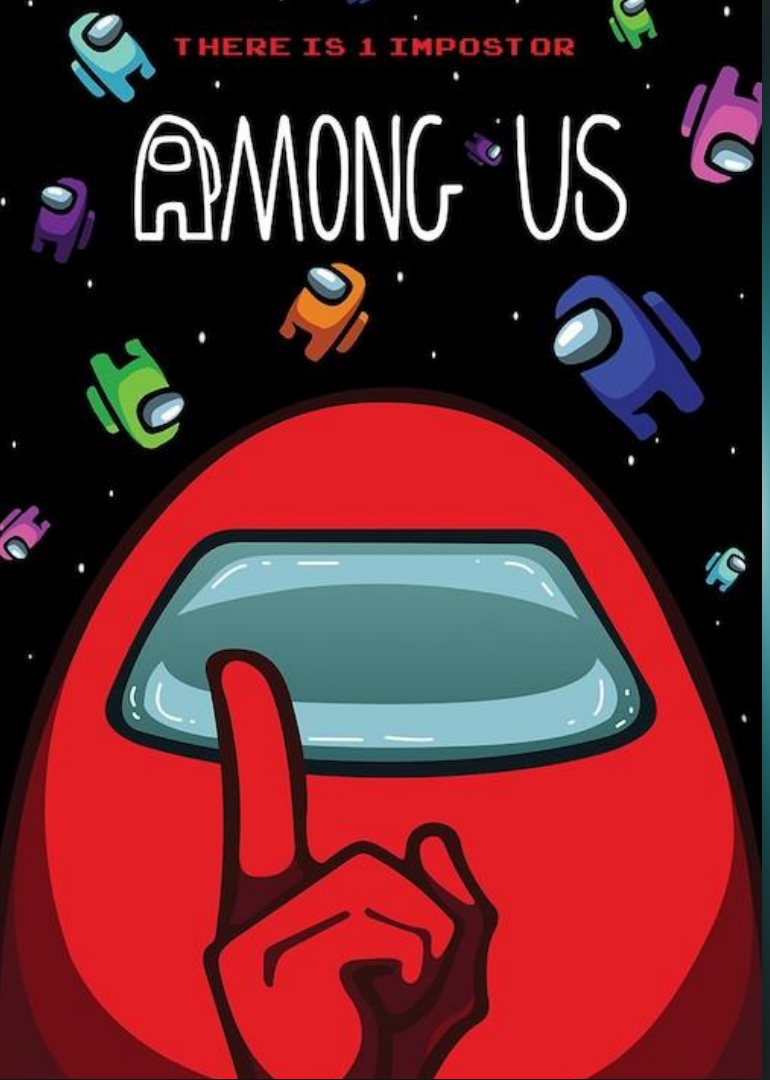
Does this app look suspicious? Report it here

Cancel | Accept


HuskyWorks Azure Tenant
eM Client app
Resource (MS Infra)

Tradecraft #2: OAuth App Attacks

“Artisanal, Hand Crafted, Farm-to-table Evil Applications”



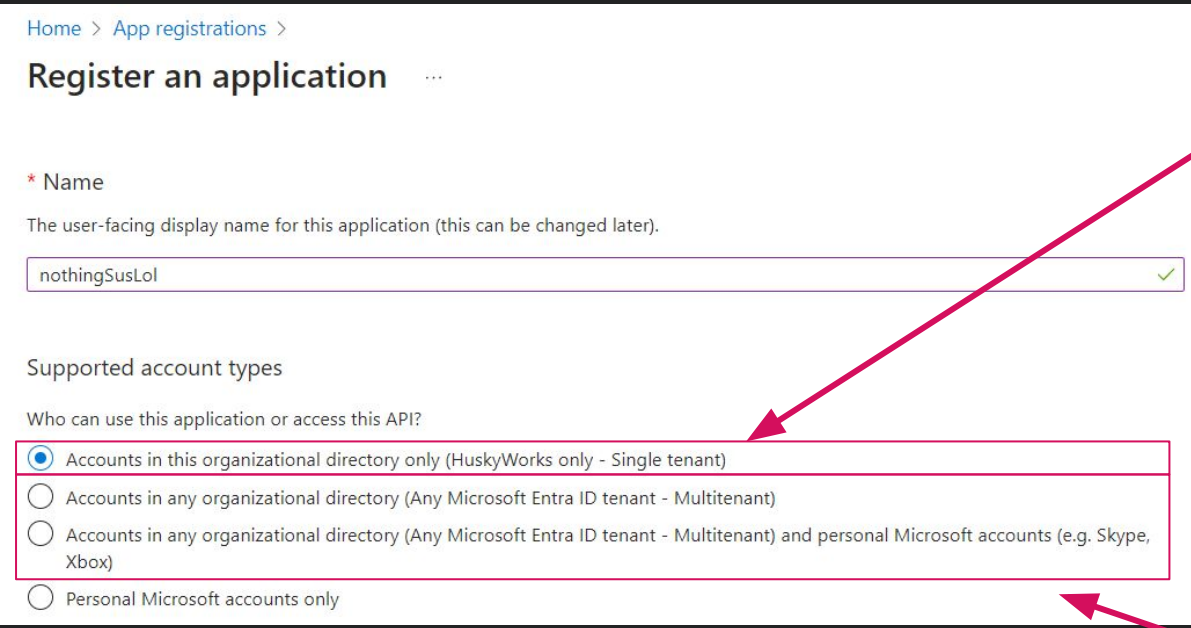
OAuth App Attacks / OAuth Consent Grant Attacks

- If Traitorware is the BYO/LOLBINs of app attacks, this is custom written malware
- Attacker creates an application specifically designed for initial access, persistence, collection, and/or attack development
- No “legitimate” use
- No two are alike:
  - Name
  - Permission scope
  - Publisher
  - etc

… wanna see how to build one?

Already have access and looking to persist?

Create it in the target tenant as a single tenant application



Using this app for initial access?

