Phishing Slack for persistence and lateral movement

By Luke Jennings

Archived: 2026-04-05 20:39:21 UTC

This is the fourth post in a series on attack chains formed by combining techniques in the SaaS attack matrix and the second post of two focused on attacking instant messaging applications with Slack as the primary example.

The previous post focused on external attackers gaining an initial foothold during the initial access phase of the kill chain. In this post we'll be focusing on persistence and lateral movement for an attacker that has already gained a foothold on a Slack tenant by compromising an internal account.

We'll build on the techniques in the previous post as well as introducing more and so will cover the following SaaS attack techniques, including chaining them together:

- SAT1018 - IM phishing
- SAT1019 - IM user spoofing
- SAT1036 - OAuth system integrations
- SAT1033 - Shadow workflows

Why focus on instant messengers?

If you've just read the previous post, you can skip this introductory piece and jump straight to the next section.

They aren't new, however, the original focus of IM apps was on internal communication and phishing and social engineering attacks are often external. Email remained the standards-based protocol that enabled external communication no matter what email vendor was in use. In recent years, however, instant messengers (IM) have become the primary method of communication for many businesses. I wanted to focus on IM here because if that's where employees are communicating, it's the best place to launch attacks against them. Even better, there's a history of users placing a higher degree of trust in IM platforms than email, so it becomes a potentially easy target.

While IM platforms were initially used solely for internal communications, organizations quickly realized that IM platforms could be used to communicate with external groups, individuals, freelancers, and contractors, with the hope of fewer emails and more instant communications.

We now have Slack Connect and Microsoft Teams external access to support this, with Slack Connect introduced in June 2020 and Teams introducing it in January 2022. This external access has increased the attack surface of these platforms considerably.

Despite decades of security research, email security appliances and user security training, email-based phishing and social engineering is still commonly successful. Now we have instant messenger platforms with:

- Richer functionality than email,
- Lacking centralized security gateways and other security controls common to email and
- Unfamiliar as a threat vector to your average user compared with email.

There's also a sense of urgency associated with IM messages due to the conversational nature compared with emails. Combined with a history of increased trust, we have the ingredients for increased social engineering success.

There's been an uptick recently in IM-based phishing research and real-world attacks, particularly for Microsoft Teams. For example, check out the great research from JumpSec on bypassing attachment protection for external Teams messages, the offensive tool TeamsPhisher and attacks distributing DarkGate malware via Teams.

However, in this article we'll focus on a few techniques specific to Slack.

Slack apps - spoofing and persistence

In the previous post, we covered user spoofing and link preview spoofing attacks that can be conducted externally. But do we have any other options available once on the inside that wouldn't be available to us externally?

What happens if you compromise a Slack account and then want to persist and/or move laterally? Ordinarily, you can maintain access until session expiry or a password change is forced or the account is deactivated/deleted. For further actual impact, you could silently read messages as the user continues to operate their account but if you start sending out malicious links to other targets, in an attempt to move laterally, then the real user is probably going to become aware of the compromise very quickly from seeing the malicious messages in their own chat client.

Alternatively, what happens in a situation where a disgruntled employee is let go and their account is terminated? Could they maintain some level of access and use it maliciously?

One key feature that some IM apps like Slack have is app integrations to allow bots and other functionality, usually using OAuth under the hood. This allows very useful functionality for users, but also opens up a whole new angle for persistence and spoofing. In Slack's case, its separation of user tokens and bot tokens allows for particularly interesting spoofing and persistence capabilities, which we'll come to later.

We could probably write several posts on OAuth apps alone. In fact, we've written about using OAuth for persistence more generally before. However, in this case we are going to focus on a couple examples of using a legitimate Slack app maliciously.

