Authomize

GETTING CLEAR TEXT PASSWORDS FROM AN IDP & MORE: OUR RESEARCH METHODOLOGY

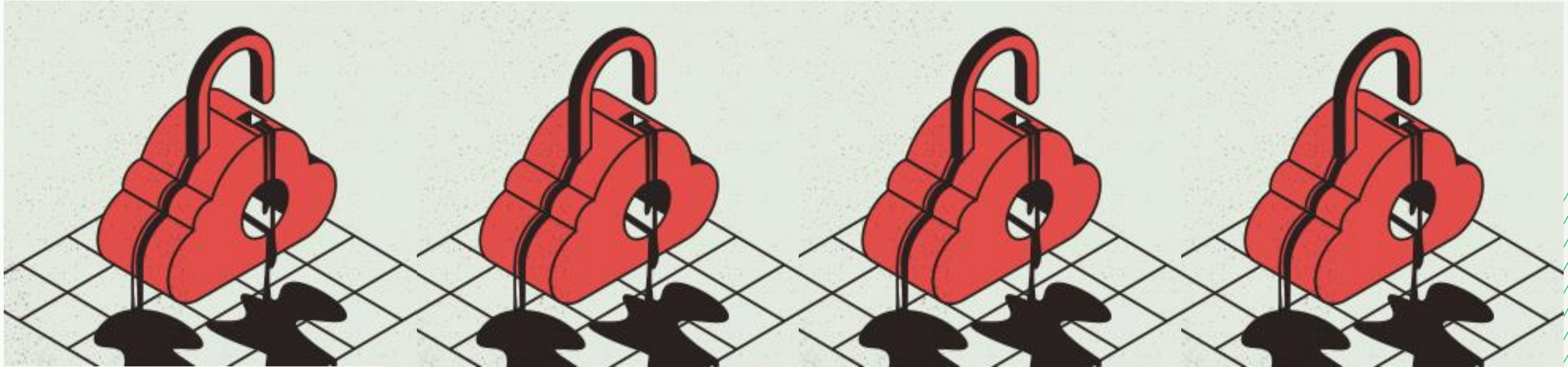
Gal Diskin / CTO & Co-Founder, Authomize
@gal_diskin on Twitter
https://diskin.org (personal blog)

Agenda

- Background
  - #PassBleed, Orange book, TCB, Graphs & knowledge graphs (KGs)
- Research process that led to #PassBleed issues
  - Bleeding passwords via SCIM
  - Impersonating anyone in the hub
- Summary
  - Write your own logic vuln detection suite in less than 20 lines of Python
  - Stop the FUD - public reaction to #PassBleed
  - What we've learned and where to go from here

Intro

- Recently my team found a set of security risks in the popular Okta platform dubbed #PassBleed, which were not recognized as vulnerabilities
- The main #PassBleed issues allow less privileged attackers in Okta to gain access to super admin privileges
- In this talk, I will walk you through the research process we went through
- I will touch on a systematic approach to discovering certain classes of logic bugs