Make it in an attacker tenant and use Multitenant

🤔 What do I need as an attacker to persist/collect info/generally wreak havoc? 🤔

Request API permissions

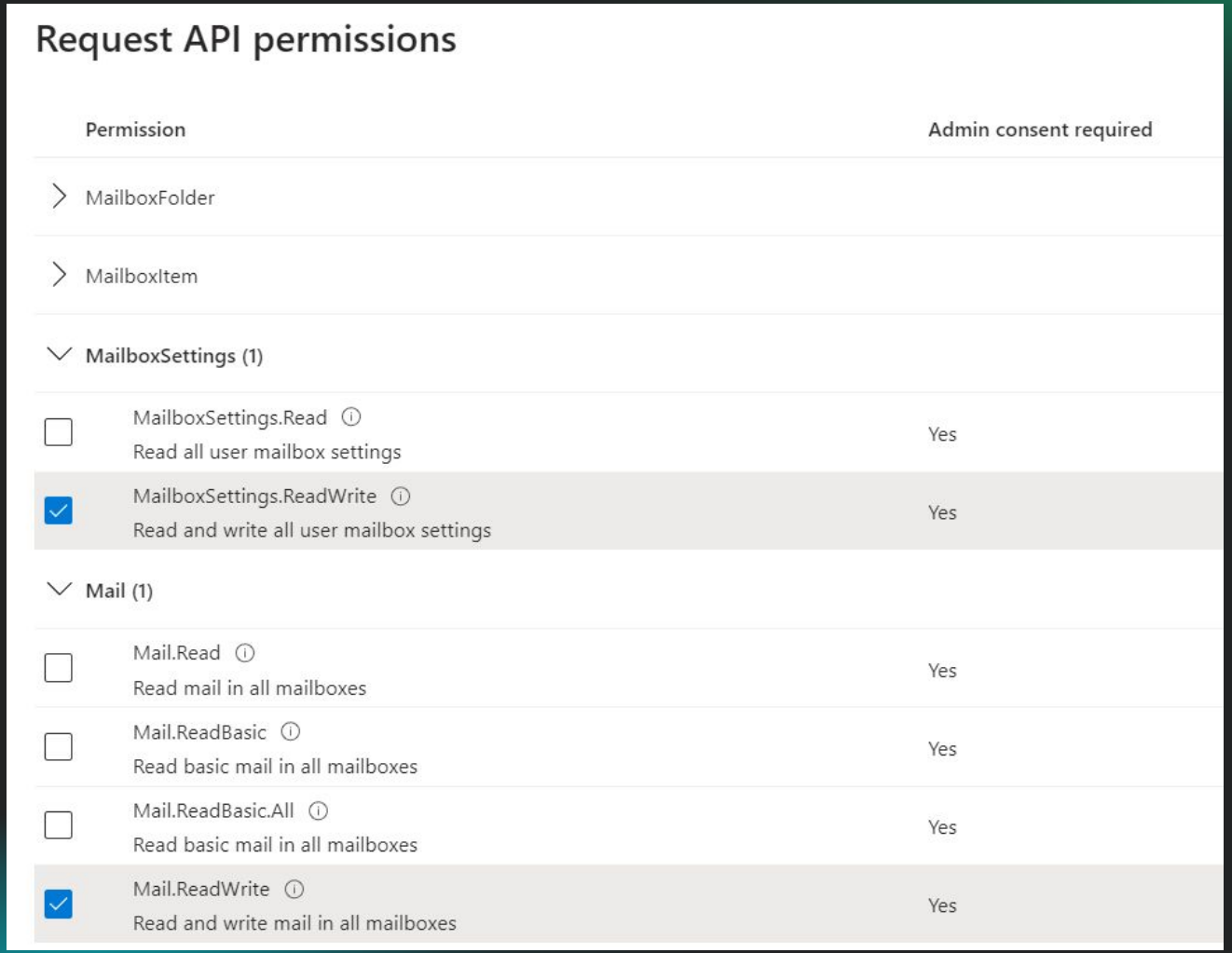
Select an API

Microsoft APIs APIs my organization uses My APIs

Commonly used Microsoft APIs

Microsoft Graph

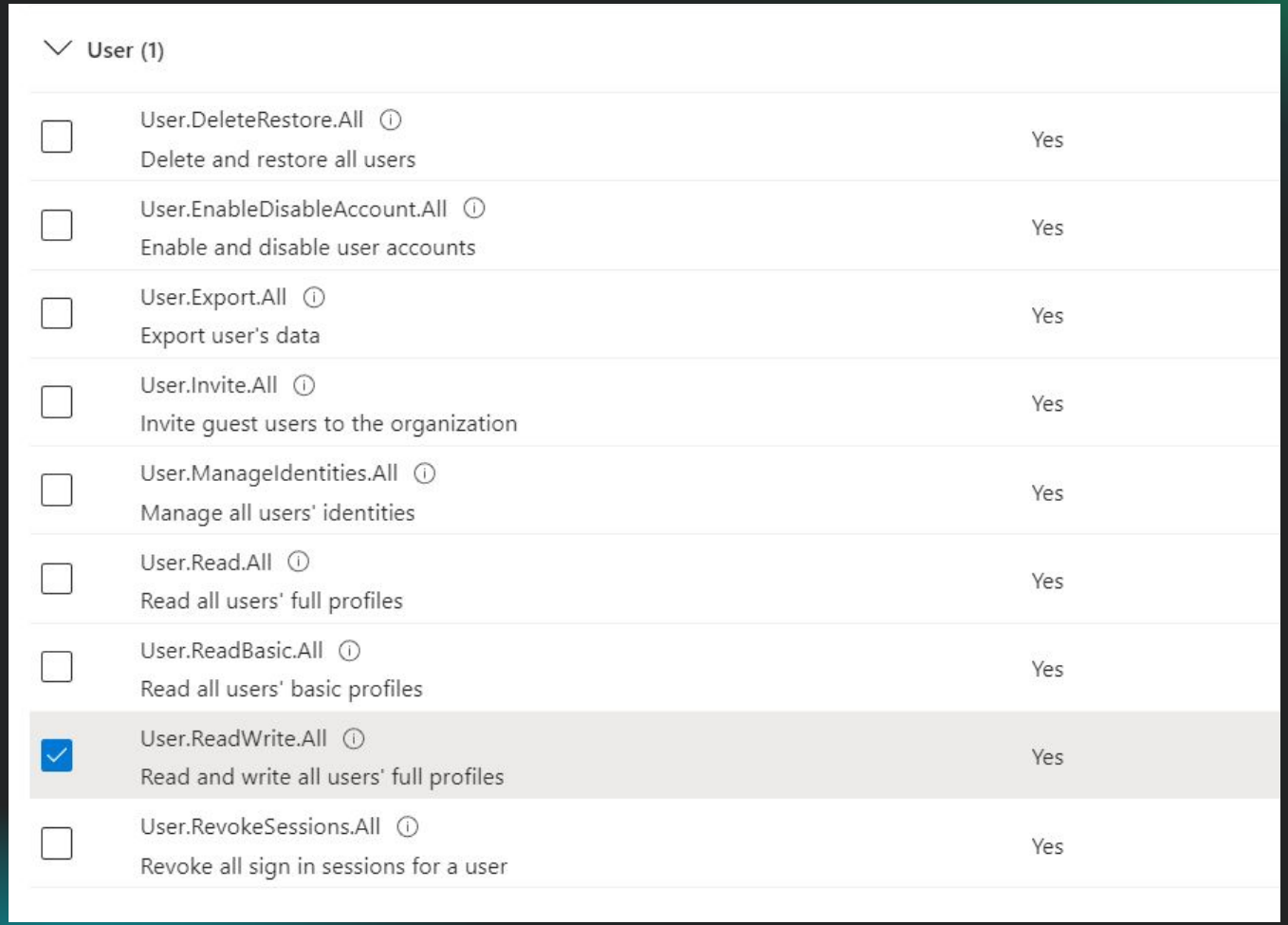
Take advantage of the tremendous amount of data in Office 365, Enterprise Mobility + Security, and Windows 10. Access Microsoft Entra ID, Excel, Intune, Outlook/Exchange, OneDrive, OneNote, SharePoint, Planner, and more through a single endpoint.



Go HAM with the permissions

Want access to/emails and inbox rules?

Done ✅