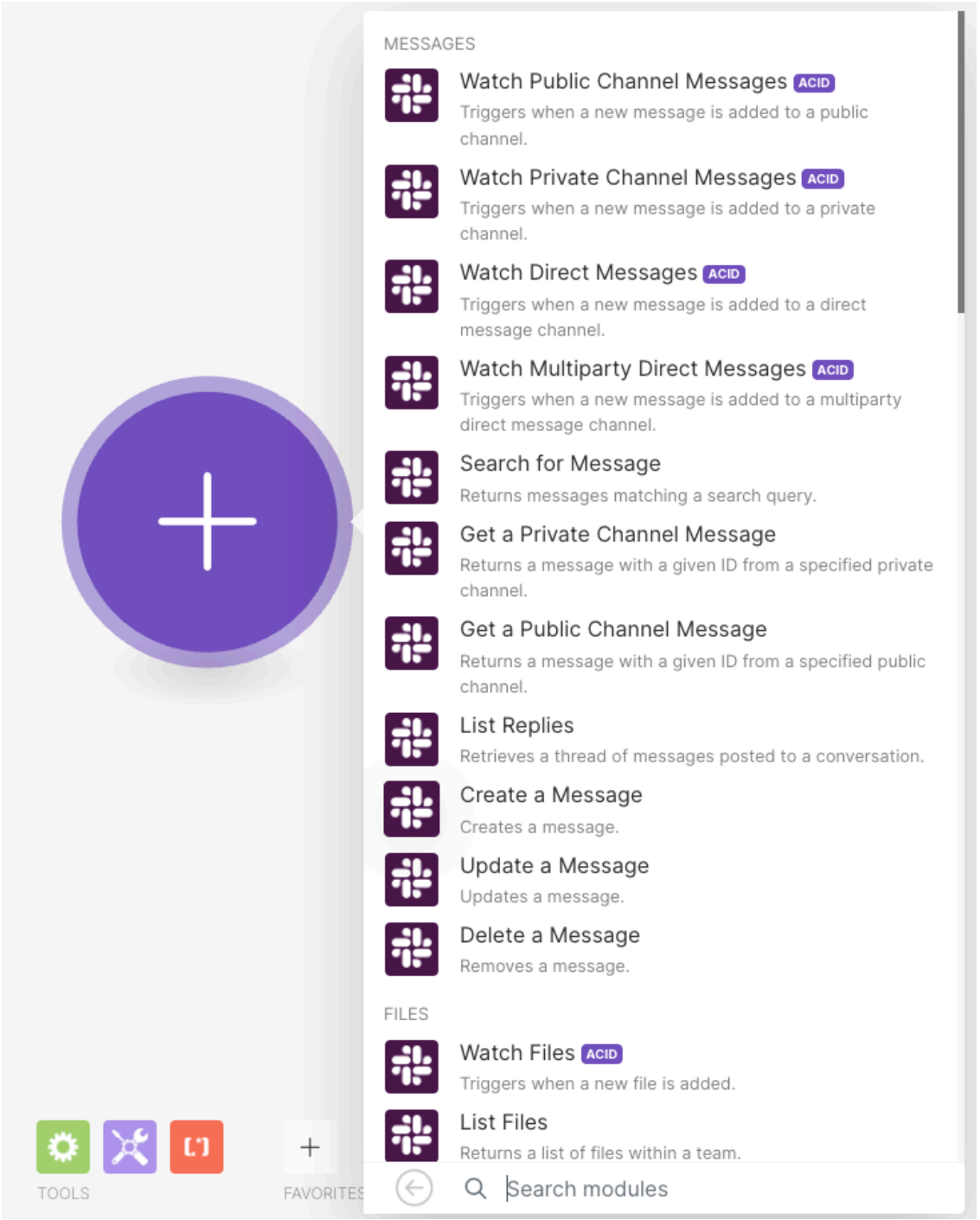
In a previous blog post in this series, we spoke about shadow workflows using SaaS automation apps. We're going to follow this theme again here and show how they can also be used with Slack. Previously, we used Zapier as our

automation app example, but this time we are going to use make.com.

Persistent spoofing

We'll show here how you can connect make.com to a Slack account you control and then maintain persistence, both as that user and partial access even if the account is deactivated or deleted, by using bot tokens. This is especially important in a disgruntled employee scenario as they could use this to maintain some level of access to Slack even if they were fired and had their account deleted.

If we create a make.com account and click to create a new scenario, we can select Slack from the long list of integration possibilities. We'll then be prompted to pick a specific Slack module. In more complicated scenarios, these can be chained together to take actions on events, but in this case we are going to create a simple scenario with just one module used to send a custom Slack message.

