



**Push**

# **SaaS Attacks Report**

**2024 edition**

A collection of SaaS attack techniques to help defenders understand the threats they face

# Table of Contents

## Foreword: One year since launch

## Introduction

- Modern work happens in the browser
- IT transformation increases the threat of identity attacks
- Attackers are consciously targeting cloud apps
- Identity is the new perimeter
- Know your enemy

## The SaaS attack matrix

- Using this resource

## Understanding SaaS attack paths

### The paths to initial access

- Ghost logins
- AitM phishing
- Credential stuffing
- Session cookie theft
- MFA downgrade

### Chaining techniques together

#### Attack scenario #1: SAMLjacking a poisoned tenant

- What's a poisoned tenant?
- What's SAMLjacking?
- Why combine them?
- Attack walkthrough
- Paths to user compromise
- Impact

#### Attack scenario #2: The shadow workflow's evil twin

- What's a shadow workflow?
- What's an evil twin integration?
- Why combine them?
- Medium stealth option: Pre-existing use by org
- High stealth option 1: Pre-existing use by user
- High stealth option 2: Azure admin consented app
- Targeting Azure using Zapier
- Enumerating potential targets
- Creating shadow workflows
- Profit
- Impact

## Conclusion

# Foreword: One year since launch

A lot can happen in a year. Since we released the [SaaS attack matrix on GitHub](#), we've seen a huge amount of activity in the world of identity attacks. Now that we're a year on since launch, it's about time that we take another look at this report and reflect on what's changed.

When we created our open source repository of SaaS-native attack techniques, we made a conscious break away from the endpoint-focused techniques captured in industry resources like the MITRE ATT&CK Framework.

At the time, we were anticipating a shift that was yet to fully materialize. But since then, we've seen the impact of SaaS account takeover attacks laid bare, with attacks on product and service providers and their customers. More than that, we've seen the cyber crime ecosystem tilt toward credential theft and stuffing attacks, with a booming market for breached credentials feeding, and fed by, a continual pipeline of data breaches – giving attackers plenty of first, second, and third-party victims to extort – often from targeting a single SaaS app. [Snowflake, billed one of the biggest breaches in history](#), is a telling example of this that we'll no doubt look back on as a watershed moment.

Even a year ago the conversations we were having with customers, and general awareness of the problem space, was very different. Businesses still needed to be convinced that they were, in fact, using more SaaS services than they realized. And that these services could be abused to access and exploit data in a similar way to traditional networks. Fewer people are surprised by this. It isn't an exaggeration or marketing fluff to say that identity attacks are the #1 threat facing organizations today.

Digital identities – the keys to said SaaS apps – are clearly the weakest link for security teams today and therefore the lowest-hanging fruit for attackers to reach for. This makes resources like the SaaS attack matrix more relevant than ever, both for red teams seeking to emulate the latest offensive techniques, and blue teams trying to defend against them. Understanding these techniques is essential for building effective defenses, and identifying where new platforms and controls are required to do so.

Whether you're a red or blue teamer, the SaaS attack matrix is designed to help you to understand how organizations can be targeted through their identity surface, and how to prepare your business or your clients to prevent, detect and respond to these techniques accordingly.

**At Push, our mission is to stop identity attacks such as those documented in the SaaS attack matrix. To find out more about how we prevent, detect and respond to identity attacks using our browser agent, feel free to [sign up for a demo or try us out for free](#).**

“If fifty people work at this place, that's fifty accounts times however many services I just listed. What, ten? So, we're talking five hundred various logins to different websites now. Who's got permission to see what and where?... This is a new territory for security teams to navigate. You hear about this in general terms like 'least user privilege' and this sort of stuff, but you don't have people who are experts in Zapier account security who will audit what apps you have given permission to regularly. This is a big challenge to keep up with.”

**Jack Rhysider, [Darknet Diaries EP:148](#), discussing Push Security threat research**

# Introduction

**Offensive security drives defensive security. We're sharing a collection of SaaS attack techniques to help defenders understand the threats they face.**

Our goal at Push is simple — to stop identity attacks.

Today, the vast majority of identity vulnerabilities exist in the context of SaaS apps. The reasons for this are clear: Security teams have reduced central oversight and control over SaaS apps than they are used to, these apps exist in large numbers per company, and the identities that are used to access these apps are... complicated, to say the least. Securing hundreds of apps, with thousands of associated identities, is therefore no mean feat.

To be able to talk about SaaS attacks, and ultimately build controls to stop them, we need to have a shared understanding of what these techniques are. To get that conversation going, we've pulled together all the techniques we're aware of, and our research team has even added a bunch of new ones. These attacks demonstrate how to compromise a company without touching the endpoint.

But before we take a look at the matrix itself, let's set the scene a little.

## Modern work happens in the browser

If someone asked you where you work, you probably wouldn't answer, "My browser." But that would be the truth.

Modern work is no longer something that happens in the office, on a local network. Or even by remotely accessing a 'conventional' network. Instead, an increasingly dominant percentage of businesses run on a vast sprawl of interconnected cloud apps. This is all achieved through the medium of your preferred web browser — the gateway to an ecosystem of apps and the identities that are used to access them.

The SaaS revolution and product-led growth have had a huge impact on the structure of company IT. Most organizations today are using tens to hundreds of SaaS applications across business functions. Some are entirely SaaS-native, with no traditional network to speak of, but most have adopted a hybrid model with a mixture of on-premise, cloud, and SaaS services forming the backbone of business applications being used. So, modern businesses effectively run on interconnected SaaS apps that are accessed by employees via their web browser.

The shift to SaaS naturally means that your data is everywhere. As employees sign up for apps — even if it's just on the trial or free tier — they're consenting to app permissions, which request access to all kinds of company data. This may be a read-write permission that gives the app access to the employee's Google Drive or OneDrive account, where they're working with and sharing confidential company information and customer data. Or perhaps the app needs access to their email to enable features like calendar-sharing or email-scheduling... You can see how this quickly becomes a third party security and data loss issue.

But, I hear you say, SaaS apps are basically web apps that are run in the cloud and accessed from endpoints, so then WebApp, endpoint, and cloud security should cover all of SaaS. Right? Well...

# IT transformation increases the threat of identity attacks

The bulk of SaaS adoption is user-driven, as opposed to centrally managed by IT, as bottom-up adoption is inherent to product-led growth. Our data shows that only 1 in 5 SaaS apps have been sanctioned by the business. The majority is simply unknown and therefore has not been reviewed at all.

Making apps easy to sign up for and low effort to support means you need to make some interesting choices when it comes to designing account creation and recovery flows. Many apps allow users to sign into apps using multiple methods, easily invite collaborators (internal and external), avoid verifying new account email addresses, and allow simultaneous login methods and user sessions. This is not laziness, these are conscious design choices in the name of encouraging sign-ups and reducing customer support — not driven by security clearly, but not accidents.

**Note:** If you want apps to prioritize security, it needs to be baked into your procurement processes – make it desirable for apps to champion security features as a revenue-driving factor!

## Attackers are consciously targeting cloud apps

The fact that SaaS app use is now so widespread, and the ways that accounts are configured leave them open to abuse, SaaS apps and cloud services are now a seriously juicy target for attackers. They know that if they can compromise an identity, they gain a foothold into the company's data. Worse, they can leverage many of the features built into these apps to socially engineer employees and move laterally to other identities and apps.

In many cases, these attacks also give attackers persistent access, resistant to many incident response activities. So, even if your security team detects something suspicious and performs the typical recovery actions on affected identities (like resetting passwords, for instance), the attacker will maintain access to your systems and data.

There's also the fact that your incident response teams aren't necessarily cloud experts, or familiar with the nuances of every app that now makes up your business IT ecosystem. There are always going to be quirky and novel ways that an attacker can abuse a specific app. While they may have the time to invest in really getting under the skin of an app that interests them, there's no feasible way for security pros on the other side to find the time to do the same.

When all the sensitive data and functionality that an attacker could want exists in internet-accessible SaaS apps, why bother with any of the traditional attack paths? These avenues are well protected by years of control investment. In contrast, if the goal is to compromise an app like Snowflake and dump the data from it, the attack chain is way shorter than a traditional network-based attack. And with the increasing popularity of SSO platforms like Okta, an identity compromise can quickly spread across apps and accounts, increasing the potential blast radius.

75% of attacks in 2023 were malware-free and "cloud conscious" attacks (instances of attackers deliberately targeting specific cloud services to leverage functionality as part of an attack) increased by 110% (**CrowdStrike**).

# Identity is the new perimeter

Cloud identities and SaaS apps are the new battleground because identities have become the new perimeter.

The SaaS cyber kill chain revolves around identity. As we'll explore later, initial access, persistence, privilege escalation and lateral movement are all predominantly identity-based when targeting SaaS apps.

With endpoints, we at least knew exactly how many machines to secure, who owned them, and what was installed on them, so they were relatively easy to monitor and maintain. With cloud identities and SaaS apps, things become much trickier. There are now thousands of identities across hundreds of apps per organization – each app with their own unique levels of configurability. These are effectively the lowest-hanging fruit for attackers to pick at today, giving rise to the saying that, **“hackers don't hack in, they log in”**.