Want to be able to add, delete, and modify, user accounts in the tenant*?

Done ✅

**requires Admin consent*

Beyond Initial Access

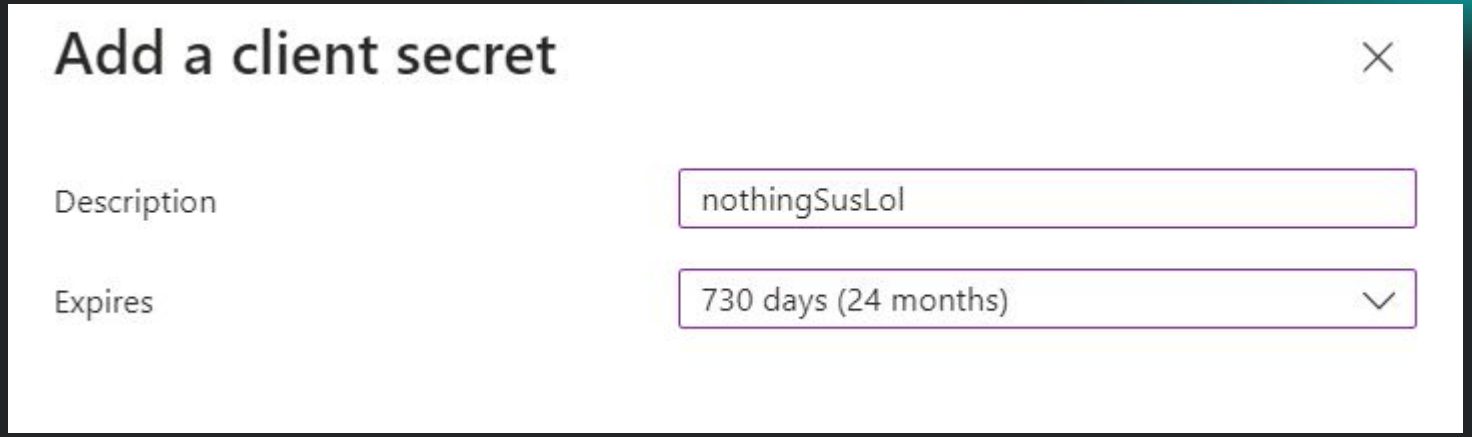
Installing the app is a great start, but how do we maintain persistence?

Set up an OAuth authentication endpoint

- When the application is installed, set up an endpoint where its access token goes
- Set up a listener
- Catch the access token!

Backdoor the service principal with a password

- Service principals are really just accounts
- Set up the service principal with a password
- Single factor, persistent access when authenticating as the service principal



Description	Expires	Value	Secret ID
nothingSusLol	9/25/2026	TkT	6fc455b7-1d20-42de-b1f8-a93abdc0fc52