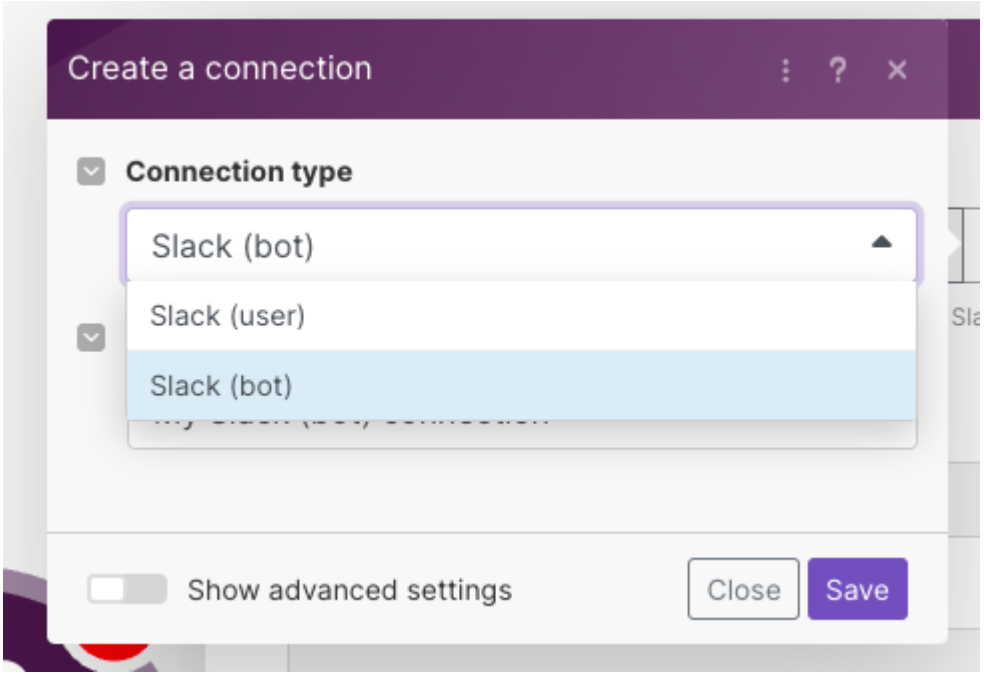


Creating a new scenario in make.com and picking a Slack module

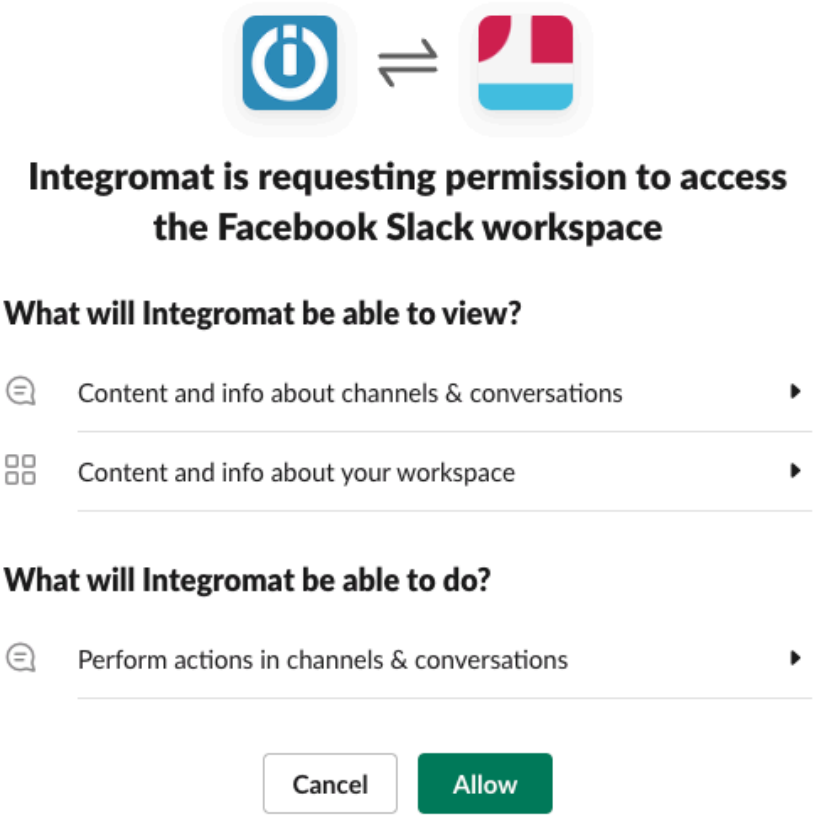
If we select the module “Create a Message” we’ll be prompted to select a Slack connection to use and then fill out the other details for the module. Since we haven’t already created a Slack connection, we’ll be prompted to create a new one. For this module, we have the option of creating either a user token or a bot token.

A user token has full capabilities and will continue to operate in the event of a password change. However, if the user account is deactivated or deleted then it will cease to work. In contrast, the bot connection is limited in capabilities compared to a full user token, but the advantage is that it will continue to operate even if the user account is deactivated or deleted.

This means gaining even temporary control of a Slack account, either through a user compromise or by being a disgruntled employee (or fired employee), could enable the permanent ability to spoof messages unless the entire app is revoked from Slack. Even with the high bar set by shadow workflows, that's a pretty epic level of persistence! So, we're going to select the bot token for this example:



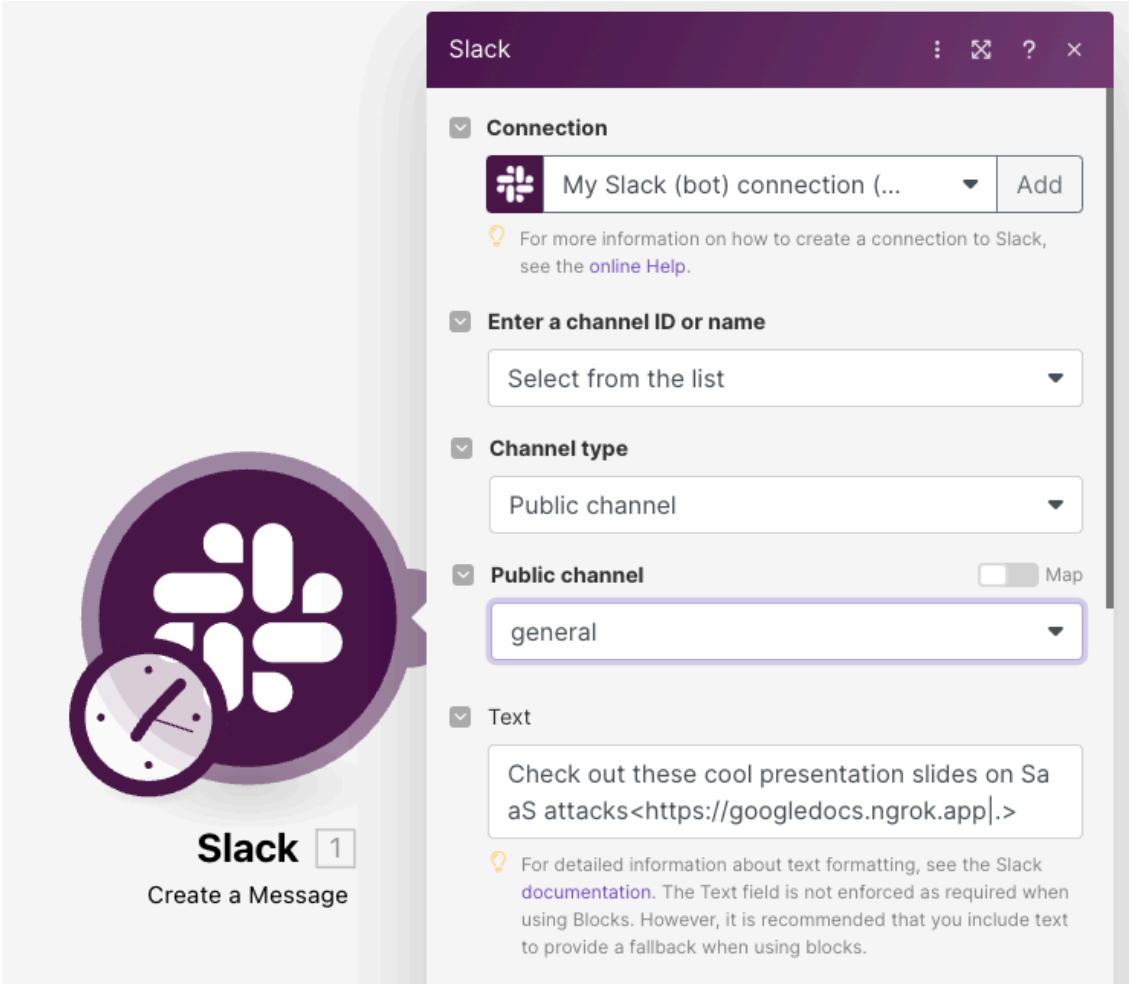
Picking a Slack bot connection when making the Slack integration



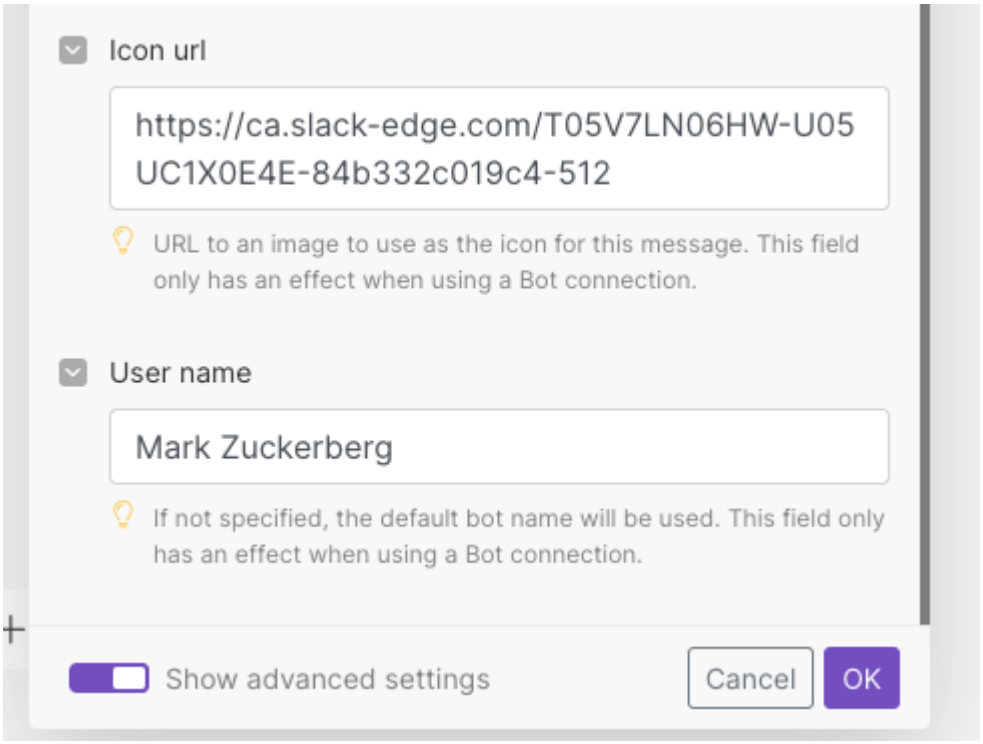
Accepting the OAuth2 permissions request for the app. Make.com still uses an app called “Integromat”, a legacy name for the company