In this new world, attacks don't even have to touch the old perimeters, because all the data and functionality they could want exists on the public internet.

Don't just take our word for it: besides real-world examples of this shift like the attacks on Snowflake's business customers (and their end-customers respectively) studies from researchers and providers are in consensus:

- 80% of attacks today involve identity and compromised credentials (**CrowdStrike**).
- 10x increase in identity attacks in 2023, average of 4,000 attacks per second (**Microsoft**).
- 79% of web application compromises were the result of breached credentials (**Verizon**).
- One million new infostealer logs are distributed every month, with 3-5% containing corporate creds (**Flare**).
- 147,000 token replay attacks were detected by Microsoft in 2023, an 111% increase YOY (**Microsoft**).

## Know your enemy

The most important thing that organizations can do is to acknowledge the severity of the threat, and resolve to do something about it – sooner rather than later. Identity attacks are not a new threat for organizations to tackle, but the changes to business IT have made it a bigger one. It's also exposed that the controls we've historically relied on – protecting the email inbox, blocking lists of known-bad domains, and reminding users not to click phishing emails – just aren't cutting it (and, well, they never really did).

You also need to know your enemy (otherwise, what are you looking for?). We think that resources like the SaaS attack matrix are a big part of the solution here. So with all that said, let's take a look at the SaaS matrix and how you can apply it to defending your organization against identity attacks.

# The SaaS attack matrix

Now we've set the scene, let's get on with the main event.

We've taken inspiration from the MITRE ATT&CK framework (certainly intended as the sincerest form of flattery), but wanted to make a conscious break away from the endpoint-focused ATT&CK techniques and instead focus on techniques that are SaaS-specific. In fact, these techniques don't touch endpoints (so they bypass EDR) or customer networks (so they bypass network detection).

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								

Note: As this is an open source project, which we'll be continuing to develop over time (along with, hopefully, contributions from you and your team!), the most updated form of this matrix will always be found at our [GitHub repo](#). There, you'll also find more detailed descriptions of these techniques.

# Using this resource

You can use the SaaS attack matrix in a number of ways. We speak to organizations who use it in various offensive and defensive security exercises, from threat modeling, to simulating cloud-native red team attacks, to detection engineering – and many steps in between.

If you're an organization looking to protect yourself against SaaS attacks, you'll need to follow three high-level steps:

## Step 1: Understand your identity threat model

The first step towards detecting and responding to SaaS attacks is to understand what those attacks are. Knowing the techniques and how they are chained together as part of an attack path has been a staple of attack detection best practice for many years, and remains key. Naturally, you also need to understand which apps and identities you're using across your organization – after all, you can't protect what you don't know about.

## Step 2: Review security telemetry and pinpoint gaps

The second step is to establish whether you have the tools and data points to be able to both identify and stop these attacks – whether by proactively preventing them (the best solution), or by detecting and reactively responding to them.

## Step 3: Establish proactive and reactive security controls

Visibility without action is just noise, so building detection and response capabilities aligned to these techniques is key. These might be things you can create using your existing security stack using your SIEM/SOAR (build), you may need to invest in a new data source or tool (buy), or a combination of the two.

The most logical place to focus on detecting and blocking SaaS attacks is at the identity layer. This means during the initial account takeover attempts (e.g. phishing, credential stuffing, session hijacking via stolen cookies) and any subsequent authentications to downstream apps and services.

Unfortunately, there's a pretty substantial telemetry gap when it comes to identity attacks. This is due to a combination of traditional network logs being unable to correlate identity data points at scale, IdP logs being limited to only the apps that are accessed via your primary IdP, and app logs being of variable quality and availability per each app vendor. And because these attacks don't touch the endpoint, controls like EDR don't come into play either.

This is the problem we're tackling at Push by bringing a new surface for detection and control enforcement to the table – the browser. If you want to learn more, feel free to [sign up for a demo or try us out for free.](#)



# Understanding SaaS attack paths

Since we're not targeting endpoints, let's talk about the new targets: identities. We found it was useful to think about these identities not as isolated islands, but much more like a graph – less a single web-server on the internet and more like many Windows endpoints in your Active Directory.

It's not just individual attack techniques and the phases of the cyber kill chain that matter – it's also how you chain attacks together. Compromising a standard user on a low-risk app can quickly snowball into a wider compromise, enabling attackers to pivot to the higher-risk apps with the functionality or data they want.

Attackers can (and will) attempt to achieve their staged objectives (take over an account, backdoor an app, spread to other apps, exfiltrate data...) using multiple techniques, often in tandem. Equally, they won't stop there. The goal is always to progress to the end of the attack chain in order to achieve whatever their objectives may be – data theft and extortion, abusing in-app functionality (e.g. to issue fraudulent payments), etc.

## Hot right now: Initial access techniques

The majority of techniques that have risen to prominence sit predominantly in the initial access phase. Since the matrix first launched, we've added more techniques to initial access than any other category, including [ghost logins](#), [AitM phishing](#), [session cookie theft](#), [MFA downgrade attacks](#), and [guest access abuse](#), all of which are methods of account takeover – complementing the classics like [credential stuffing](#).

We'll spend a bit of time delving into these techniques in the next section, but let's first consider what this tells us about SaaS attacks.

The initial identity attack designed to achieve account takeover is the most important part of the SaaS attack chain. The fact that attackers are focused on finding new ways of compromising identities illustrates the value, but also the fragility of the identity controls that most organizations are relying on (which may also be one of the reasons attackers are fixated on it). Whether we're talking about anti-phishing protections, conditional access policies, or MFA – attackers are continually finding new ways of getting around them.

We only really need to look at what recent high-profile breaches show us about how lucrative it can be for attackers to find ways to take over workforce identities in order to access web-based business applications – with [the recent Snowflake attacks](#), going down collectively as one of the biggest breaches in history, being the elephant in the room.

If all an attacker really needs to do to cause harm is log into an app and abuse its legitimate features and functions, there really is no margin for error – you need to successfully stop the initial identity attack every time. You can't rely on your endpoint and network controls to catch them later like you used to. Equally, it's unlikely that your CASB or DLP solution can stop a legitimate app using legitimate features like [API-based workflows](#) from sending data to attacker-controlled infrastructure.

It's a classic case of attackers only needing to win once. And right now, it's a numbers game that they're winning enough to keep them coming back for more.

# The paths to initial access

Given the importance of initial access, let's take a closer look at the most notable techniques from the last year.

## Ghost logins


[Ghost logins](#) is a technique that exploits the fact that SaaS user accounts often enable multiple simultaneous logins using different sign-in methods.

Ghost logins can be used for both the initial access and persistence stages of a cyber attack, doubling up as a defense evasion technique due to the low visibility that most organizations have of this issue.


## Initial access

For initial access, the technique exploits the fact that local and SSO logins can exist simultaneously. Given that many apps are self-adopted by users, it's likely that many users will default to a local username and password login at this stage. If the app is later adopted companywide and brought into SSO, the original local login will continue to exist unless explicitly disabled or deleted. Because MFA is applied at the app and SSO level independently, it's possible to end up with an SSO login that requires MFA, but a local login that does not.

This creates an easy target identity for attackers to look for. When combined with other identity vulnerabilities such as weak, breached, and/or reused passwords, attackers can easily automate ghost login discovery and exploitation at scale.



We saw the impact of ghost logins for initial access with the recent ShinyHunters campaign against Snowflake customers. Because Snowflake accounts did not require mandatory MFA for accounts, or give admins the ability to enforce MFA by default, attackers were able to find and exploit a large number of Snowflake accounts using breached credentials from historical data breach dumps. Much of the industry response focused on ensuring SSO and MFA were deployed, but the practicalities of gathering data and manually unsetting local passwords in Snowflake meant that ghost logins were easy to overlook by organizations responding to the attacks. [Check out this clip from our recent webinar](#) to see just how tricky some apps make it to find and fix ghost logins!



## Persistence

Ghost logins can also be created after an attacker has established access to an app. For example, if a social login is used to access an account, an adversary may be able to configure a separate username/password login, or even (though much less commonly) connect a second social account that the adversary controls. If the account has sufficient privileges, it may also be set up or change the SAML login settings to inject a malicious URL (for example to an attacker controlled tenant) or simply configure API access to forgo the need to log in entirely.

# AitM phishing

[Adversary-in-the-Middle \(AitM\)](#) phishing is a newer variant of phishing that uses dedicated tooling to act as a web proxy between the victim and a legitimate login portal for an application the victim has access to, principally to make it easier to defeat MFA protection (with the victim responding to the MFA request as part of the attack). Because the attacker is sitting in the middle of this connection, they are able to observe all interactions and take control of the authenticated session.

As it's a proxy to the real application, the page will appear exactly as the user expects, because they are logging into the legitimate site – just taking a detour via the attacker's device. For example, if accessing their webmail, the user will see all their real emails; if accessing their cloud file store then all their real files will be present, etc. This gives AitM an increased sense of authenticity and makes the compromise less obvious to the user.