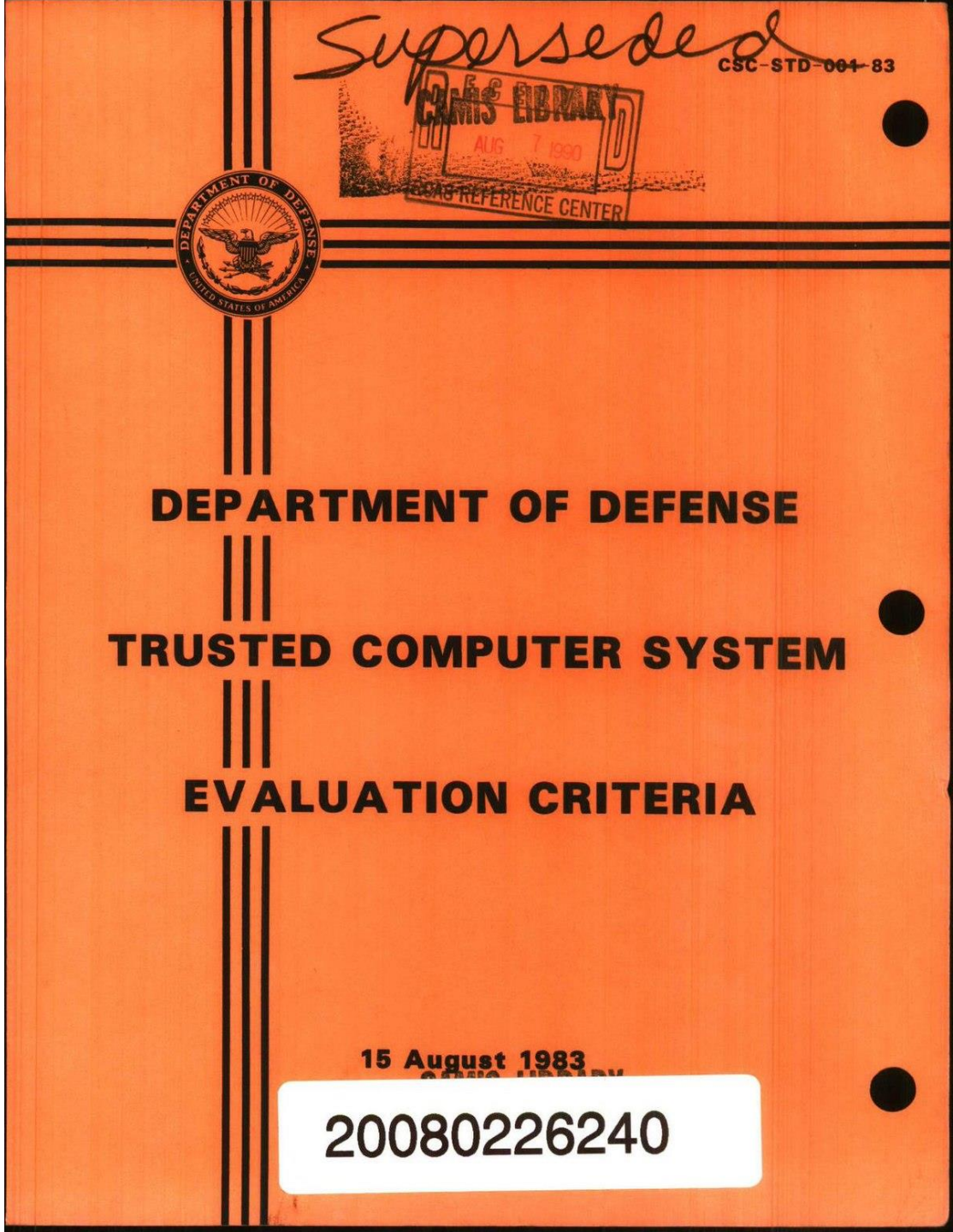


Trusted Computing Base

Per the ***Orange Book:***

The heart of a trusted computer system is the Trusted Computing Base (TCB) which contains all of the elements of the system responsible for supporting the security policy and supporting the isolation of objects (code and data) on which the protection is based. [.....]

Thus, the TCB includes hardware, firmware, and software critical to protection and must be designed and implemented such that ***system elements excluded from it need not be trusted to maintain protection****.*



Authomize

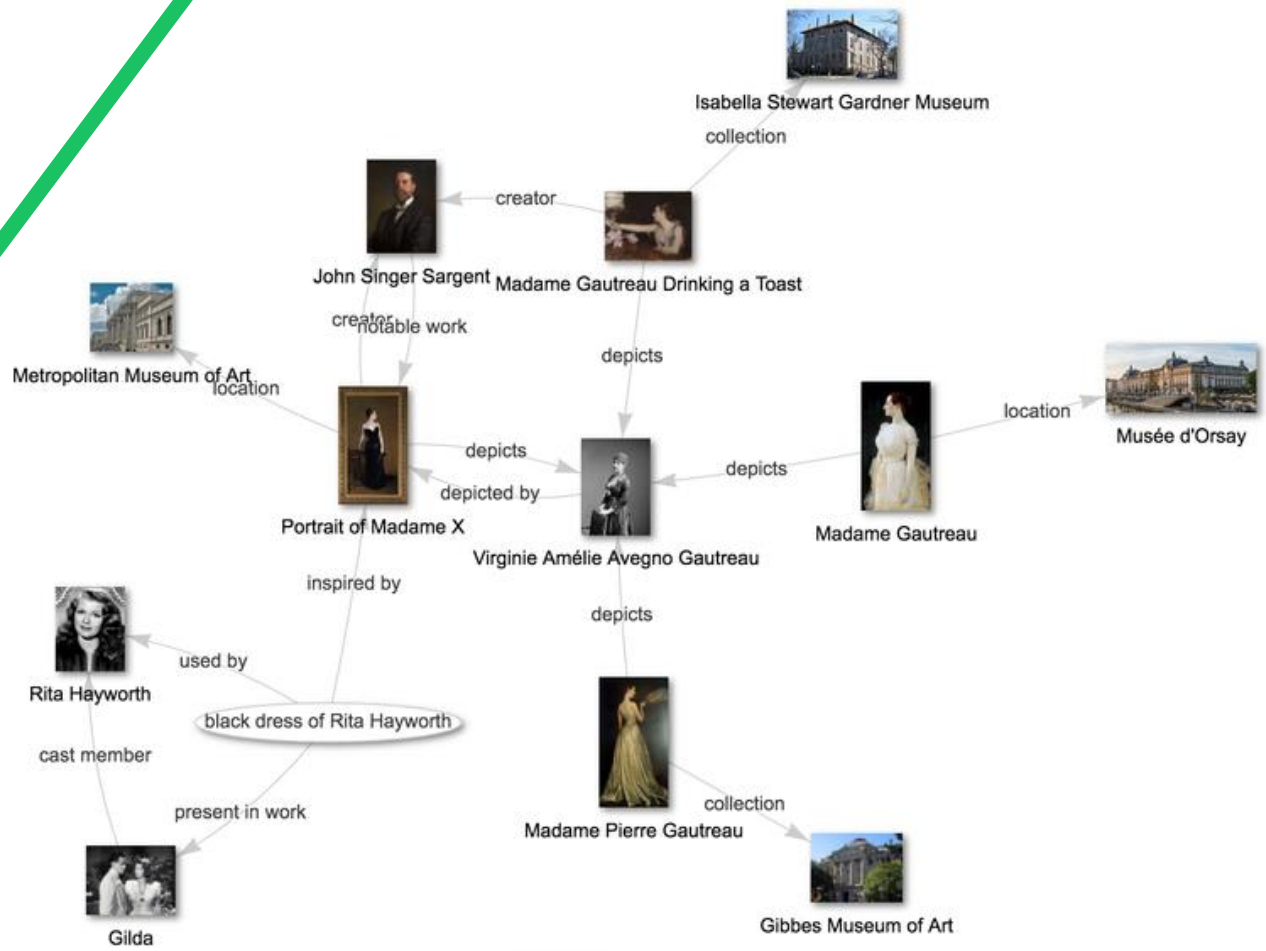
https://commons.wikimedia.org/wiki/File:Trusted_Computer_System_Evaluation_Criteria_CSC-STD-001-83.pdf

Graphs networks in security

- What are Knowledge Graphs (KGs)?