Now that we’ve finished setting up the bot connection, we can configure the specifics for the module itself. In this case, we’ll demonstrate using it to send the same type of spoofed message we covered in the previous post, only it’ll be from a bot account.

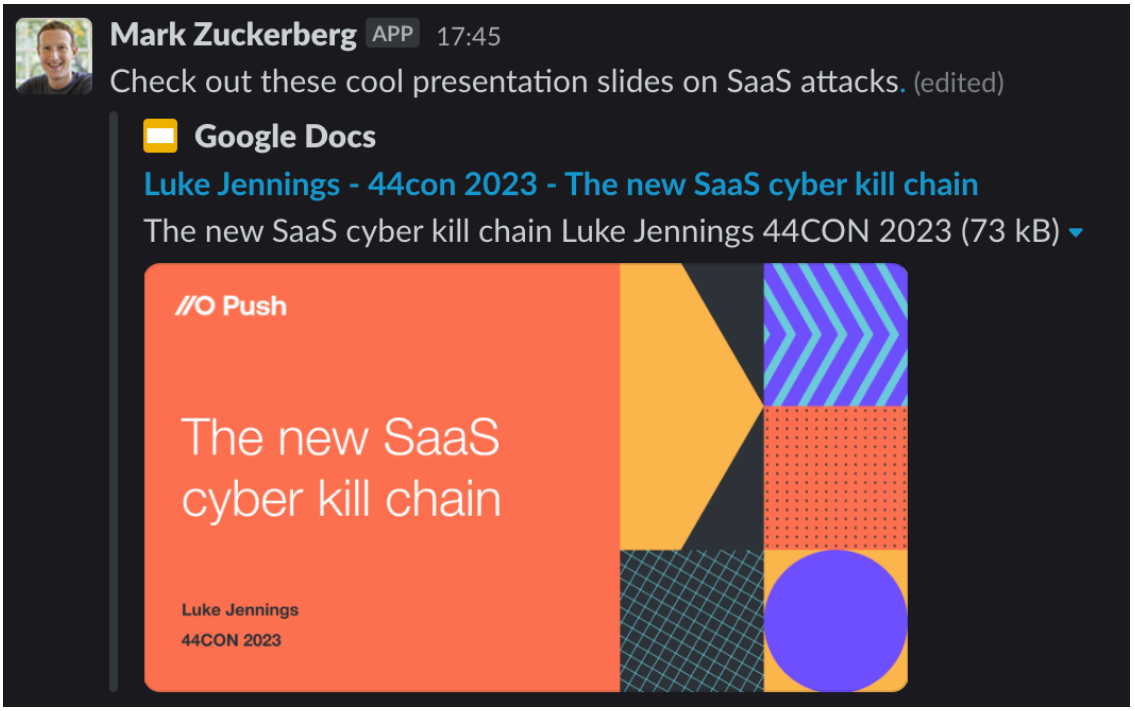
By default, it’ll use the name and icon of the Slack app, in this case Integromat (Make.com’s former name). Alternatively, we can choose to override this, which we will do in this case to mirror the user spoofing attacks we covered earlier. The only difference to a normal user message is there will be a small “APP” icon after the user.



Setting the module to send a message to a public slack channel as the bot account