Alongside AitM phishing is Browser-in-the-Middle (BitM), really a form of sub-technique. Rather than act as a reverse web proxy, this technique tricks a target into directly controlling the attacker's own browser remotely using desktop screen sharing and control approaches (such as VNC and RDP). This enables the attacker to harvest not just the username and password, but all other associated secrets and tokens that go along with the login. In this case, the victim isn't interacting with a fake website clone or proxy – they are literally controlling the attacker's browser.

**This is the virtual equivalent of an attacker handing their laptop to their victim, asking them to login to Okta for them, and then taking their laptop back afterwards.**

A growing majority of modern phishing attacks typically leverage AitM or BitM tooling – they are now the standard choice for threat actors, offering the ability to bypass MFA without any real tradeoff.

# Credential stuffing

[Credential stuffing](#) attacks continue to pose a risk to organizations. Despite the fact that MFA has now become an expected control, accounts without MFA continue to be hacked as a result of using weak, reused, and/or previously breached credentials.

The recent Snowflake attacks are a timely reminder of this risk. Credential stuffing is being fed by an increase in the number of [info-stealer](#) attacks designed to harvest credentials to be sold on criminal marketplaces. Info-stealers have been boosted by the success of Snowflake attacks ([where 80% of the credentials used to access accounts could be traced back to info-stealer infections dating back to 2020](#)). The [successful attacks on Microsoft earlier this year](#) were also the result of an initial credential stuffing campaign to breach 'test' identities, which was subsequently used to compromise an OAuth application with access to the Microsoft corporate environment.

# Session cookie theft

Attackers are increasingly [targeting session cookies](#) to be able to hijack live user sessions as a means of getting around MFA. Although session cookies are predominantly stolen via infostealers, a malware that targets user endpoints to harvest data from web browsers, techniques like AitM and BitM phishing described above are also methods of stealing session cookies and hijacking sessions.

While the majority of infostealer data dumps result in credential stuffing attacks rather than session hijacking, as the infostealer marketplace continues to heat up, it's likely that more instances of session cookie theft will be the cause of breaches going forward.

## MFA downgrade

While many organizations are waking up to the fact that it's not enough to have any old MFA method, it's still often overlooked that you need to actually remove or disable the phishable methods. Otherwise, in many cases they remain valid, opening affected identities up to [MFA downgrade](#) attacks.

Just because a user has a phishing-resistant factor setup (such as passkeys) and may use them by default, it does not mean they are necessarily enforced. Often, services support the use of multiple authentication options, particularly for second factors. In particular, passkeys are device-bound and so enforcing their use prevents logins from other devices and can cause recovery issues in a lost/broken device scenario. Therefore, it's common for the default case to be that passkey authentication is optional, rather than required.

When used in combination with AitM phishing tools, it's possible for attackers to modify requests/responses so as to prevent the ability of passkeys to be selected as a login option and prompting the user to use vulnerable factors, such as passwords, TOTP and push notifications instead. Since the server-side supports other authentication options, if the user continues and enters one of these alternative factors then their authenticated session will be compromised – despite the fact they usually use phishing-resistant MFA methods like passkeys or similar.

# Chaining techniques together

Now, let's explore how we might create attack chains by creatively combining techniques across different stages of the attack lifecycle, once a level of initial access has been achieved.



## Stop right there!

Are you skimming through? Hopefully this caught your attention!

If you're more of a visual learner than a reader, you can check out a [video demo from our VP R&D, Luke Jennings, on our YouTube channel](#). In the demo, Luke covers:

- Initial access using AitM and BitM toolkits.
- Persistence and post-exploitation activity through session token theft and abusing OAuth integrations.
- Lateral movement and privilege escalation via the compromise of downstream apps.
- How Push detects and blocks this kind of attack.

If you're happy reading, then feel free to keep scrolling!

# Attack scenario #1: SAMLjacking a poisoned tenant

By combining two of our favorite new SaaS attack techniques, poisoned tenants and SAMLjacking, you can make a simple but effective attack chain.

## What's a poisoned tenant?

[Poisoned tenants](#) involve an adversary registering a tenant for a SaaS app they control and tricking target users to join it, often using built-in invite functionality. The end goal is to have some target users actively using a tenant you (the adversary) control.

## What's SAMLjacking?

[SAMLjacking](#) is where an attacker makes use of SAML SSO configuration settings for a SaaS tenant they control in order to redirect users to a malicious link of their choosing during the authentication process. This can be highly effective for phishing as the original URL will be a legitimate SaaS URL and users are expecting to provide credentials.

This is especially useful for targeting high-value SSO apps like Google or Microsoft Entra, which are typically better protected than your average app – so some of the more straightforward initial access techniques outlined in the previous section might not cut it.

There are also some pretty novel ways of [targeting Okta specifically \(we've labeled this 'Oktajacking'\)](#), which we don't have time to cover here – but you can check out our blog post for more information.

## Why combine them?

A poisoned tenant on its own could be an epic supply chain attack if you get really lucky. Imagine discovering an organization was wanting to migrate to Slack and then catching some key teams with a Slack poisoned tenant and gradually getting the whole organization migrated over. You'd have a goldmine of information as an administrator of the platform.

However, it's probably not that realistic to trick a whole organization into using an attacker controlled slack instance without anyone realizing. But it could be a lot easier to, for example, successfully invite a marketing team into using a new marketing app that helps them do SEO. This might be easier to perform, but it doesn't really give the attacker valuable data in the poisoned tenant of the marketing app...

On the other hand, what about SAMLjacking? It's a great technique on its own, but you still need to get users to login to the app. Sure, you'll be sending them a legitimate SaaS URL with a valid TLS certificate etc and so it's going to pass the sniff test for many people and also bypass email security appliances and similar security tools.

However, you're still effectively phishing them for credentials, the one thing we train users to be most suspicious about, so there is still a possibility they will spot the attack.

But what if you could combine these techniques so that a poisoned tenant didn't need to be a big, juicy target to be useful and a SAMLjacking attack didn't even necessarily require phishing someone directly? What if the attack could be successful just from a target accessing their own bookmarks or open tabs for an app they already use?

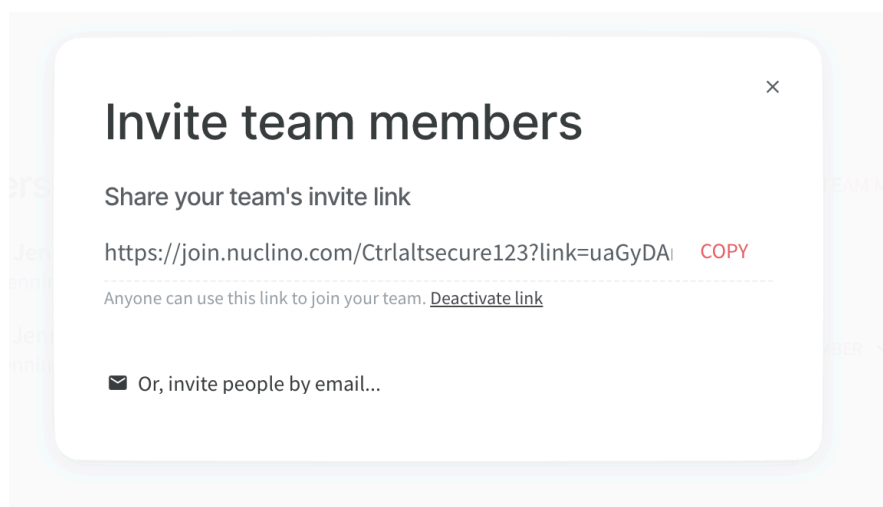
In a combination scenario, a user doesn't need to be phished for SAMLjacking. One day they go back to their tab and it's logged out and they get SAMLjacked while logging back in – they don't even have to click a phishing link.

## Attack walkthrough

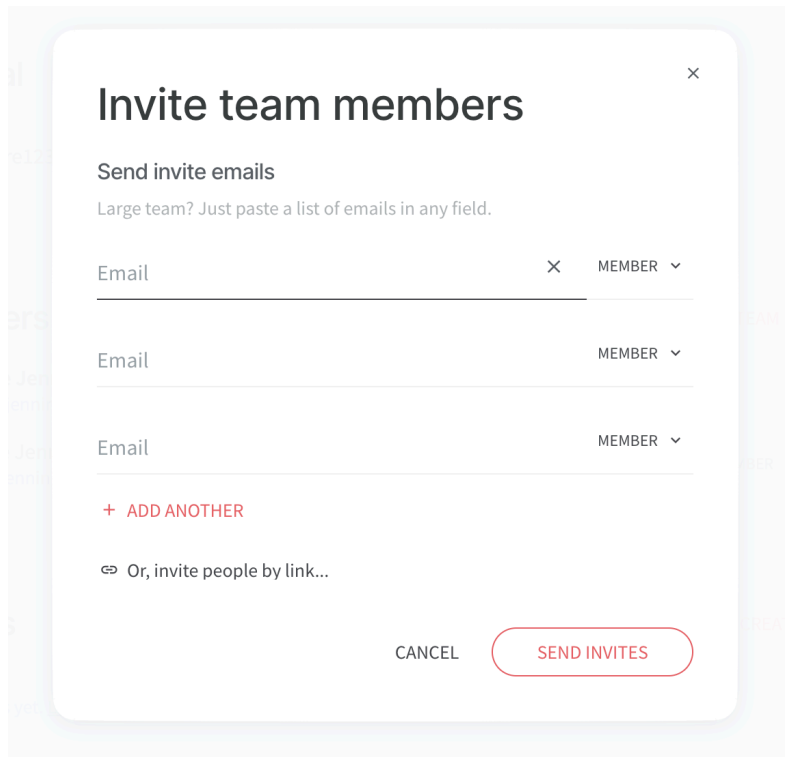
Let's consider an example of this making use of the SaaS-based wiki, Nuclino. This isn't a vulnerability with Nuclino per se and it won't be limited to Nuclino either. We've used Nuclino as an example because it's a great wiki platform we use at Push Security, so we're familiar with it.