Copied

This gets you single factor, persistent access as the SP

Authenticate as the service principal…

```
PS C:\Users\Matt> az login --service-principal -u "ae316cfc-9e78-4159-a5a8-da132db81be0" -p "TkT8
                  " --tenant "38a26ef0-285e-46b8-a957-d5ed1ad057d3" --allow-no-subscription
[
  {
    "cloudName": "AzureCloud",
    "id": "38a26ef0-285e-46b8-a957-d5ed1ad057d3",
    "isDefault": true,
    "name": "N/A(tenant level account)",
    "state": "Enabled",
    "tenantId": "38a26ef0-285e-46b8-a957-d5ed1ad057d3",
    "user": {
      "name": "ae316cfc-9e78-4159-a5a8-da132db81be0",
      "type": "servicePrincipal"
    }
  }
]
```

```
PS C:\Users\Matt> $headers = @{Authorization = "Bearer $token"}
PS C:\Users\Matt> $response = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/users"
-Method Get -Headers $headers -UseBasicParsing -ContentType "application/json"
PS C:\Users\Matt> $response.value


businessPhones    : {}
displayName       : christina
givenName         :
jobTitle          :
mail              :
mobilePhone       :
officeLocation    :
preferredLanguage :
surname           :
userPrincipalName : christina@HuskyWorks.onmicrosoft.com
id                : 46e87b36-14fb-4258-9047-6bd111764d38

businessPhones    : {}
displayName       : GlobalAdmin
givenName         :
jobTitle          :
mail              : GlobalAdmin@HuskyWorks.onmicrosoft.com
mobilePhone       :
officeLocation    :
preferredLanguage :
surname           :
userPrincipalName : GlobalAdmin@HuskyWorks.onmicrosoft.com
id                : df5cf0ad-dd04-4907-b039-d7d262290395
```

Now use the application's token for post-exploitation

Enumerate users…

Enumerate a user's inbox rules…

```
PS C:\Users\Matt> $response = Invoke-RestMethod -Uri "https://graph.microsoft.com/beta/users/df5cf0ad-dd04-4907-b039-d7d262290395/mailFolders/inbox/messagerules" -Method Get -Headers $headers -UseBasicParsing -ContentType "application/json"
PS C:\Users\Matt> $response.value


id          : AQAAALlXAbk=
displayName : asdasdasdasd
sequence    : 1
isEnabled   : True
hasError    : False
isReadOnly  : False
conditions  : @{fromAddresses=System.Object[]}
actions     : @{moveToFolder=AAMkAGY4NGI0NTg0LWM1ODAtNGUxNy05NzExLTMwNGEyOWJiMjIyOAAuAAAAAACB
              aOZXVXm7T4BgPCdZiDGTAQAJBVdqLA3CS4gIP6enHr32AAAAAQ6MAAA=;
              stopProcessingRules=True}
```

etc etc, you get the idea

In summary...

The SMB is getting pwned and it's *Our Problem™*

Applications used for initial access, persistence, and further development of attacks

Apps are customizable, persistent, and usually flying under the radar

Some are Among Us (evil app blending in)
Some are like a crow bar (legit tool, but used for evil)

Hunting at the SMB Scale

“Let’s level the playing field”

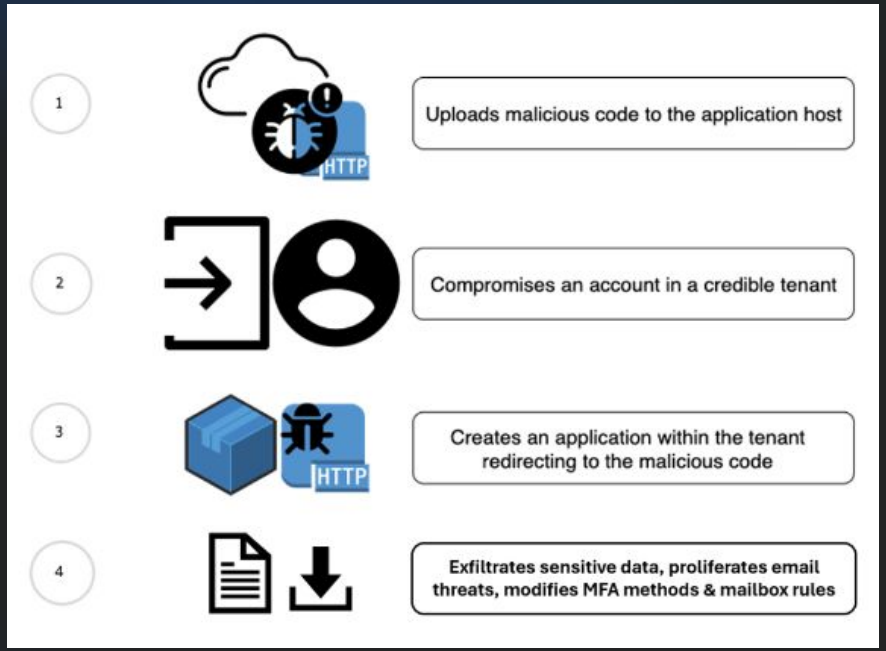
Objective

What are we looking for?

- We know what threats lurk out there in the Azure ecosystem
- ~20% of all categorized threats relate to OAuth Apps

We want to identify the following threats:

1. **Traitorware**: known app names, legitimately published, aid in attackers
2. **OAuth app attacks**: no two are alike, different scopes, different powers



Malicious Applications Created in Compromised Credible Tenants (MACT) Attacker Kill Chain can occur if attacker has the right permissions to create applications and gain persistence

HUNTRESS

Hypothesis

Answering the question: How do we find OAuth App Attacks & Traitorware?

We have a few ideas & potential red flags 🚩

- Prevalence
  - What are the most/least common apps we see?
  - Is there a common denominator in app perms and behaviors we see?
  - Unverified publisher domain – maybe? 🧐
- Application permissions/scopes
  - Which apps have more “powerful” perms?
  - Where do we see apps with these permissions?
- User persistence
  - What's the potential attack surface?
  - Which perms enable persistence to the user? How?

We will come back to this! 👀

Background

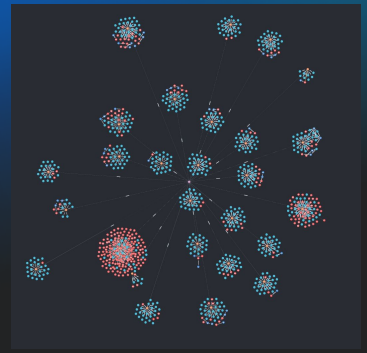
Where to do we get this data from?