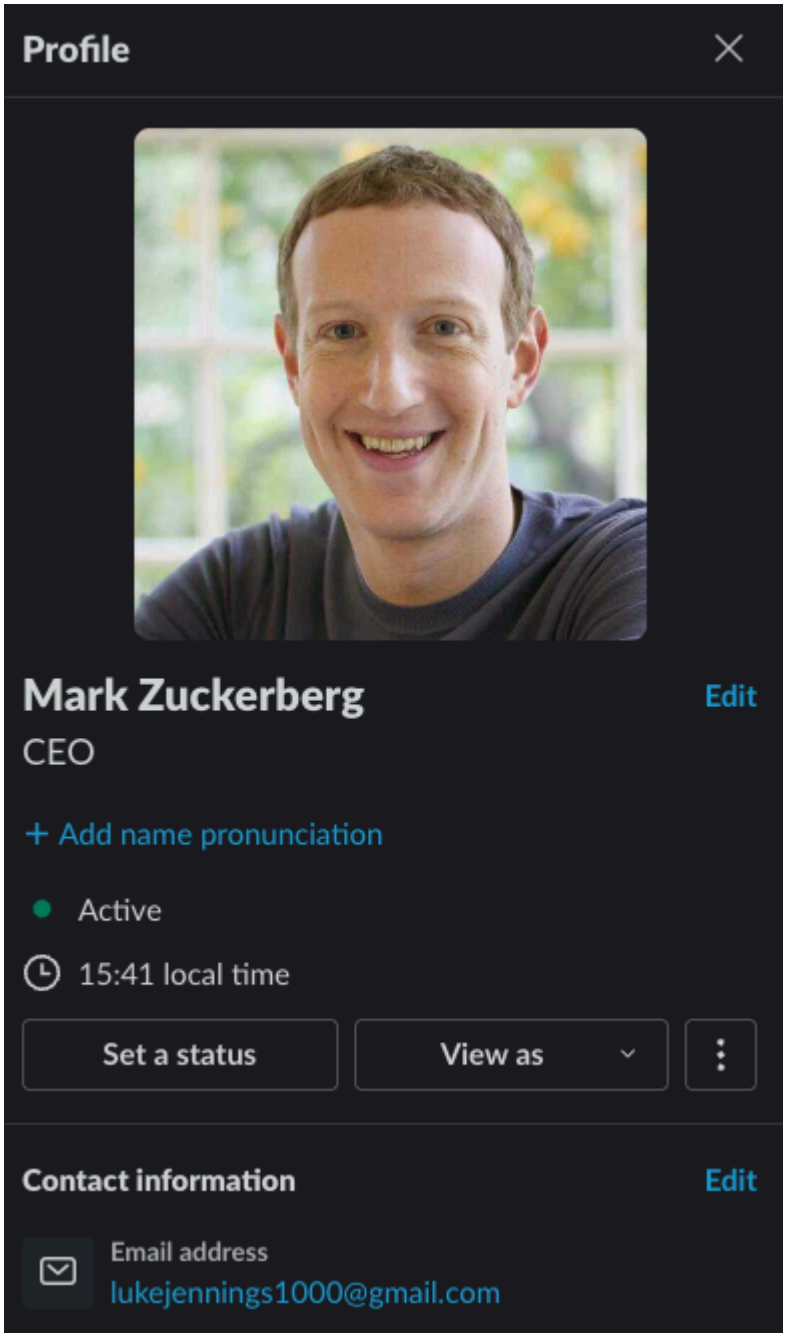
Configuring the bot to use a different name and photo in order to spoof the user



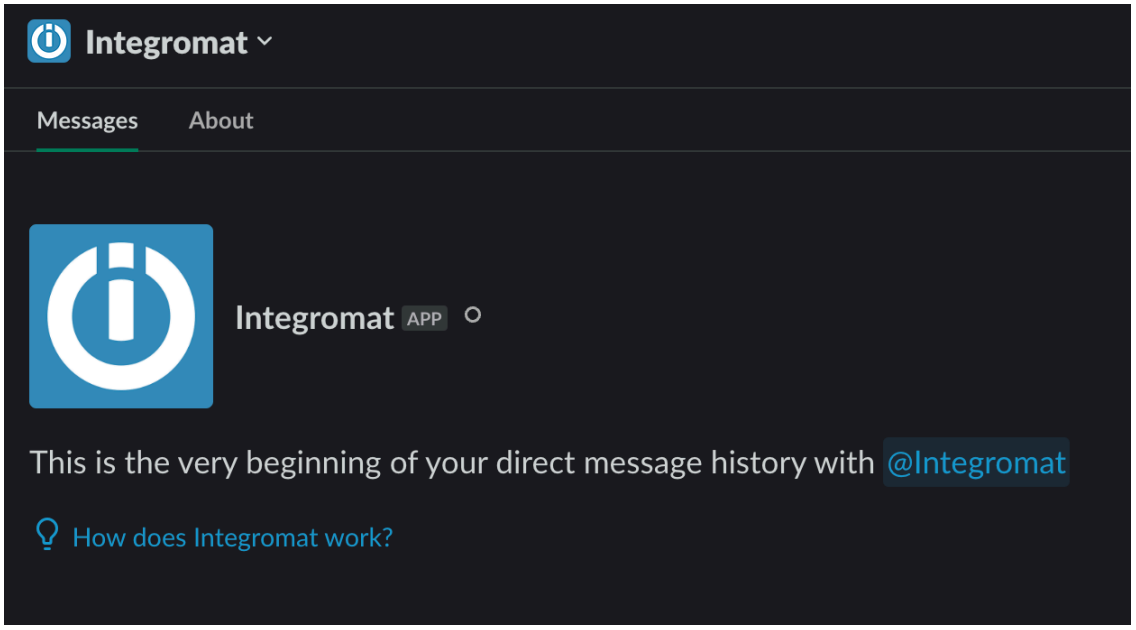
The phishing message sent once the scenario is run, which is almost identical in appearance to what we saw previously except for “APP” after the name

The other great advantage with this is that it’s difficult to see which user is actually responsible for the spoofing. If a compromised user account is used to send spoofed messages, not only may the real employee see the messages and alert security, but if the messages are investigated by a target or the security team, it’s quick to click on the user and see the real email address associated with the account.