### Setup a poisoned tenant and invite target users

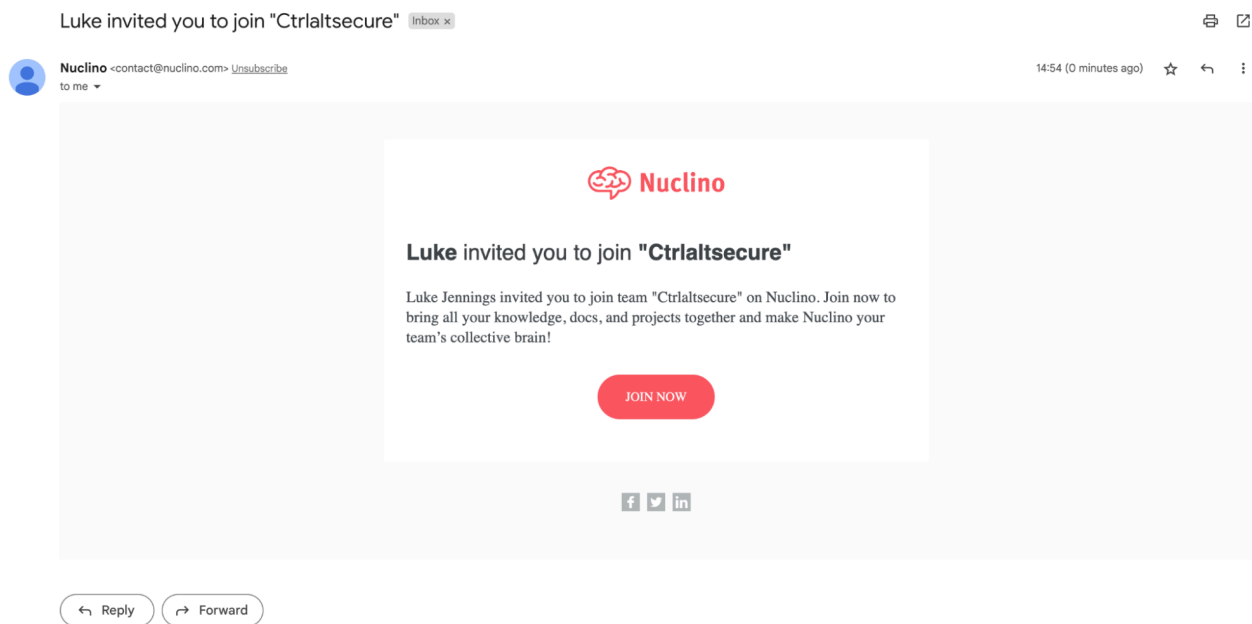
The first step for an adversary is to set up their poisoned tenant and then make use of the invite functionality to target some employees of the target organization. With Nuclino, you can either do this by sending sharing links directly to the target or invite them through the Nuclino app, and it will send out legit email invitations on your behalf.



Sharing link method of inviting new users



Email invite method of inviting new users



Example legit email a target user will receive from Nuclino when invited to join a workspace



## Target responds to the invitation or later signs up for Nuclino

The interesting thing here is that whether the target signs up for Nuclino directly from the joining link or they sign up for an account separately in future, they get mapped to the workspace they have been invited to by default.

**Create an account**

Join team Ctrlaltsecure123 on Nuclino.

First name Last name  
Johnny Victim

Email available  
victim@target-org.com

Password  
\*\*\*\*\*

**CONTINUE**

By proceeding to create your account and use Nuclino, you are agreeing to our [Terms of Use](#) and [Privacy Policy](#).

Already have an account? [Log In](#)

Account creation process the target user is prompted with on joining the workspace

### Step 3 - Configure a malicious SAML server

Once the adversary has a critical mass of users on their poisoned tenant, they can later engage the SAMLjacking attack.

To do this, they need to configure a custom SAML server. You can point this to a fake authentication provider they control that mirrors the appearance of the SSO provider the target users are accustomed to using in order to capture credentials.

### 📄 Data for your SSO provider

Copy and paste the following data to the settings of your SSO provider during setup:

#### ACS URL

<https://api.nuclino.com/api/sso/2c2777fe-46ca-4ae0-8ab2-a1323c398190/acs>

COPY

Copy and paste this to the settings of your SSO provider.

#### Entity ID

<https://api.nuclino.com/api/sso/2c2777fe-46ca-4ae0-8ab2-a1323c398190/metadata>

COPY

Copy and paste this to the settings of your SSO provider.

### 📄 Data supplied by your SSO provider

Copy the data supplied by your SSO provider from their settings and paste it here:

#### SSO URL

<https://my-evil-saml-server.com>

The SAML 2.0 Endpoint URL supplied by your SSO provider.

Custom SAML server settings pointing to a malicious SAML server

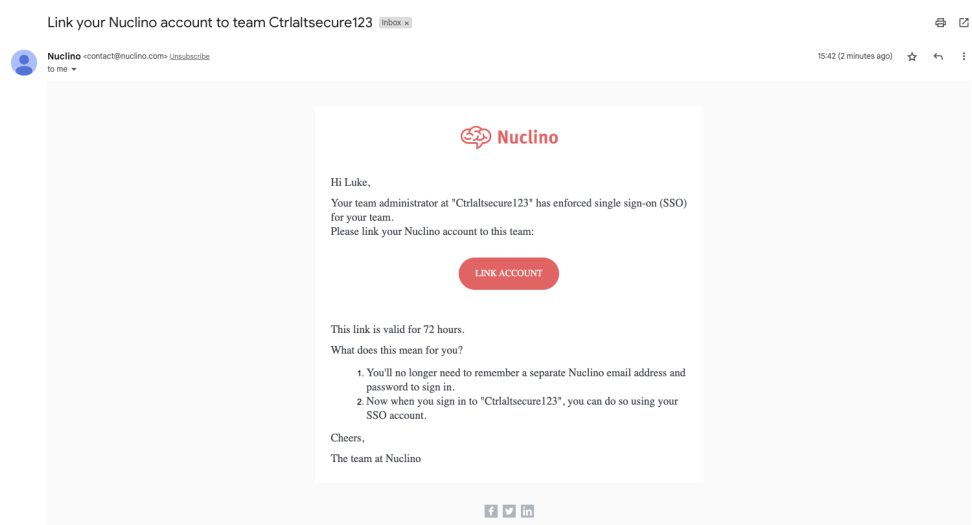
If you toggle the setting to require SSO, existing users will be sent emails prompting them to link their accounts to SSO. That leads to two possible paths to a user compromise.

## Paths to user compromise

### Path #1: User clicks link to configure SSO to poisoned tenant

This compromise occurs when the target sees the email that SSO has been configured and clicks the link in order to link their account to SSO. A smart adversary may improve the social engineering quality with an email sent out in advance informing users that the internal security team has requested Nuclino be linked to SSO. This makes the target expect the email and consider it legitimate.

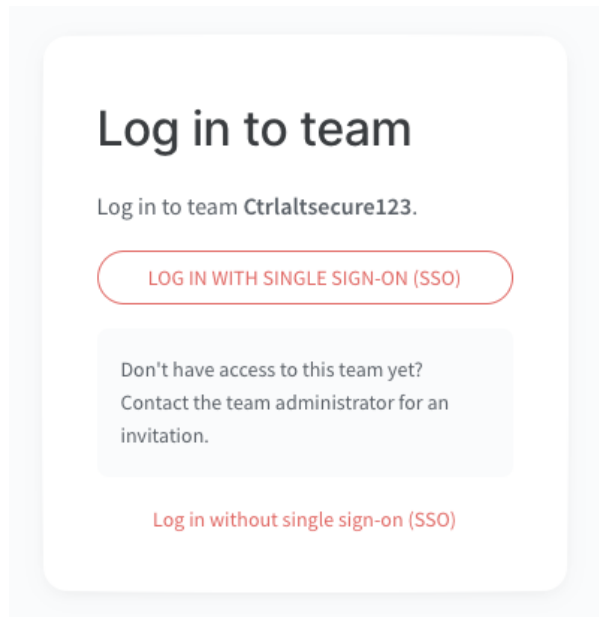
Even though the email is an official email from Nuclino and the link contained is an official Nuclino URL, it will immediately redirect to the malicious SAML server that has been configured, where credentials can then be captured.