Graph API v1.0

1. What raw data from tenants do we need?
    a. Users
    b. Applications
    c. Service Principals
2. Relationships
    a. Tenant >> App
    b. Tenant >> Service Principal
    c. Tenant >> Users
    d. App >> Permission Scopes / Grants
    e. Users >> Permission Scopes / Grants

Background

How can we analyze & visualize this data?



ElasticSearch/Kibana

- Easily search & filter with DSL
- ES | QL & aggregations could be helpful
- e.g., prevalence; most/least common apps seen

Neo4j

- Visually represent relationships between apps, users, service principals, permissions, etc.
- Identify unusual or uncommon relationships across nodes

Implementation / Dev Plan

How did we build this?

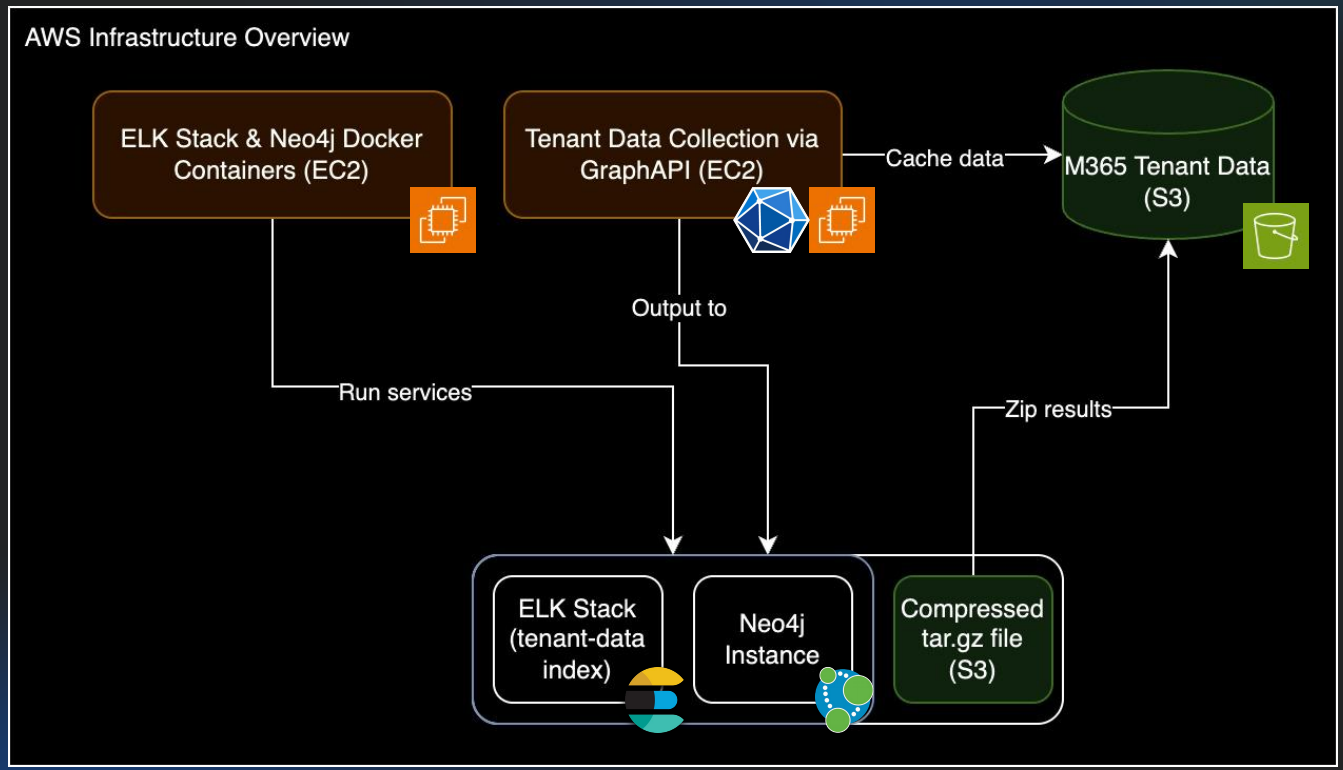
1. Get tenant data
2. Cache tenant data in S3
3. Bulk ingest data into ES
   a. ~1.8m documents, 2GB*
4. Multithreaded ingest to Neo4j
   a. ~1.8m relationships, ~569k nodes*
5. Create zip file of results and store in S3 for ad-hoc analysis
6. Analyze tenant data and relationships for *rogue apps*

Python data collection script

*Note: We ran this on ~2.5k tenants, about 10% of available Azure tenants

Implementation

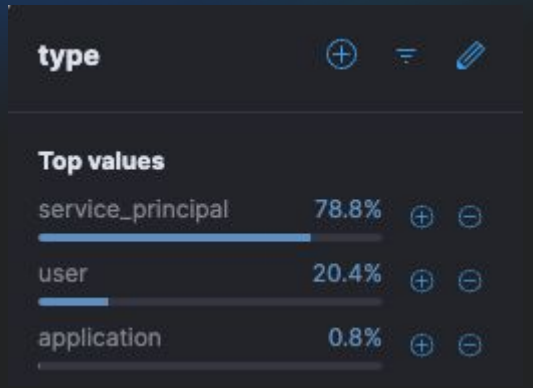
How did we build this?