- KGs applications to cybersecurity
  - Maltego – social and even attack surface mapping
  - “Attackers think in graphs”
  - Bloodhound
  - CIEM
  - Attack simulation
  - …

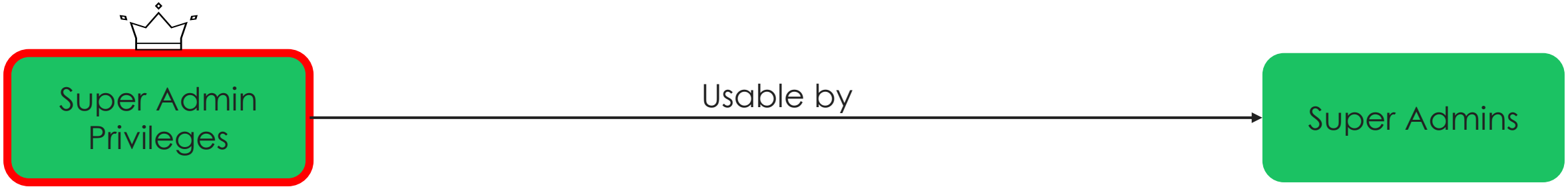
Sample Knowledge Graph

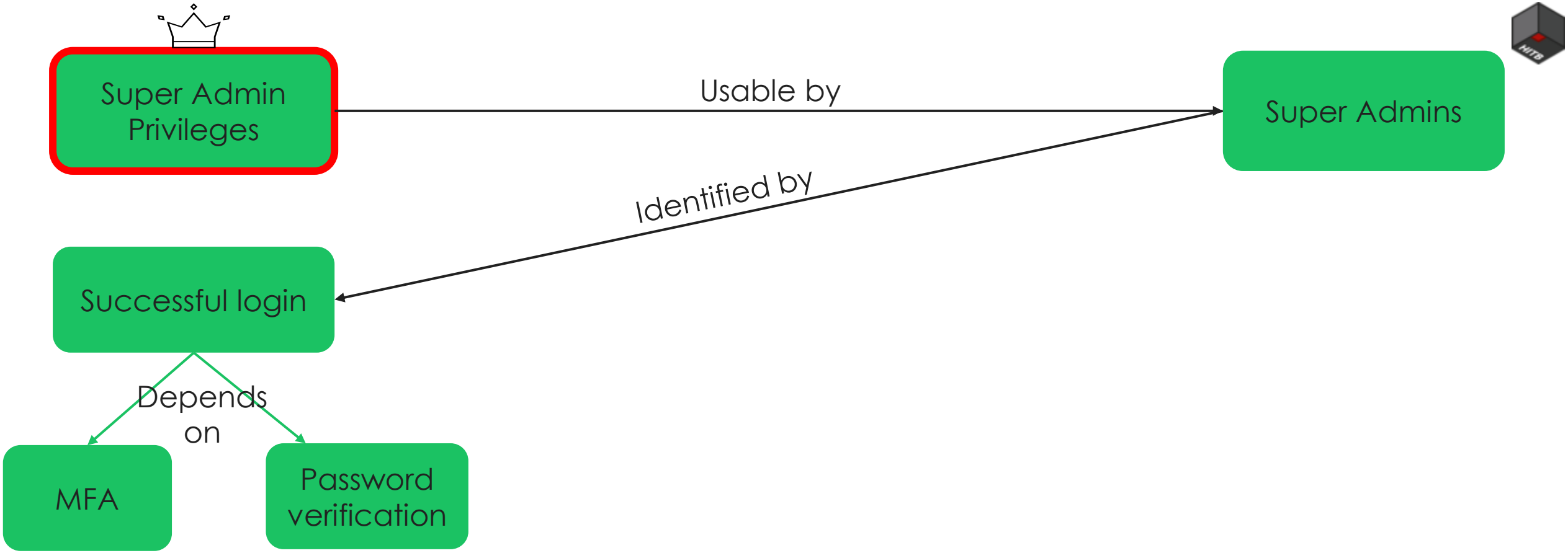


https://commons.wikimedia.org/wiki/File:Wikidata-knowledge-graph-madame-x-2019.png

Graphs as maps of trust

- Knowledge Graphs can be used to map trust to systematically find logic vulns
- How?
  - Nodes = assets, privileges, or actors
  - Edges = trust relations
  - Our “innovation” - mistrust edges
  - Any cycle with a mistrust edge is a vuln
- Why?
  - Collaboration in teams
  - Knowledge preservation
  - Visualization for manual work & prioritization
  - Automation – faster and prevents human error


Super Admin Privileges
Usable by
Super Admins


Super Admin Privileges
Usable by
Super Admins
Identified by
Successful login
Depends on
MFA
Password verification

Before we continue – a short explanation

Access to something is the opposite direction of trust, because the "accessed" entity must trust the "accessing" entity

Bleeding passwords via SCIM

What is SCIM?

- Identity provisioning protocol
- Automate sync of user and group data
- Okta extended it to support Password sync