However, when it's done with a bot token for an app, *you can only see the Slack app that was responsible, not the actual user account it originated from*:



Sending a spoofed message without the use of a bot token allows someone to click on the user and see the original email address



Sending a spoofed message using a bot token from make.com takes the user to the Integromat app if they click on the user, so they can't easily see who was responsible

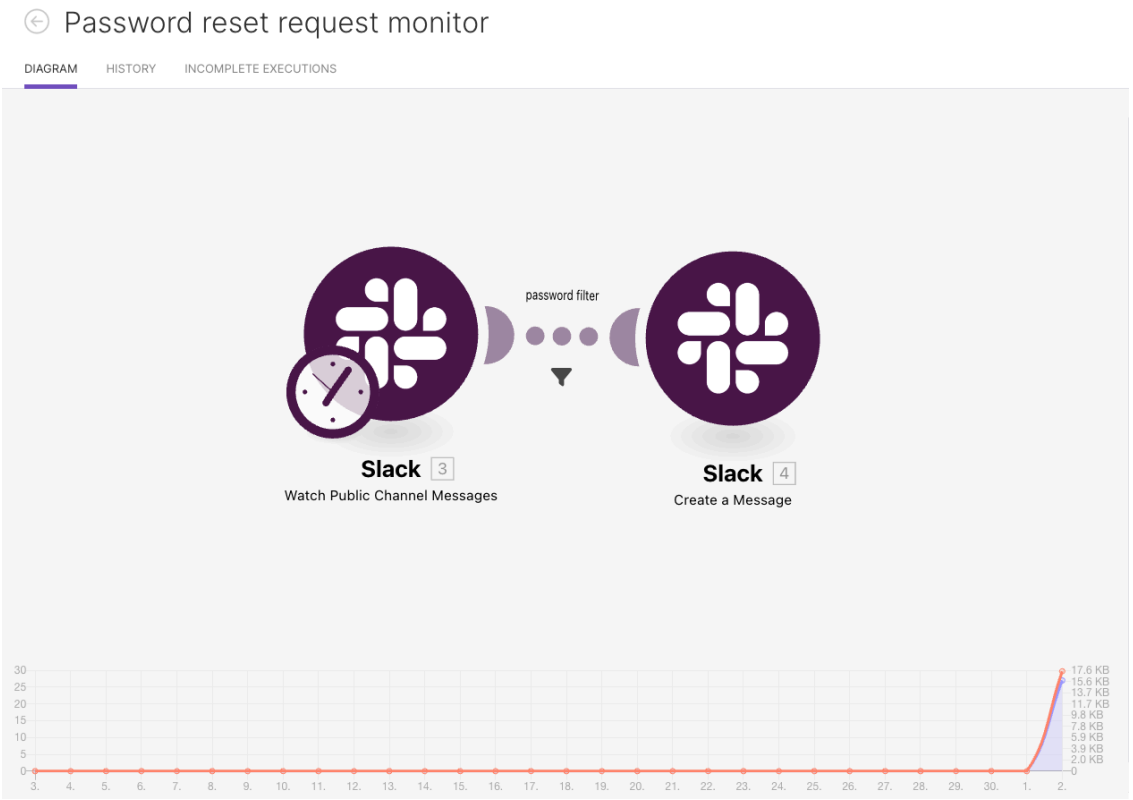
Automated phishing replies

Ok, so we've just seen how you can internally spoof a message via a Slack app in a way that's harder to track back to the original compromised user account and also achieve persistence at the same time. Pretty neat! But can we do more?

One of the great features of IM apps is the fact they are…well…instant! By making a slightly more sophisticated scenario with make.com, we can monitor public channels for messages that meet certain criteria and then immediately spoof a target phishing link as a reply. Phishing where the target is the one to reach out originally is much more likely to be successful as it's more like a watering hole attack - the phishing message itself won't be seen as unsolicited.

For example, let's consider a scenario where someone has forgotten their password, or some other common IT support request, and they raise a question on a Slack channel about it. We could monitor for that and automatically respond.