Implementation

User Data Model for a Single Document

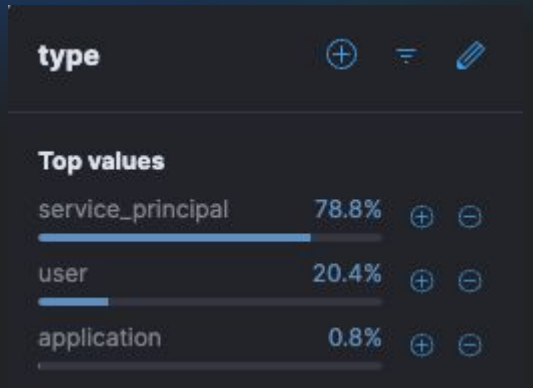

```
"_index": "tenant-data",
"_id": "alpha-num-user-id",
"_source": {
  "id": "alpha-num-user-id",
  "tenant_id": "husky-works-tenant-id",
  "type": "user",
  "attributes": {
    "user_id": "alpha-num-user-id",
    "tenant_id": "husky-works-tenant-id",
    "display_name": "Admin",
    "user_principal_name": "christina@huskyworks.onmicrosoft.com"
  },
  "ingested_at": "2024-10-14T13:17:59.775372+00:00"
},
```

Implementation

Application Data Model for a Single Document

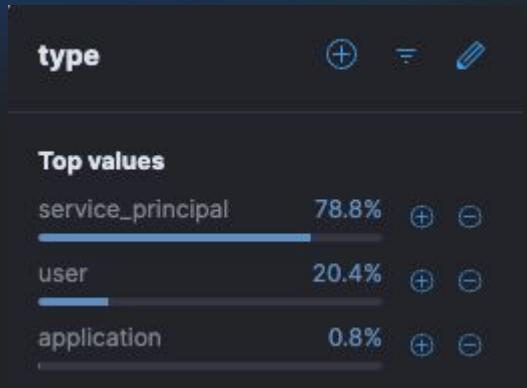

```
  "_index": "tenant-data",
  "_id": "alpha-num-app-id",
  "_source": {
    "id": "alpha-num-app-id",
    "tenant_id": "another-tenant-id",
    "type": "application",
    "attributes": {
      "id": "alpha-num-object-id",
      "app_id": "alpha-num-app-id",
      "tenant_id": "another-tenant-id",
      "display_name": "Test App for Email",
      "created_date": "2021-09-04T00:34:00",
      "publisher_domain": "huskyworks.onmicrosoft.com",
      "sign_in_audience": "AzureADMyOrg",
      "password_credentials": [...],
      "resource_accesses": [...],
      "oauth2_permission_scopes": [...]
  },
  "ingested_at": "2024-10-14T13:17:59.775372+00:00"
    },
```

Implementation

Service Principal Data Model for a Single Document


```
"_index": "tenant-data",
"_id": "alpha-num-sp-id",
"_source": {
  "id": "alpha-num-sp-id",
  "tenant_id": "another-tenant-id",
  "type": "service_principal",
  "attributes": {
    "sp_id": "alpha-num-sp-id",
    "sp_app_id": "alpha-num-app-id",
    "display_name": "Domain Controller Services",
    "created_date": "2024-010-06T13:10:52",
    "deleted_date": null,
    "sign_in_audience": "AzureADMultipleOrgs",
    "tenant_id": "another-tenant-id",
    "oauth2_permission_grants": [...],
    "app_role_assigned_to": [...],
    "oauth2_permission_scopes": [...]
  },
  "ingested_at": "2024-10-14T13:17:59.775372+00:00"
},
```

What rogue apps did we find?

Findings

Traitorware

- Across 2500 Huntress partner tenants, about 270 (~10%) of those tenants had at least 1 rogue app, which has been commonly observed during Azure identity attacks 😬
- In some cases, 1 identity from a tenant actually had many applications, and some have been lurking in Azure environments for some time.
  - We did notify partners where we noticed these rogue apps 😌
- Needle in the haystack for determining sus permissions/scopes & user persistence 🤔

Findings

Prevalence of Service Principals Overview

Prevalence Bracket	Number of Applications	Number of Service Principals
90-100%	0	159
80-90%	0	77
70-80%	0	73
60-70%	1	64
50-60%	0	32
40-50%	0	28
30-40%	0	32
20-30%	0	29
10-20%	1	136
5-10%	2	120
1-5%	38	459
>1%	6192	12904

These tables were generated from summarizer script we had, based on latest compressed file of all tenants

Findings

Prevalence of Service Principals Overview

Prevalence Bracket	Number of Applications	Number of Service Principals
90-100%	**0**	**159**
80-90%	0	77
70-80%	0	73
60-70%	1	64
50-60%	0	32
40-50%	0	28
30-40%	0	32
20-30%	0	29
10-20%	1	136
5-10%	2	120
1-5%	38	459
<1%	**6192**	**12904**

	Metric	Value
	Total Tenants Analyzed	2525
	Total Applications Analyzed	6234
	Total Service Principals Analyzed	14113
	Average Applications per Tenant	2.47
	Average Service Principals per Tenant	5.59

What does this tell us?

- A tenant with a lot of SPs is quite common
- However, there are *significantly* more SPs and applications unique to each tenant

Findings

Prevalence of Service Principals Overview

Prevalence Bracket	Number of Applications	Number of Service Principals
90-100%	0	159
80-90%	0	77
70-80%	0	73
60-70%	1	64
50-60%	0	32
40-50%	0	28
30-40%	0	32
20-30%	0	29
10-20%	1	136
5-10%	2	120
1-5%	38	459
<1%	**6192**	**12904**

Metric	Value
Total Tenants Analyzed	2525
Total Applications Analyzed	**6234**
Total Service Principals Analyzed	**14113**
Average Applications per Tenant	2.47
Average Service Principals per Tenant	5.59

What does this tell us?

∴ Vast majority of SPs and apps fall into this <1% prevalence bracket

- 99% of apps
- >91% of service principals

HUNTRESS

Findings

Most Common M365 Apps by Tenant

1. Microsoft To-Do
2. SharePoint Notification Service
3. Media Analysis and Transformation
4. Microsoft Teams Web Client
5. P2P Server
6. Microsoft Graph
7. O365 Secure Score
8. Microsoft Approval Management
9. Signup
10. AADReporting

… and Huntress and more Azure-based apps 😉

This list is determined by aggregating App Display Name by unique tenant in ES

Findings

Prevalence ✅

- Prevalence across the tenant data set can at least inform how we proceed in narrowing down the unknown-unknowns
- Rarity: identify most/least common apps
  - Most common: largely Microsoft related, not sus
  - Least common: YMMV, but helpful for querying sus sounding app names

Findings

OAuth App Attacks

Metric	Value
Total Tenants Analyzed	2525
Total Tenants with Rare Entries	1240
Total Rare Applications and SPs	10947
Total Applications Analyzed	13640
Total Service Principals Analyzed	**1086095**
Total Rare Applications Identified	0
Total Rare Service Principals Identified	**10947**
Total with Reason 'BroadImpact'	6697
Total with Reason 'Phishing'	2503
Total with Reason 'Collection'	1559
Total with Reason 'PrivEscalation'	188

This allows us to shrink the original haystack from over 1mil SPs to 1% of SPs.