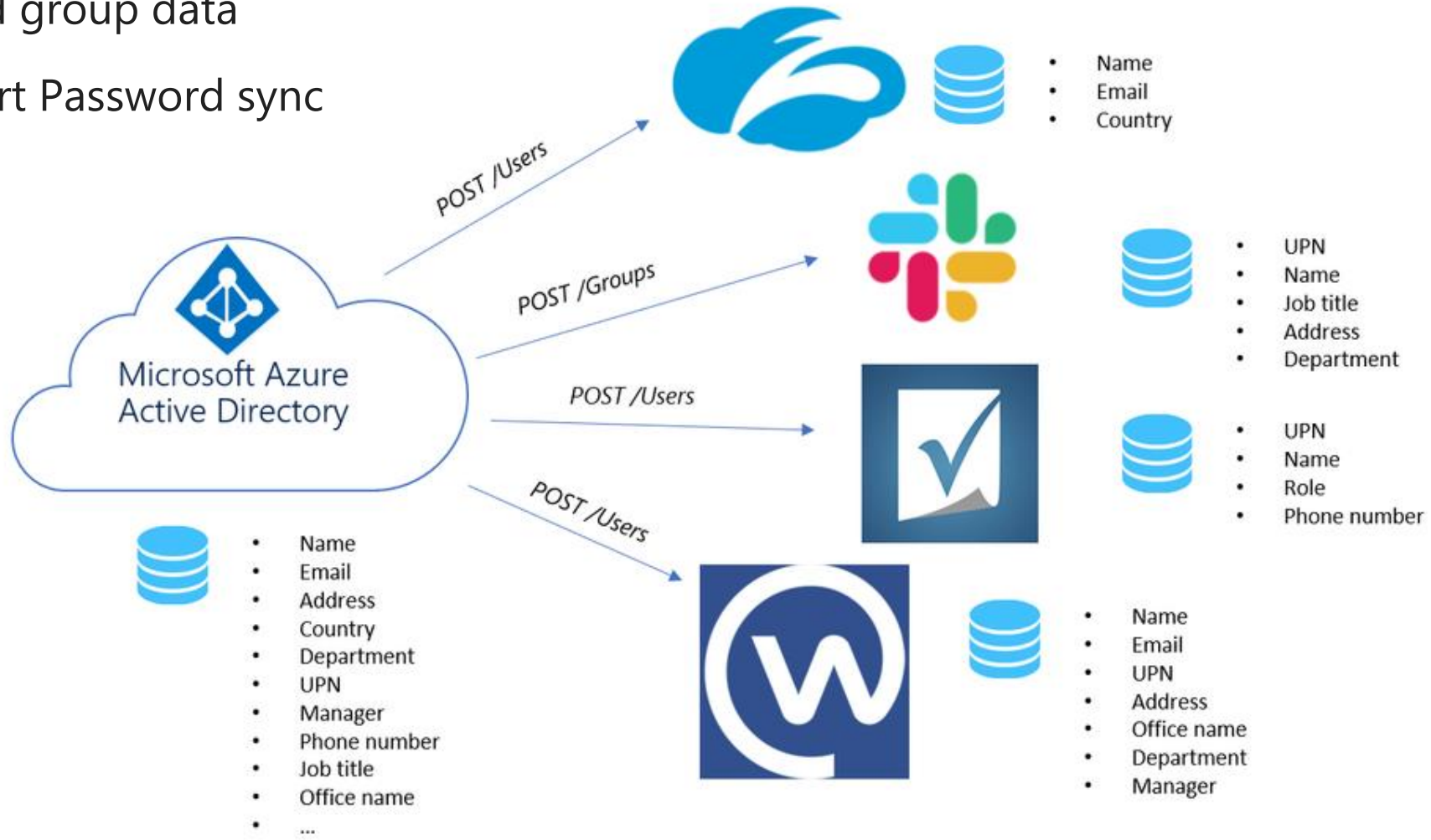
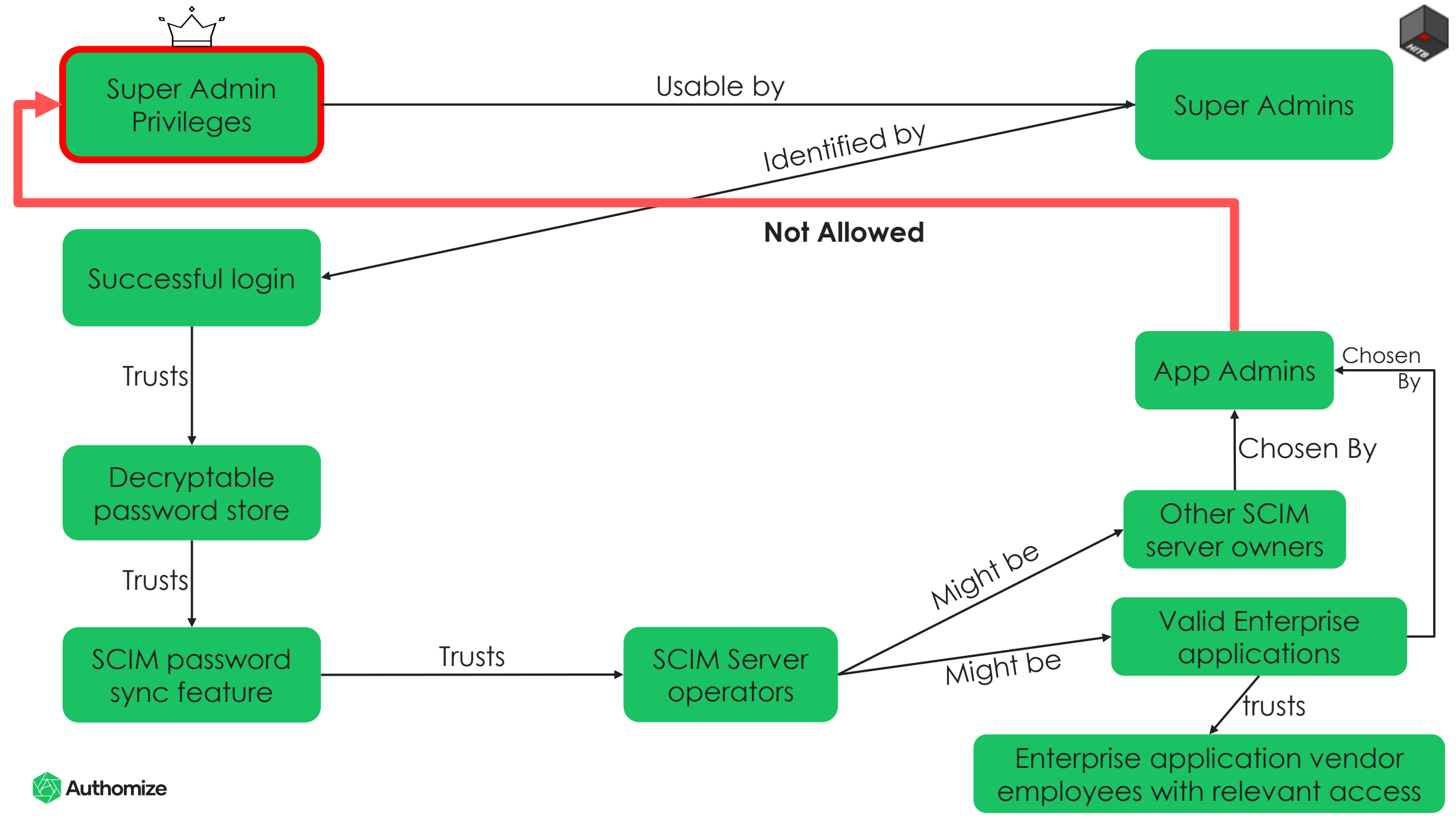