SSO linking email sent by Nuclino to existing users

## Path #2: User (re)authenticates to poisoned tenant on timeout

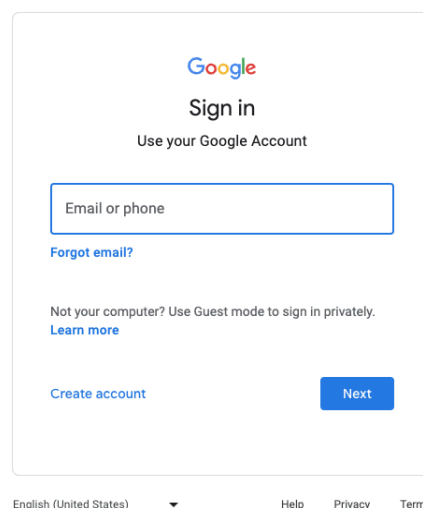
If the user ignores the email, the other potential outcome occurs when their session expires and they need to login again to regain access. This is similar to a watering hole attack. When their session expires, the target's open tabs or bookmarks will redirect back to the workspace specific login page, which will now look like this:



Workspace login page post SSO configuration

Clicking the button to login with SSO will immediately redirect to the malicious SAML server and launch the attack. Alternatively, if the target attempts to login without SSO, the login will fail with an error message telling them to login with SSO.

Either way, once the SAMLjacking has taken effect, they'll be faced with a familiar-looking SSO login page from a trusted source at a point they are expecting to enter their credentials - something even the most paranoid of users could easily fall for unknowingly.



Fake Google SSO login page the target user is redirected to

# Impact

At this point, having compromised multiple user's Google credentials, an adversary has a lot of options available:

- Access all data in Google apps like GMail, Google Drive etc
- Access other SaaS apps that use SSO with the same Google account
- Access other SaaS apps that use [passwordless logins](#)
- Access other SaaS apps via email [account recovery](#)

Essentially, this can potentially lead to a compromise of every SaaS application accessible by the compromised user — all from the use of a poisoned tenant for an app with no particularly sensitive data or permissions.

## Even low-risk apps can be a significant threat

We have seen how two new SaaS-focused attack techniques can be combined into one more effective attack chain. This shows how a successful poisoned tenant attack for even a low risk app can still be a significant threat when combined with a SAMLjacking attack.

This demonstrates even the least sensitive edge cases of SaaS sprawl can represent a vector to laterally move to compromise much more valuable assets. History taught us that protecting core production assets was not enough. Adversaries often achieved compromises via test systems and unsecured development resources. What we are seeing now is that this parallel exists in the SaaS-native world too. Therefore, we need to be protecting all SaaS resources with greater vigilance than their standalone sensitivity would indicate.

# Attack scenario #2: The shadow workflow's evil twin

This attack chain is nearly invisible and is a simple, but very stealthy approach that maintains persistent access.

We'll be combining shadow workflows with an evil twin integration for an especially sneaky and flexible method of persistence. We'll be using Zapier integrating with Azure as our primary example.

## What's a shadow workflow?

A [shadow workflow](#) is a technique for using SaaS automation apps to provide a code execution-like method for conducting malicious actions from a legitimate source using OAuth integrations. This could be a daily export of files from shared cloud drives, automatic forwarding and deleting of emails, cloning instant messages, exporting user directories — basically anything that is possible using the target app's API.

The fact automation apps utilize OAuth integrations means they also function as a very effective method of maintaining persistence. Think of shadow workflows as the offensive PowerShell of the SaaS world.

## What's an evil twin integration?

Creating a new OAuth integration, even if using a legitimate SaaS application, could be viewed as suspicious if seen by a security team or the affected user. This is especially true if an account compromise is discovered and an IR team sees a consent for a new OAuth integration in the log that the compromised user does not recognize.

An [evil twin integration](#), however, reduces the chances of discovery by reusing an existing legitimate integration for malicious purposes.

## Why combine them?

While shadow workflows are incredibly powerful on their own, as malicious use of OAuth integrations becomes more common, security teams will start regularly checking for new, or unknown, integrations in response to security incidents. While automation apps are legitimate SaaS services, shadow workflow attacks could still raise question marks during incident response if it's connected shortly after a compromise and/or if the affected user has no knowledge of it.

Additionally, as use of security tools that provide visibility of OAuth integrations (like Push!) increases it will become increasingly dangerous for an adversary to create a new OAuth integration. That's because the target user and possibly even security teams may be notified.

This leads us on to evil twin integrations. Their power is in making use of existing integrations so they can avoid appearing as a new integration and getting flagged or sending alerts to security teams. That makes them much stealthier and increases the likelihood of a successful attack. There are three possibilities here that lead to two different levels of stealth for the attack:

- 1. Medium stealth option:** Making use of an automation app used legitimately by the organization, but not by the target user specifically.
- 2. High stealth option 1:** Making use of an automation app used legitimately by the target user themselves.
- 3. High stealth option 2:** Making use of an automation app that has been granted admin consent.

## Medium stealth option: Pre-existing use by org

This option is by far the most likely option to be applicable in a real-world situation. Here's how it works:

The consent for the targeted user will be new and will generate an audit event to show that, but the integration itself will not be new inside the organization and may even be formally approved by the security team already. This will help evade general detection mechanisms as it won't be seen as a brand new integration at the organization level that requires careful scrutiny. It's much harder to evaluate new consents on a per-user basis for existing integrations if the organization is of any significant size.

The downside, however, is that this attack stands a greater chance of detection if notifications are delivered directly to the affected user. Alternatively, if the original compromise is discovered, incident responders are more likely to discover this consent during an investigation. That's because the affected user would know they aren't using the automation app and incident responders are likely to explore logs showing consents to new OAuth integrations and permissions shortly after a successful compromise.

Using Azure as an example, while no new service principal is created in this case, the audit logs still show a new consent for the targeted user to the existing Zapier app:

Core Directory	ApplicationManagement	Consent to application	Success	Zapier To Do
Core Directory	UserManagement	Add app role assignment g...	Success	Zapier To Do, luke.jennings...
Core Directory	ApplicationManagement	Add delegated permission ...	Success	Microsoft Graph, 8d2f8b13...

Azure audit logs showing a new user consent for a Zapier integration already in use by other users inside the organization

## High stealth option 1: Pre-existing use by user

This is the holy grail option, but is likely to require more luck in the real world. It requires that the target user is already using an automation app, which the adversary could compromise and utilize. If the compromised user has already consented to permissions useful to the adversary, such as access to sensitive data like email and file stores, then new malicious workflows can be created without requiring the user to consent to new permissions.

Consequently, there will be no new integration observed at the organization level, no new user-specific consents for sensitive permissions and the target user would indicate they're just using a legitimate app if questioned by incident responders. None of the three audit log entries shown above would be present in this scenario either.

## High stealth option 2: Azure admin consented app

There is a mixed scenario when permissions for an automation app (or any app you want to use for an evil twin integration) have been granted tenant-wide [admin consent in Azure](#). In this case, the administrator has effectively consented to permissions for all users, even if they aren't currently active users of the app.

This means when a new user integrates the app, it does not generate a new permission grant since it is effectively already granted. Consequently, the three log entries shown above would not be present in this scenario even if integrating the app for a user that has never used it before.

This gives the best level of flexibility for an adversary as they can avoid generating new permission grant logs for any user. However, it's not quite as stealthy as when the targeted user already makes use of the app as there is no history of legitimate app logins or activity for the user prior to the compromise to blend in with.

## Targeting Azure using Zapier

In this case, we're going to use Zapier as our automation app example and Azure as the primary target for integrations and there will be no admin consent involved. We'll also be using Google Workspace for data exfiltration. There are many other examples we could have used here, though - Make.com, IFTTT, Retool, Tines, Microsoft Power Automate and many other SaaS apps have powerful automation and integration capabilities and could be used for similar purposes.

Azure and Google Workspace are also obvious juicy targets for integrations, but automation apps support integrations with vast numbers of other SaaS applications, so there are many possible targets.

So, let's say we've compromised a target user's Azure account. Perhaps we have conducted a successful credential stuffing attack, a phishing attack including MFA code proxying or even achieved a traditional endpoint compromise and have stolen the user's session tokens.

Whatever the case, we have temporary control of the user's account, either until the session expires or the user changes their password. If the original compromise is detected, that could happen quickly, so we want to conduct some malicious actions to make use of the access while we have it and to also gain persistence so we maintain our access beyond a password change.