We could further analyze & review this subset further 🧐

Findings

OAuth App Attacks

- Find apps & SPs with the following characteristics:
  - Global prevalence of less than 1% in the surveyed tenants…
  - … that contain at least one example of an OAuth scope that can be abused
    - Example 1:
      - "MailboxSettings": {"Privilege": "High", "Reason": "Phishing"},
      - "User.ReadWrite.All": {"Privilege": "High", "Reason": "BroadImpact"},
      - “Application.ReadWrite.All”: {“High”, “BroadImpact”}
      - … where those scopes are granted for AllPrincipals in the tenant
    - Example 2:
      - SP permission grants/scopes containing Directory.ReadWrite.All
        - Global tenant admin perms

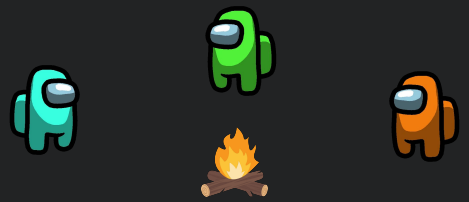
Findings

OAuth App Attacks

Let’s dive further into some examples:

Name	Type	Prev %	Publisher	Permissions	Reason
IT Glue Full Integration	Service Principal	0.0	Unknown Publisher	Directory.ReadWrite.All	BroadImpact
couchdrop	Service Principal	0.5	Unknown Publisher	Files.Read.All	Collection
Backup Application [REDACTED]	Service Principal	0.0	Unknown Publisher	Directory.ReadWrite.All	BroadImpact
WindowsAdminCenter-https://localhost:6516	Service Principal	0.6	Unknown Publisher	Directory.ReadWrite.All	BroadImpact
rclone	Service Principal	0.5	Unknown Publisher	Files.Read.All	Collection

Collection + Files.Read.All perms = SUS
Directory.ReadWrite.All = SUS
Unknown Publisher = maybe SUS



What does this tell us?

These could *potentially* be malicious, but many are hiding out in the wild

Findings

Based on our original hypothesis, did we find instances of OAuth App Attacks & Traitorware?

- Prevalence ✅
  - Identified most common apps and SPs in M365
  - Assigned prevalence % for more common apps
  - Determined reputable publishers
- Application permissions/scopes ✅
  - Identified apps & SPs with powerful permissions
  - Identified patterns of permission scopes/grants for potentially malicious apps
- User persistence ✅
  - Users have potential to establish global admin perms, dangerous if in the wrong hands