Image Source: Microsoft Documentation


Super Admin Privileges
Usable by
Super Admins
Identified by
Not Allowed
Successful login
Trusts
Decryptable password store
Trusts
SCIM password sync feature
Trusts
SCIM Server operators
Might be
Might be
Other SCIM server owners
Valid Enterprise applications
App Admins
Chosen By
Chosen By
trusts
Enterprise application vendor employees with relevant access
Authomize

Validation phase 1 – SCIM PrivEsc

"We made a mistake somewhere, there is no way an app admin can access all passwords including super admin passwords. We're missing something, let's test again"

… [TESTING] → we got the cleartext password of super admin …

"Ok, probably app admins are very highly privileged or rare"

"Let's check frequency at our customers"

"Some customers have 10s to 100s of app admins"

"Let's try to understand why"

[Several days later] → Conclusion: this is the lowest built-in role that can assign users the permission to use apps

Permission	Super Admin	Org Admin	Group Admin	App Admin	Read-Only Admin	Mobile Admin	Help Desk Admin	Report Admin	API Access Management Admin	Group Membership Admin
View applications or application instances	•			•^	•	•			•*	
Add and configure applications	•			•^					•*	
Assign user access to applications	•			•^					•*	
Create users in staged status via app import	•			•^						

Authomize

Some technicalities – JSON payload

- Example JSON payload sent by Okta to (any) SCIM server, taken from Okta documentation
- I like the self humor password ☺

PII

CLEAR TEXT PASSWORD

```
{
  "schemas": [
    "urn:scim:schemas:core:1.0"
  ],
  "userName": "myemail-pending@example.com",
  "emails": [
    {
      "primary": true,
      "value": "myemail-pending@example.com",
      "type": "primary"
    },
    {
      "primary": false,
      "value": "mypersonalemail-pending@example.com",
      "type": "secondary"
    }
  ],
  "phoneNumbers": [
    {
      "value": "123-444-5555",
      "type": "mobile"
    }
  ],
  "name": {
    "familyName": "LastName-pending",
    "givenName": "FirstName-pending"
  },
  "active": false,
  "password": "verySecure",
  "groups": [
    {
      "display": "secondGroup",
      "value": "1002"
    }
  ]
}
```

https://help.okta.com/en-us/Content/Topics/Provisioning/opp/OPP-provision-SCIM-messages.htm

SCIM based attack scenarios to steal Passwords and PII

1. App Admin insider → Bribed by initial access broker (IAB) → Exfiltrates passwords → $$$
2. Attacker compromises one of 10s of App Admins → Exfiltrates super-admin / automated account password → Logs in using password, may need to beat MFA → Privilege Escalation success!
3. Naïve App Admin configures SCIM to HTTP → Attacker sniffs network traffic → Finds passwords and PII → $$$