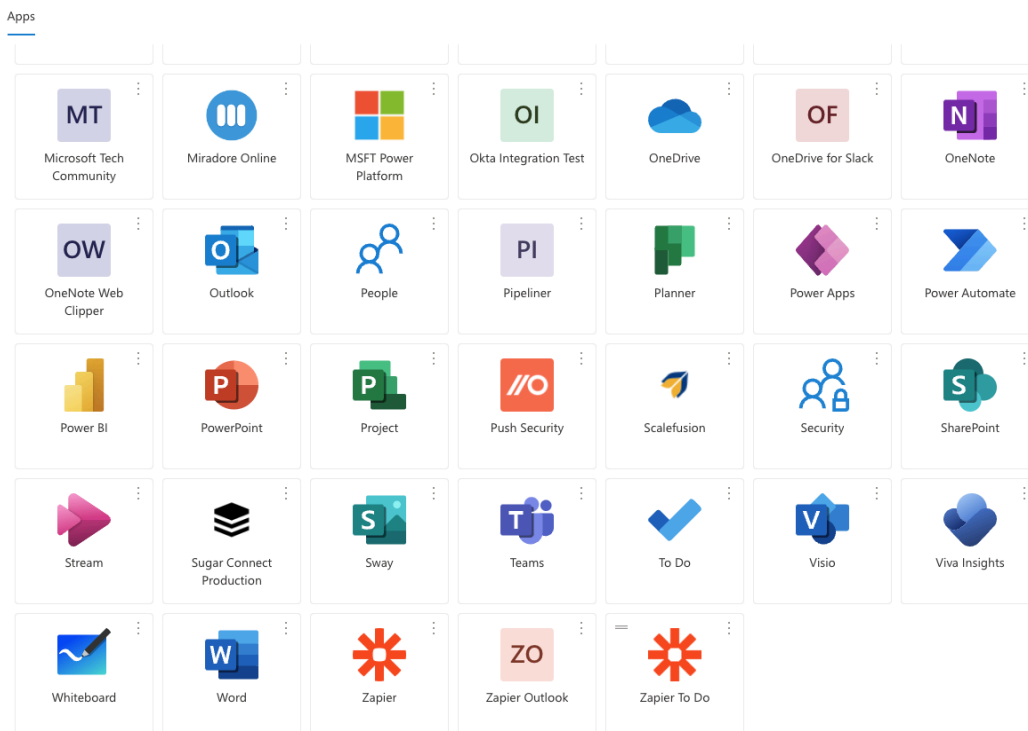
We want to use an automation app, but we'd prefer to be as stealthy as possible by also making it an evil twin integration. We'd like to see if the target user has existing integrations with any apps we'd like to use - especially an automation app for that high stealth option we mentioned above.

# Enumerating potential targets

We could perform something as simple as an email search for evidence of sign-ups, but that won't necessarily show us if actual OAuth integrations have been configured and what permissions are in use. What we really need is a way to perform an [OAuth token enumeration](#) attack.

## The first method: myapps.microsoft.com

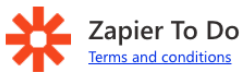
Make use of <https://myapps.microsoft.com> to see which apps are listed and which permissions have been granted. We can see Zapier is in use and the user has granted it access to their email and files, making it a great target.



Listing apps installed for the user from myapps.microsoft.com



← Return to Dashboard

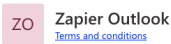


**Permissions you consented to**

Some apps use your data to customise the experience. You can revoke consent for these permissions, but you will be asked to grant it again when you launch the app.

<p>Microsoft Graph</p> <p>Sign you in and read your profile</p> <p>Have full access to your files</p> <p>Have full access to all files you have access to</p> <p>Maintain access to data you have given it access to</p>
--

Listing granted permissions for Zapier-related apps (1/2)



**Permissions you consented to**

Some apps use your data to customise the experience. You can revoke consent for these permissions, but you will be asked to grant it again when you launch the app.

Revoke consent

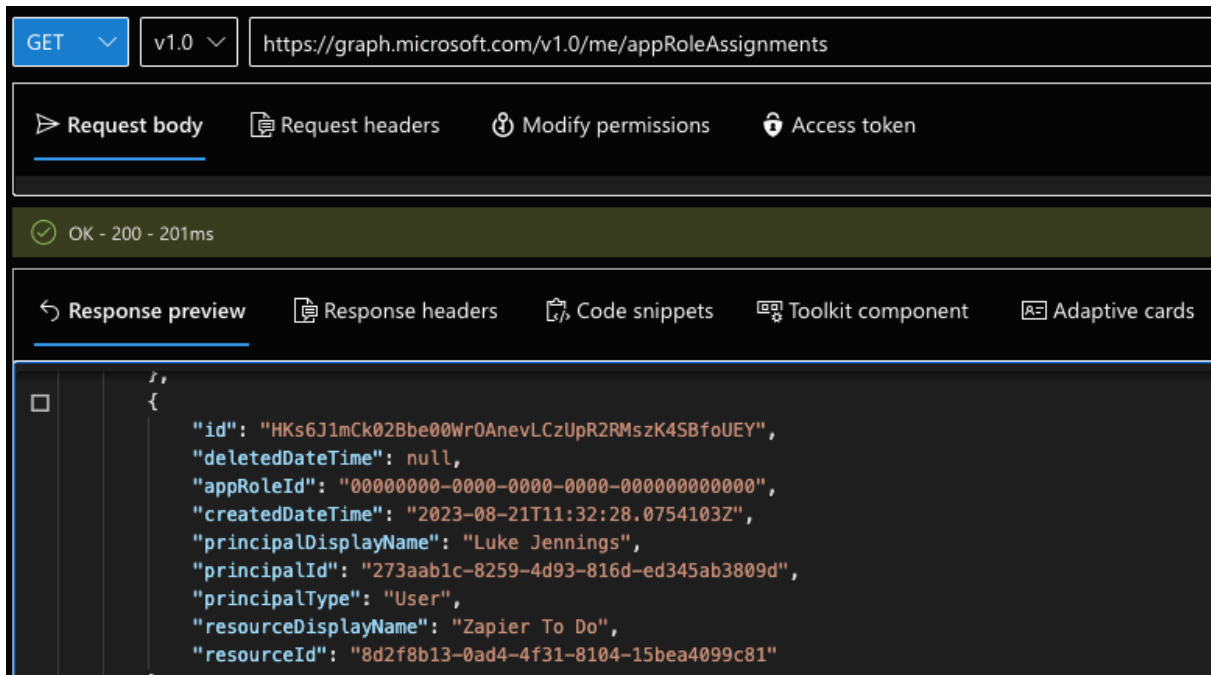
<p>Microsoft Graph</p> <p>Maintain access to data you have given it access to</p> <p>Sign you in and read your profile</p> <p>Have full access to your calendars</p> <p>Read and write to your and shared calendars</p> <p>Read and write access to your mail</p> <p>Read and write mail you can access</p> <p>Send mail as you</p> <p>Send mail on behalf of others or yourself</p> <p>Have full access of your contacts</p>
---

Listing granted permissions for Zapier-related apps (2/2)

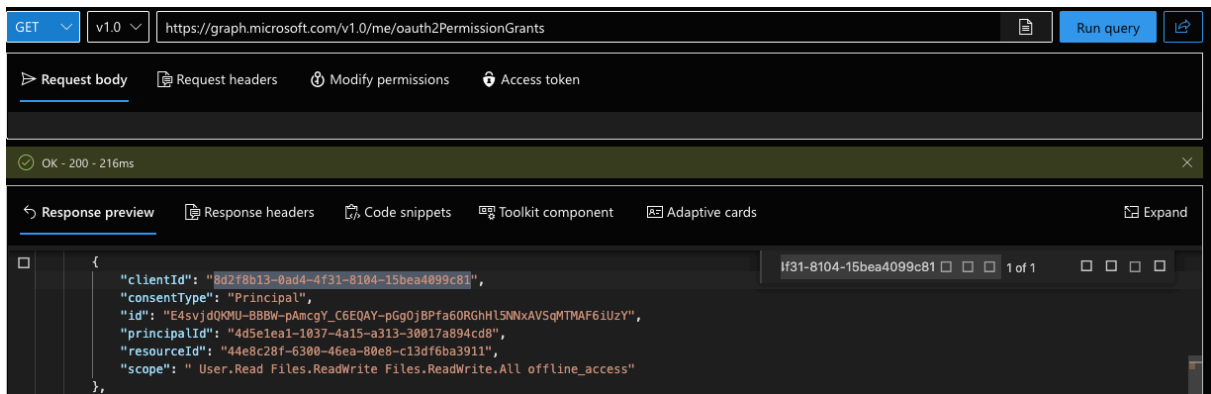
## The second method: Microsoft’s graph API

Microsoft’s graph API doesn’t make it possible to list out service principals without admin permissions, but you can enumerate individual OAuth permission grants and app role assignments for your own user account.

The client ID listed for permission grants is actually the tenant-specific service principal ID, rather than the globally unique OAuth app ID, but the app role assignments call gives us the app display name. We can match up the IDs from the app role assignments with the OAuth permission grants to see which permissions have been granted to the given app.