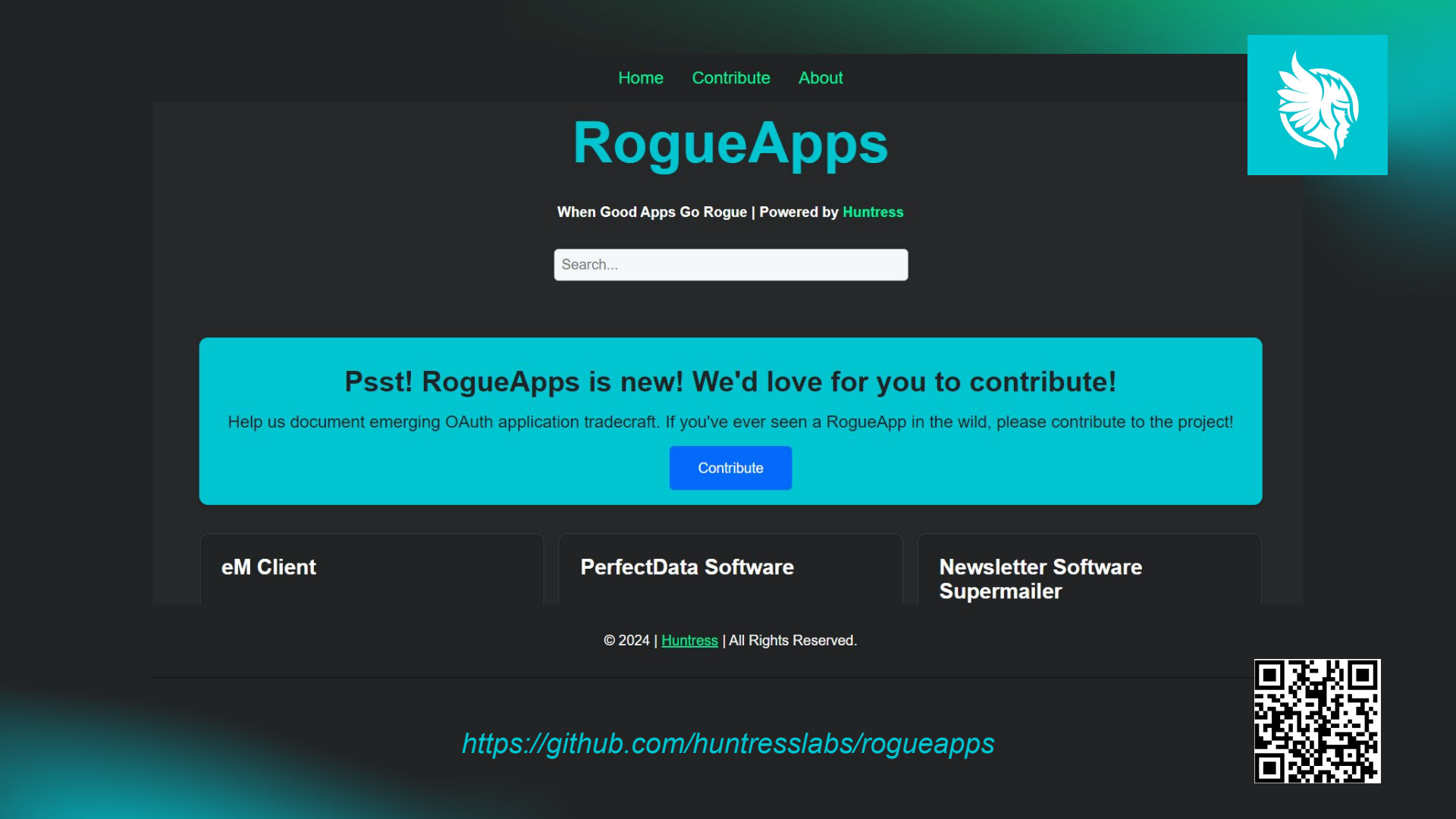
Recap

TL;DR

1. It is not a one-size fits all approach for finding rogue apps in SMB
   a. Apps are widely customized and unique to each tenant
2. Needle in the haystack problem, and we can tease out apps that *appear to be malicious*
3. We can use app and SP prevalence as a high-level indicator
4. Permissions/scopes:
   a. Could restrict app perms for users; principle of least privilege
5. User persistence:
   a. Could use risk scoring, such as MITRE TTPs

💫 **But wait, there's more! ~**

What should you do if you see rogue apps in the wild?



https://github.com/huntresslabs/rogueapps

Limits/Constraints & Future Work

Limits/Constraints:

- Time
  - Good for first iteration, but scope of this project encompasses roughly 10% of Azure data we have visibility into
- ETL Bottlenecks
  - Took 1 week to load the data we have from Graph API
  - Didn't load all data into Neo4j 🥺

Looking Ahead:

- Data collection / ETL optimizations
  - Graph API (asynchronous functions)
  - Neo4j (multithread vs bulk ingestion)
- Does existing data model scale for analysis? 🤔
- Stay tuned for a v2.0! 🥁

Final Takeaways

How can we safeguard against rogue apps?

- MFA / 2FA:
  - Make it more difficult for threat actors to get credentials
- No shadow IT
  - Normal users shouldn't be allowed to add new apps
  - Proactively, admins can manage user content to apps in M365
- Reduce attack surface:
  - Audit all apps on routine basis
  - Remove unused apps
  - Do not over-permission apps
  - Check app for verified publisher & from reputable source
- Look out for the 99%: Share what you know with the SMB community at-large
  - Increasing but but limited research exists for M365
  - Also check out **Rogue Apps**

Acknowledgements

Special thanks to…

⭐ **Dave Kleinatland** for getting us started with app credentials & starter code!

⭐ **Sharon Martin** for writing the initial Huntress blog post on *Legitimate Apps as Traitorware for Persistent Microsoft 365 Compromise*

Thank you!

HUNTRESS

Questions?

Feel free to reach out if any further questions!

Christina Parry
christinaparry.com | @CyberCorg
christina.parry@huntresslabs.com

Matt Kiely
@HuskyHacksMK
matt.kiely@huntresslabs.com

References

Abitan. Gif. Giphy, https://giphy.com/abitan. Accessed 17 October 2024.

Friedman, Assaf, and Itir Clarke. "How Attackers Use Compromised Accounts to Create and Distribute Malicious OAuth Apps." Proofpoint Blog, 5 May 2021, https://www.proofpoint.com/us/blog/email-and-cloud-threats/how-attackers-use-compromised-accounts-create-and-distribute-malicious.

Friedman, Assaf, David Krispin, and Eilon Bendet. "The Dangerous Consequences of Threat Actors Abusing Microsoft's 'Verified Publisher' Status." Proofpoint Blog, 2023, https://www.proofpoint.com/uk/blog/cloud-security/dangerous-consequences-threat-actors-abusing-microsofts-verified-publisher.

Martin, Sharon. "Legitimate Apps as Traitorware for Persistent Microsoft 365 Compromise." Huntress Blog, 2023, https://www.huntress.com/blog/legitimate-apps-as-traitorware-for-persistent-microsoft-365-compromise.

Microsoft. "Incident Response Playbook for Application Consent." Microsoft Learn, 2024, https://learn.microsoft.com/en-us/security/operations/incident-response-playbook-app-consent.

Microsoft. "User Consent in Microsoft 365." Microsoft Learn, 2024, https://learn.microsoft.com/en-us/microsoft-365/admin/misc/user-consent?view=o365-worldwide.

Microsoft Identity Tools. "Azure AD Consent Grant Permissions Table." GitHub, 2023, https://github.com/AzureAD/MSIdentityTools/blob/main/assets/aadconsentgrantpermissiontable.csv.

Robbins, Andy. "Directory.ReadWrite.All Is Not As Powerful As You Might Think." Posts By SpecterOps Team Members, 12 Feb. 2024, posts.specterops.io/directory-readwrite-all-is-not-as-powerful-as-you-might-think-c5b09a8f78a8.

Savill, John. "Azure AD App Registrations, Enterprise Apps and Service Principals." John Savill's Technical Training, YouTube, 2023, https://youtu.be/WVNvoiA_ktw?si=znKVzZicDnZEF5ar.

Strom, Blake, et al. "Use Alternate Authentication Material: Application Access Token." MITRE ATT&CK, 30 Jan. 2020, last modified 28 Apr. 2024, https://attack.mitre.org/techniques/T1550/001/.

Swartz, Nir, Eilon Bendet, and the Proofpoint Threat Research Team. "Revisiting MACT: Malicious Applications in Credible Cloud Tenants." Proofpoint Blog, 2022, https://www.proofpoint.com/us/blog/cloud-security/revisiting-mact-malicious-applications-credible-cloud-tenants.