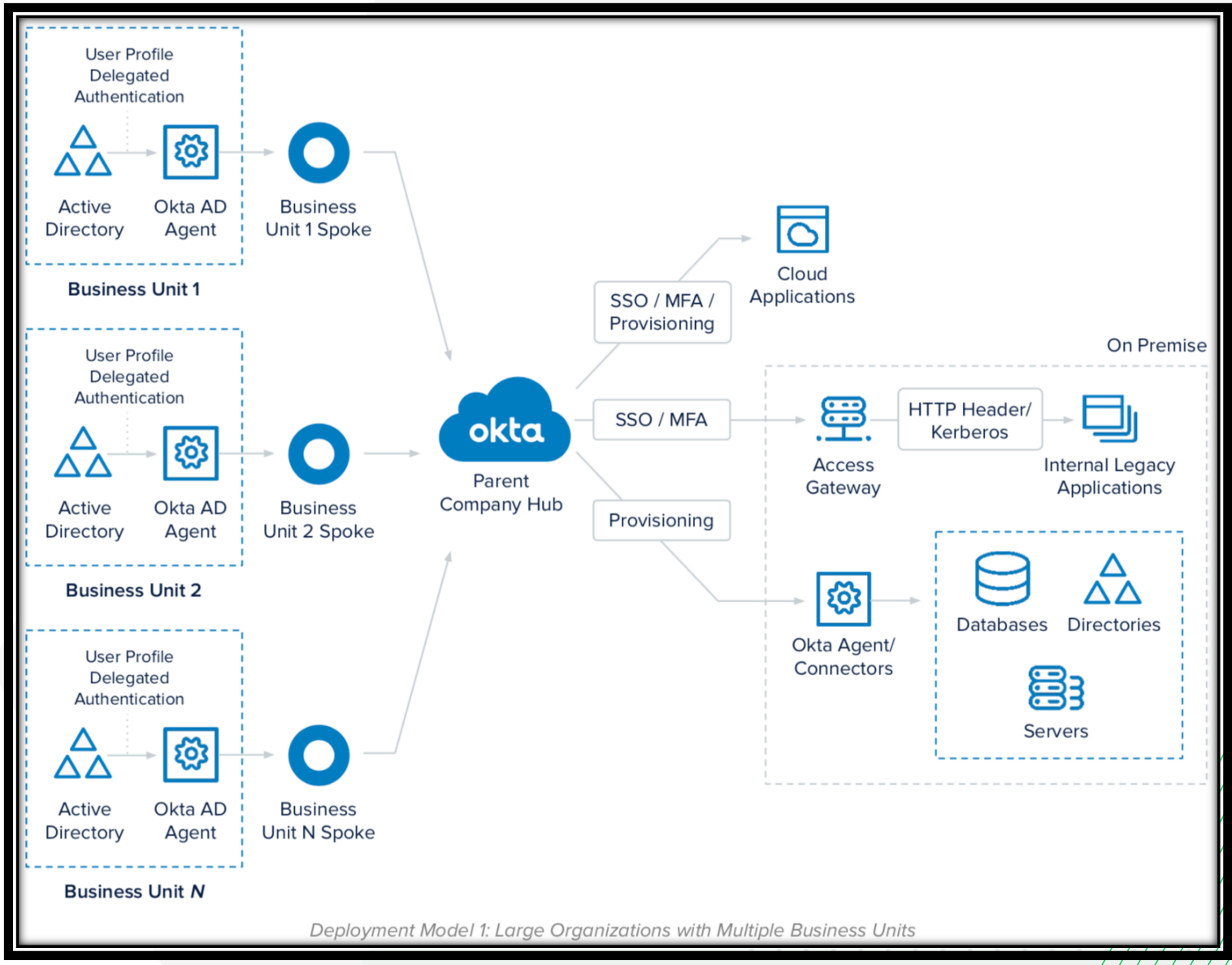
Impersonating anyone in the hub

Hub & Spoke architecture explained

- Deployment model 1: Large Organizations with Multiple Business Units

- Each BU gets a spoke, all spokes are connected as IdPs to the main company acting as the hub