Listing app role assignments and finding a Zapier integration



Listing OAuth permission grants for the Zapier integration to confirm permissions

## Creating shadow workflows

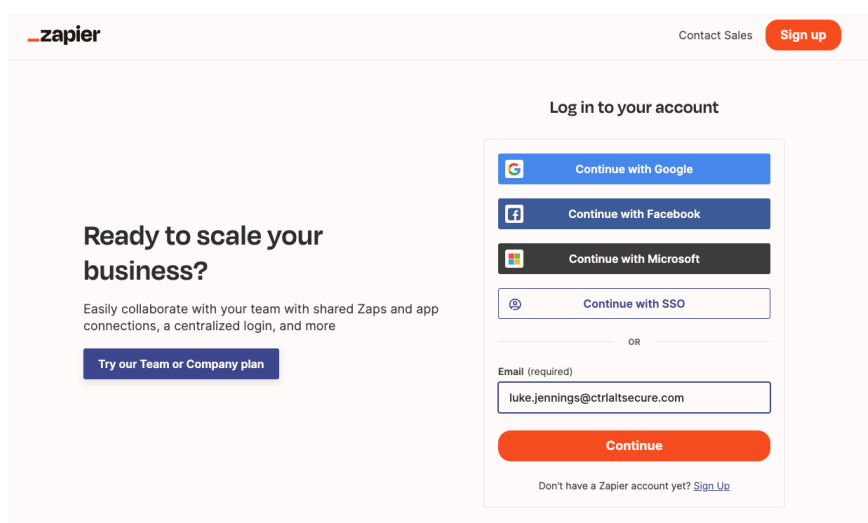
Ok, so we've figured out the user already makes use of Zapier and they've even already granted access to their email and files - that's a juicy target we can't turn down! So the next step is to create our own malicious workflows, or shadow workflows if you will, to get Zapier to do our dirty work for us.

First of all, we'll see if we can scope out the user's existing Zapier account to better understand the setup. Then we'll create a new Zapier account and link it to the target user's account that we've compromised.

Here's how that would work:

# Scope out the existing Zapier account

If the user uses SSO or social logins then we can login directly and, since we now control their Azure account, we can just log directly into their Zapier account!

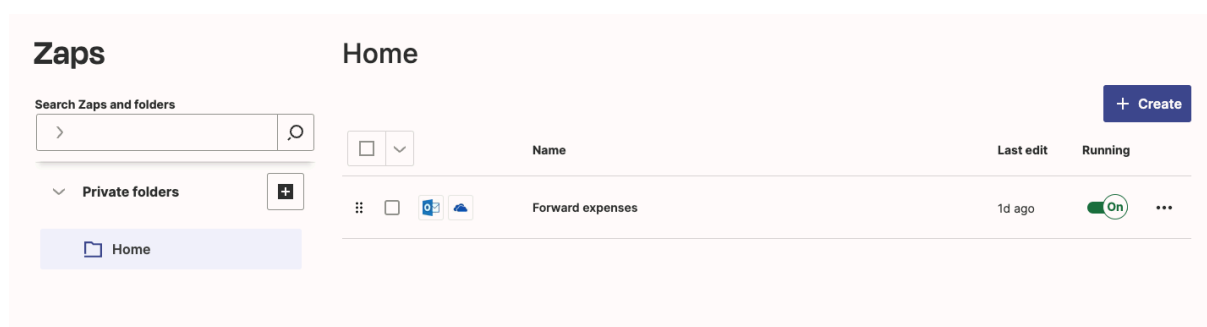


Login to Zapier via SSO or social login

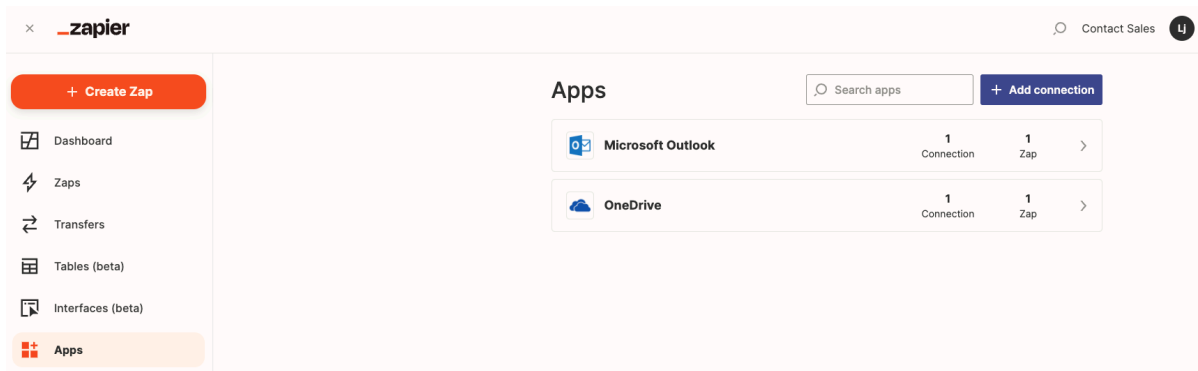
Alternatively, if they have created a standard password account, then we might already know the password if it's the same used for their Azure account. Otherwise, we could potentially make use of an [account recovery](#) attack to gain access.

Once we have logged into their account, we can see their existing workflows and integrations. Technically, we could backdoor these or create new ones - a form of an [abuse existing OAuth integrations](#) attack. However, that runs the risk of the user discovering our shadow workflows and also almost certainly being locked out of the account during the next password change.

Instead, we can stick to an evil twin integration from our own Zapier account, which we'll create later.



The user's existing workflows, or "Zaps" in Zapier terminology



The user's connected apps, showing Outlook and OneDrive have already been integrated

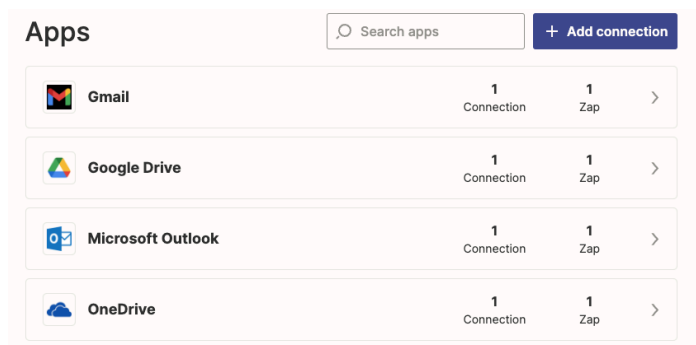
Now we can see what the user was actually using Zapier for — they've set up an integration with both Outlook and OneDrive so they can forward emails related to their business expenses to a folder in their OneDrive. Probably a time-saving hack, which we can take advantage of since it won't be unusual to see Zapier regularly accessing their Outlook and OneDrive. That means our attack will be extra stealthy.

## Create our own malicious Zapier account

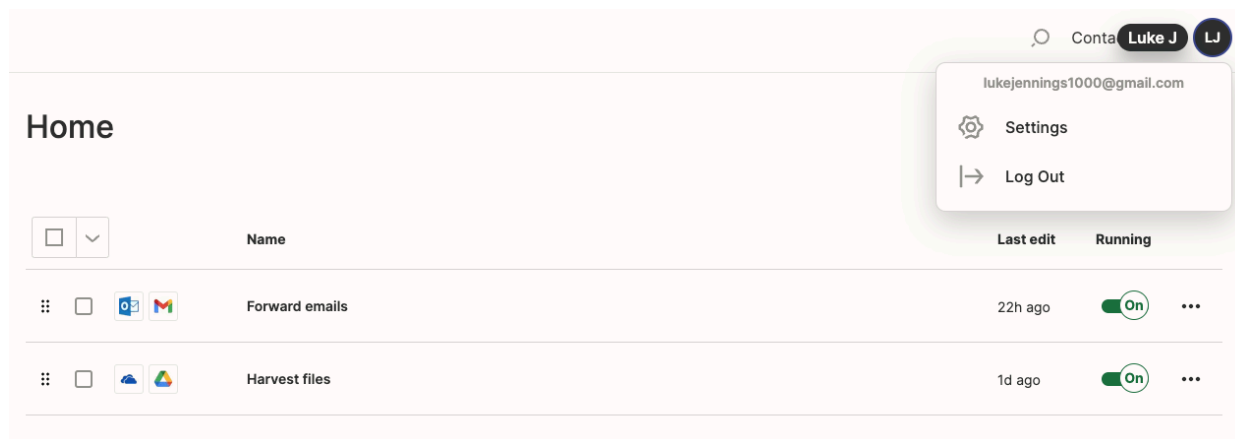
Given in this case we, at least temporarily, control the user's Azure account there is nothing stopping us connecting this to our own malicious Zapier account completely separately from the user's legitimate Zapier account. We then maintain full control over the Zapier account and the user will not be able to discover our shadow workflows as they won't have any knowledge of our Zapier account:

Let's create our own shadow workflows:

- One that sends every new OneDrive file to our own separate Google Drive account. This allows us to maintain a complete view of the user's files into the future.
- And one to forward every new Outlook email to our own GMail account.



We have connected the user's Outlook and OneDrive, as well as our own GMail and Google Drive for exfiltration



Shadow workflows we have created to monitor the target user's emails and files

We can now see we are logged in with a separate GMail account, but have created shadow workflows to forward emails from the user's Outlook to our GMail account and harvest files from their OneDrive to our Google Drive.

The major benefit of creating our own Zapier account for an evil twin integration is that once we are locked out of the target user's account via a password change or otherwise, not only do our existing shadow workflows continue to operate via OAuth, but we are able to create new shadow workflows and reuse the existing OAuth connections. That's the power of having full control of the Zapier account.

One small downside to this approach is that creating the new OAuth integrations inside a new Zapier account generates an interactive login event for the Zapier integrations from the adversary's IP address. This occurs due to creating integrations from the new Zapier account, but because the user has already consented to all the relevant permissions for Zapier's own OAuth apps there are no audit logs for new consents or applications, just the login event itself.