Source: Okta documentation



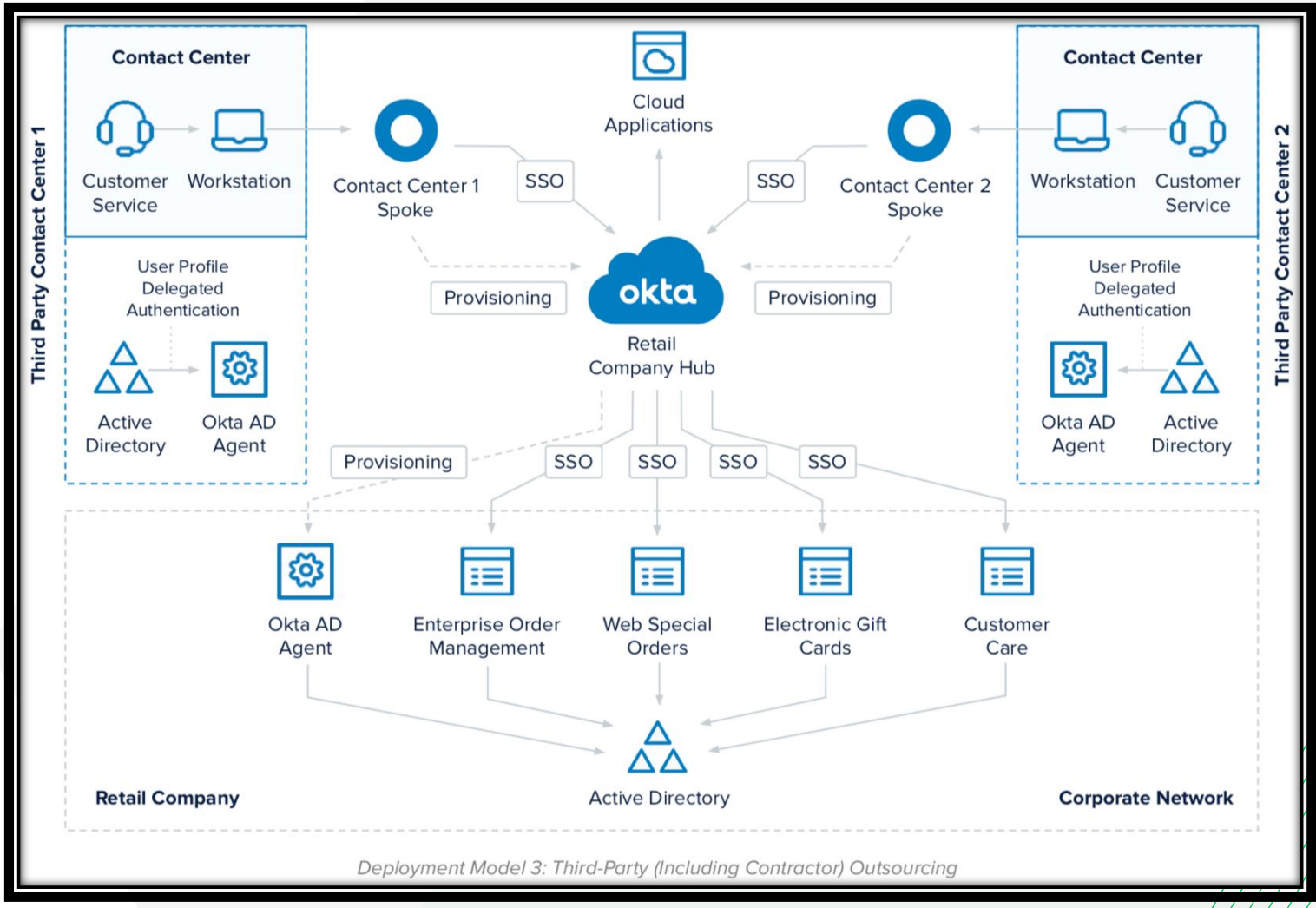
Hub & Spoke architecture explained

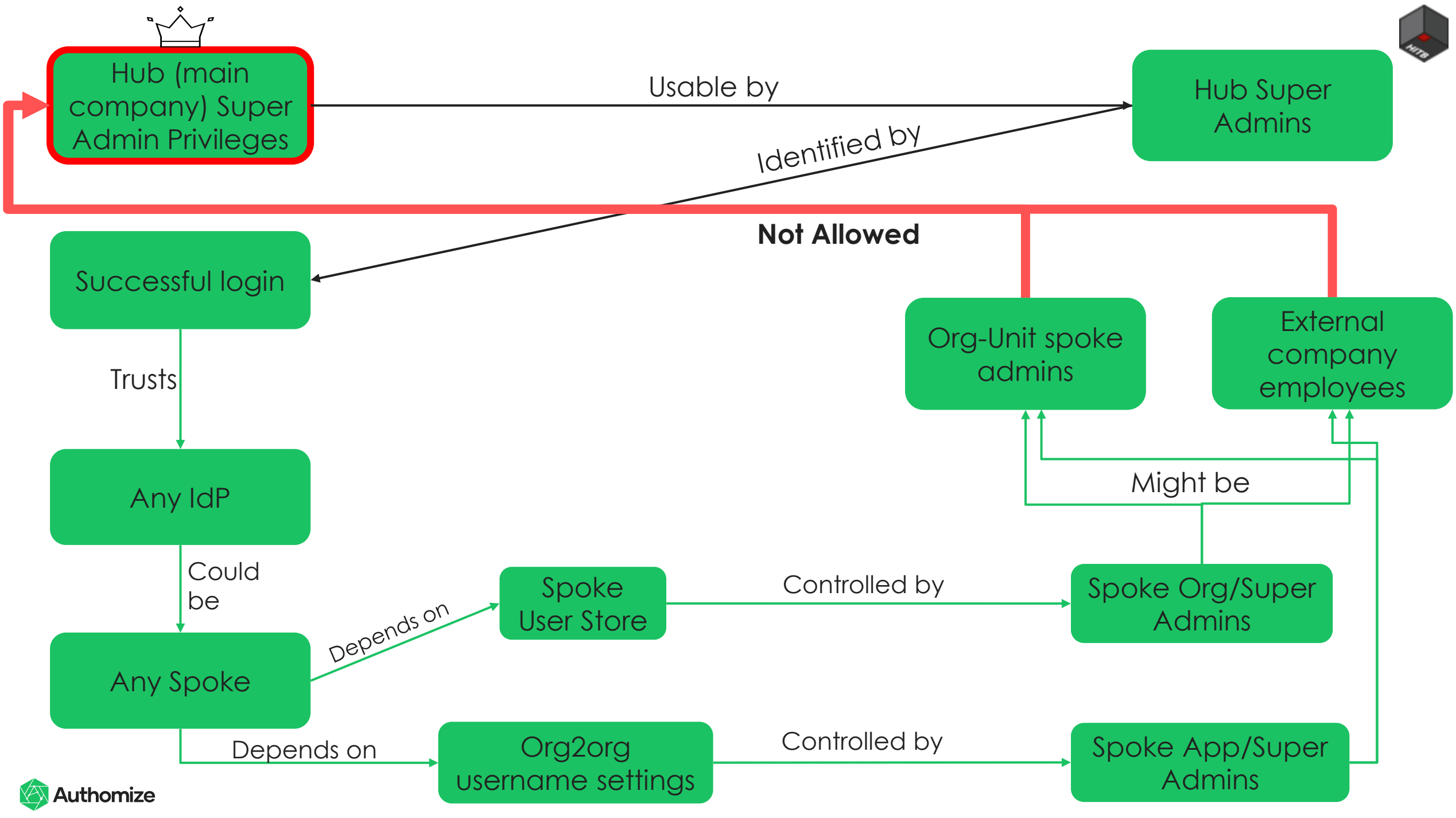
- Deployment model 2: Third-Party (including contractor) Outsourcing

- Each third-party provider gets a spoke, all spokes are connected as IdPs to the main company which acts as a hub

- There are more deployment scenarios such as M&A, Geo based, …

Source: Okta documentation




Hub (main company) Super Admin Privileges
Usable by
Hub Super Admins
Identified by
Not Allowed
Successful login
Org-Unit spoke admins
External company employees
Trusts
Any IdP
Might be
Could be
Spoke User Store
Controlled by
Spoke Org/Super Admins
Depends on
Any Spoke
Depends on
Org2org username settings
Controlled by
Spoke App/Super Admins
Authomize

Validation phase 2 – Hub & Spoke impersonation

"There's no way that the hub trusts the spoke implicitly"

"But this is what it means to be an IdP and it says so on the whitepaper page in Okta site"

"You're saying that admins of org-units, acquired companies, and outsource companies have hub admin access? No way - they must block duplicate usernames"

"Yes – see the documentation, it says *duplicate name support* is a feature and besides, it works"

- **Out-of-the-box duplicate name support:** This configuration offers native support for duplicate usernames, since duplicate usernames can exist in different orgs.

-- Silence --

"Let's try to understand why"

[Several days later] → Conclusion: it's the easiest way for Okta implement OU-like functionality

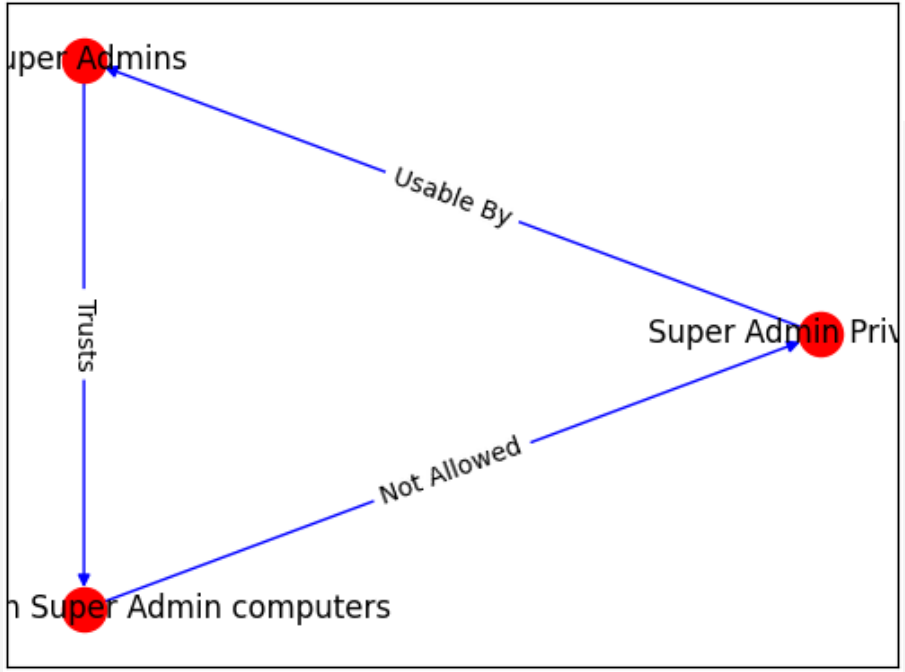
Okta has controls to prevent admin login (RegEx – which is bad) that don't prevent downstream app impersonation + ways to prevent downstream impersonation (undocumented & requires customer architecture changes)

Hub & Spoke based attack scenarios to escalate privs

1. Small acquired company added as spoke → Admin in the acquired company spoke is compromised → Spoke admin used to impersonate super-admin → $$$
2. A third-party outsource company is connected as spoke → Admin in spoke (non-employee) impersonates CEO in apps → Gains access to confidential data → Ransom demand arrives
3. Insider admin creates an Okta developer tenant → He connects or modifies a spoke to point to his spoke → Continues to maintain access post “retirement” → $$$

Conclusions, reactions and summary

Build your own graph analysis



Input data

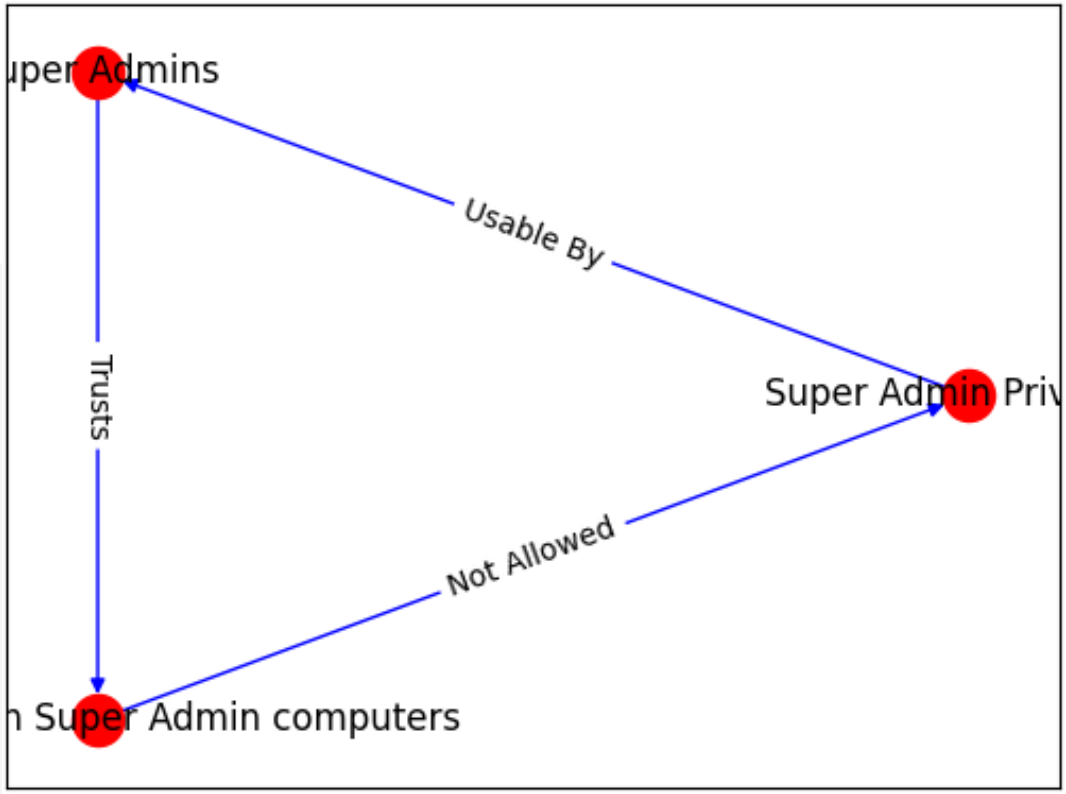
```
import networkx as nx

G = nx.DiGraph()

G.add_node("Super Admin Privileges")

G.add_node("Super Admins")

G.add_edge("Super Admin Privileges", "Super Admins", type="Usable By")

G.add_node("Local Admins on Super Admin computers")

G.add_edge("Super Admins", "Local Admins on Super Admin computers", type="Trusts")

G.add_edge("Local Admins on Super Admin computers", "Super Admin Privileges",
type="Not Allowed")
```

Build your own graph analysis

Find suspicious cycles

```
nx.find_cycle(G, source="Super Admin Privileges")
```

```
nx.find_cycle(G)
```



Visualize quickly (many better packages out there)

```
import matplotlib.pyplot as plt
```

```
nx.draw_networkx(G, pos=nx.circular_layout(G), node_color='r', edge_color='b')
```

```
nx.draw_networkx_edge_labels(G, pos=nx.circular_layout(G),
edge_labels=nx.get_edge_attributes(G, "type"))
```

```
plt.show()
```

Don’t forget to save your graphs …

Public reaction

- Lots of FUD regarding this research
- Questions to ask
  - If app admins can get super admin password – why bother having a different role?
  - If one factor is enough, why configure MFA?
  - If the master password should be shared, why is no one else in the same field doing this? (password managers or SSO vendors supporting password based SSO)
  - If Spokes represent a lower security domain (OU, third party, M&A), why is the default config allowing them to impersonate anyone in the higher domain – the Hub (and this is advertised as a feature)?
- How losers try to capitalize on such opportunities 💩

Disclosure

- We have worked closely with Okta's security team disclosing these issues
- On one hand, those are considered as non-vulnerabilities – part of the system specification
- On the other hand, there is an intent to add controls around these areas and the separation between roles is broken

- More coming soon…

Summary

- KGs are a very useful tool to collect knowledge for attack and defense
- You can apply KGs to do trust analysis at scale and systematically find logic bugs, it is simple to code that
- The implications of #PassBleed issues are:
  - App Admins can get anyone's, including Super Admin, passwords
  - Spoke (OU, M&A, third party) admins can impersonate anyone in the hub, or downstream apps, in the default config
- Don’t believe the FUD ☺

Questions?

Learn more:

- **Find me on twitter: @gal_diskin**
- **See our YouTube Demos explaining and demonstrating #PassBleed risks step by step or read the blogs on authomize.com/blog**