However, determining that a successful login to an app a user legitimately uses is actually malicious in this case is obviously extremely difficult to build detection logic for.

<u>User sign-ins (interactive)</u>	User sign-ins (non-interactive)	Service principal sign-ins	Managed identity sign-ins	
Date	Request ID	User	Application	Status
24/08/2023, 12:21:03	3f8018e6-0eef-4235-b802-1c4c44331e00	Luke Jennings	Zapier Outlook	Success
24/08/2023, 12:17:05	9c01a6e7-1ecb-4bc9-979e-624561ca3d...	Luke Jennings	Zapier To Do	Success

Interactive sign-in events caused by the new integrations with the new Zapier account

Beyond the initial login events, the only evidence of malicious activity in the future will be from the activity logs showing the actions conducted by our shadow workflows every time they are triggered to run.

For example, the following screenshots show that the Zapier Todo app (ClientAppId 29246358-1970-4d6d-bc75-acf34edc758b) has been seen both uploading a file and downloading a file:

**IP Address**  
52.1.205.184

**Users**  
luke.jennings@ctrlaltsecure.com

**Activity**  
Downloaded file

**Item**  
my secret file.docx

**Details**  
Downloaded from "Documents"

**AppAccessContext**

```
{  
  "ClientAppId": "29246358-1970-4d6d-bc75-acf34edc758b",  
  "CorrelationId": "3058d3a0-20e5-7000-4776-88ab9d5d06da"  
}
```

Audit log showing the Zapier integration downloading a file - this is a result of the shadow workflow

**IP Address**  
20.190.151.160

**Users**  
luke.jennings@ctrlaltsecure.com

**Activity**  
Uploaded file

**Item**  
~tmp7F\_expenses\_AAMkAGU0NWMxODRiLTg1OTktNDFiYS1hM2I3LTcxYTE1NTJhNj  
ljNgBGAAAAAAp6WWuehUHSJUnlGpOfGnzBwCd9IPOI-  
PXSpUOY\_MIE5tVAAAAAEMAACd9IPOI-PXSpUOY\_MIE5tVAAGWc3iDAAA=.html

**Details**  
Uploaded to "Documents/Expenses"

**AppAccessContext**

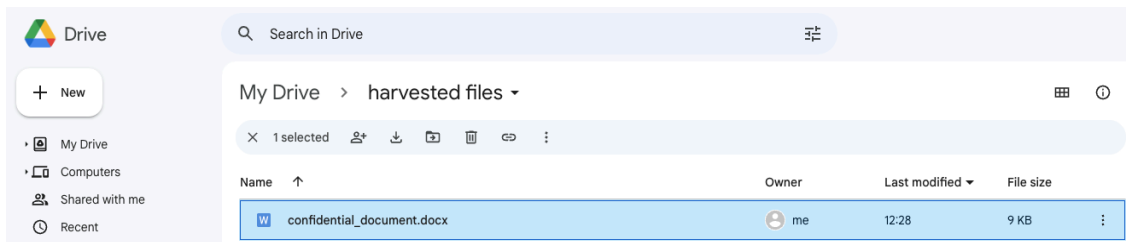
```
{  
  "APIId": "00000003-0000-0000-c000-000000000000",  
  "ClientAppId": "29246358-1970-4d6d-bc75-acf34edc758b",  
  "CorrelationId": "a4c8ef5d-d760-4c3e-b62e-6a5105b1c1e6",  
  "TokenIssuedAtTime": "2023-08-23T16:45:36",  
  "UniqueTokenId": "saFy4KHyZUKgD8J5ZATnAA"  
}
```

Audit log showing the Zapier integration uploading a file - this is the result of the legitimate integration that forwards business expense emails

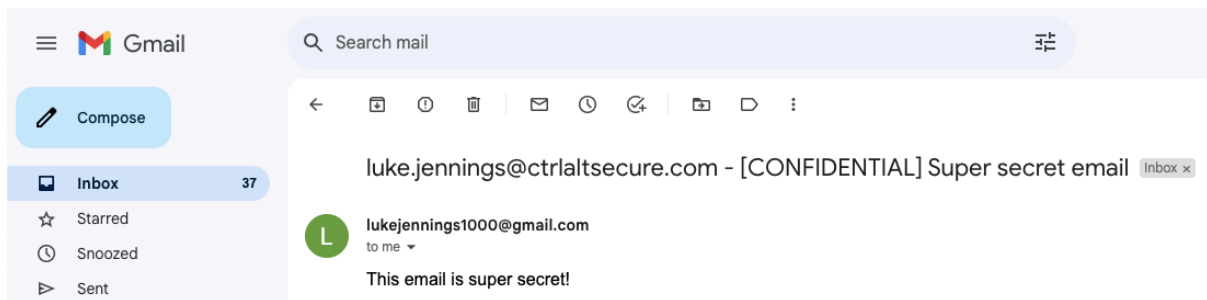
The file upload in this case relates to the legitimate workflow and the file download relates to the shadow workflow. The IP addresses relate to Zapier's legitimate infrastructure so really only a very thorough and specific investigation is going to be able to uncover that one of these events is malicious.

# Profit

Now we just need to sit back and let our shadow workflows do the work for us, 24/7 and from Zapier's infrastructure via a legitimate OAuth integration. Here we can see files the user created in OneDrive and emails they received in Outlook mirrored to our own GMail and Google Drive via the magic of shadow workflows.



Confidential document from OneDrive appears in Google Drive



Confidential email from Outlook appears in Gmail

# Impact

We've covered a lot of ground here so it's worth taking a step back and considering the key impact points of this attack chain:

- An adversary who has gained (temporary) access to a user account that supports OAuth integrations can use shadow workflows to execute malicious actions and to maintain persistence. This access will continue even if the user changes their password or resets MFA
- Not only do existing shadow workflows continue to work after password changes, an adversary can continue to create new ones and reuse the existing integrations.
- Any relevant logs will show access via legitimate IP addresses and OAuth integrations for SaaS automation apps
- Automation apps are so flexible that an adversary can do pretty much anything - it's basically the offensive PowerShell of the SaaS world. Just some examples:
  - Monitor all emails and files the user creates
  - Delete email security alerts before the user sees them
  - Intercept password reset and passwordless login emails to access other apps
  - Monitor instant messaging apps and use it to send targeted internal social engineering emails
- If targeted users are already using automation apps legitimately, it's even more stealthy - you won't even see any new integrations or permission grants appear as the user will have already granted these legitimately.
- If admin consent has been granted to the automation app, any user can be targeted without generating new permission grant logs even if they have never used the app.

## The attacker maintains control even after compromise is detected

We've seen how two new SaaS-focused attack techniques can be combined into one more effective attack chain — in this case, a particularly nasty and stealthy persistence technique. This shows how even if a user compromise is detected very early, with password and MFA resets immediately issued, adversaries can maintain control over the account regardless.

This shows how even legitimate SaaS applications have incredibly powerful offensive use cases and very careful attention needs to be paid to integrations with highly sensitive permissions, even when they are approved and vetted applications. Incident response teams especially need to be well aware of these techniques when investigating potential user account compromises as persistence approaches can extend much further than endpoint implants and stolen passwords.



# Conclusion

If you've made it to the end, thanks for sticking with us – we hope you've found this paper useful!

We're thrilled that many organizations are making use of the SaaS Attacks Matrix, and we hope it continues to be a useful tool for red teamers and pentesters alike, helping those defending their organizations against these attacks better understand their potential areas of weakness so they can shore up their defenses.


We'll be continuing to publish research on these techniques and demonstrating how they can be chained together. Follow us on [LinkedIn](#) and [X](#) to see the latest.

As we mentioned earlier, the best way to defend against these types of attacks is to have visibility into your entire identity attack surface and then focus on hardening those accounts. You'll also, of course, want to remove apps and accounts that are too high-risk, request excessive permissions to your data or systems, and any that are dormant, unused and cause unnecessary risk.

Push gives you the tools to discover and harden your user accounts, but [we also provide advanced browser-based capabilities to not just detect, but intercept and block identity attacks as they happen](#). This includes features like straight up stopping users from entering their passwords into phishing pages (even if the page has never been flagged as malicious previously) and detecting advanced phishing tools at the point they render in the user's browser.

Turning the browser into a source of security telemetry and control enforcement, in the same way that the introduction of EDR earmarked a paradigm shift for endpoint-based malware detection and response, is a game changer for security teams looking to stop identity attacks.

So [grab a quick demo](#) with our team to see how we can help you to overcome identity security threats.



“I think it's one of those things that went from being a theoretical problem for a lot of people, so CISOs weren't spending money on it, and then all of a sudden we're starting to see issues, right? Like the big Snowflake disaster, which has whacked all sorts of global brands. And they're sort of scrambling for solutions and that's what they're starting to look at, which is this new generation of web browser plugins... This is the new technological approach to solving some pretty serious security challenges... it's a control that everybody will have to have.”

**Patrick Gray, [Risky Biz podcast](#), Push Security advisor**