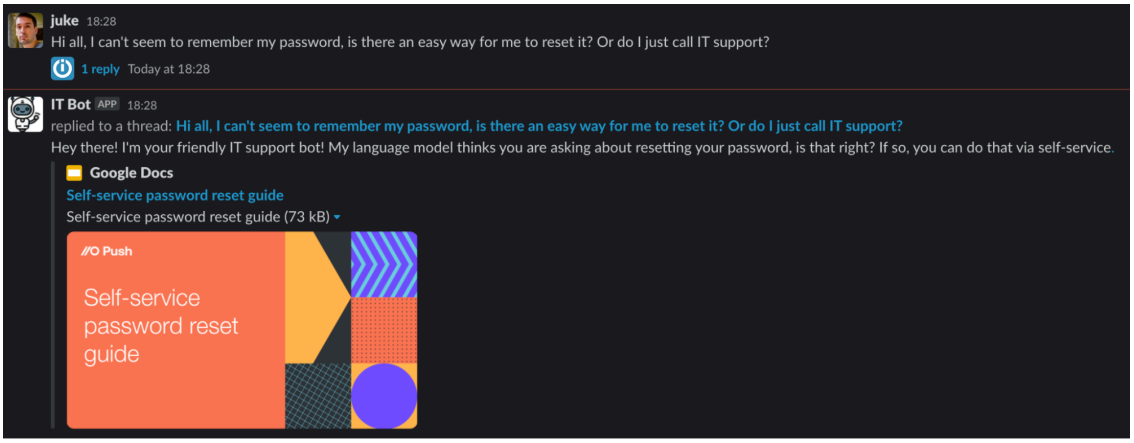
One caveat here is make.com requires we use a user token for the message monitoring part and therefore this attack couldn't survive a deactivated/deleted Slack user account. However, it will still survive password changes and so is still a useful persistence option too. Additionally, the bot token can still be used for the message sending component in order to mask the source of the attack as above.



Creating a scenario to monitor public channel messages for certain keywords in order to reply with phishing messages

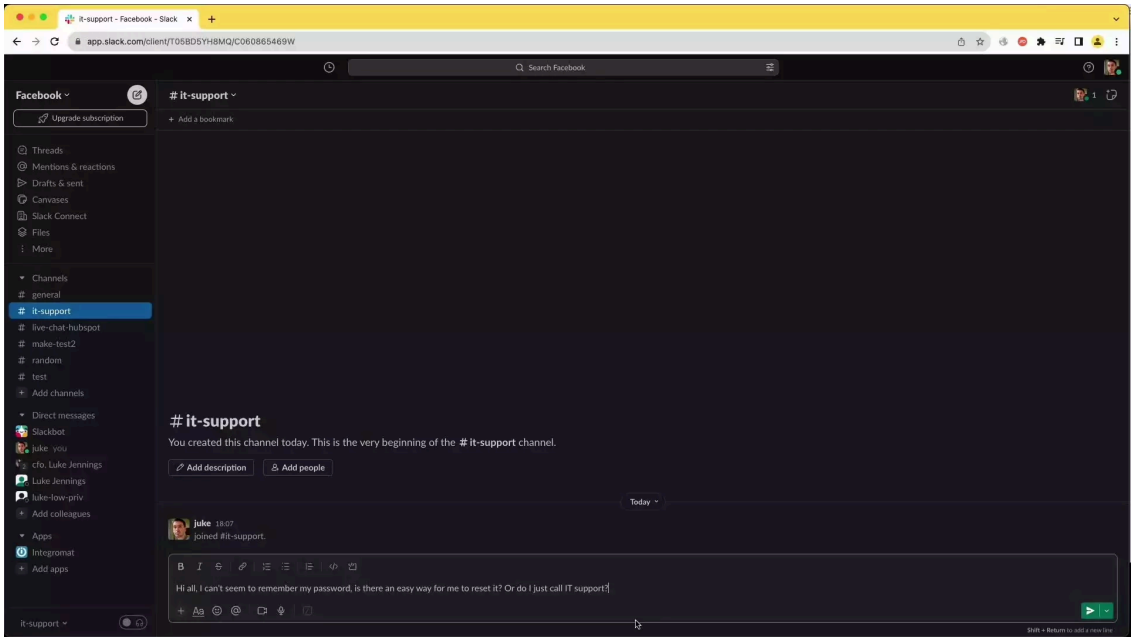
In this case, we have configured a Slack module to watch public channel messages using a user token and apply a filter on those containing the words “password” and “reset”. If that is the case, we then trigger a spoofed threaded reply using the bot token and impersonating an “IT bot” and giving a link to documentation for how to perform a self-service password request.

This makes use of the same link preview spoofing techniques we covered in the previous post and the actual link will present a fake Google login page to harvest credentials.



Our make.com scenario automatically responding in a thread with a targeted phishing link from a spoofed bot user, using a spoofed link preview

Here's a quick video demonstrating this combination of user spoofing, link preview spoofing and a shadow workflow in action:



To summarize, heres a diagram to show how this all fits together:

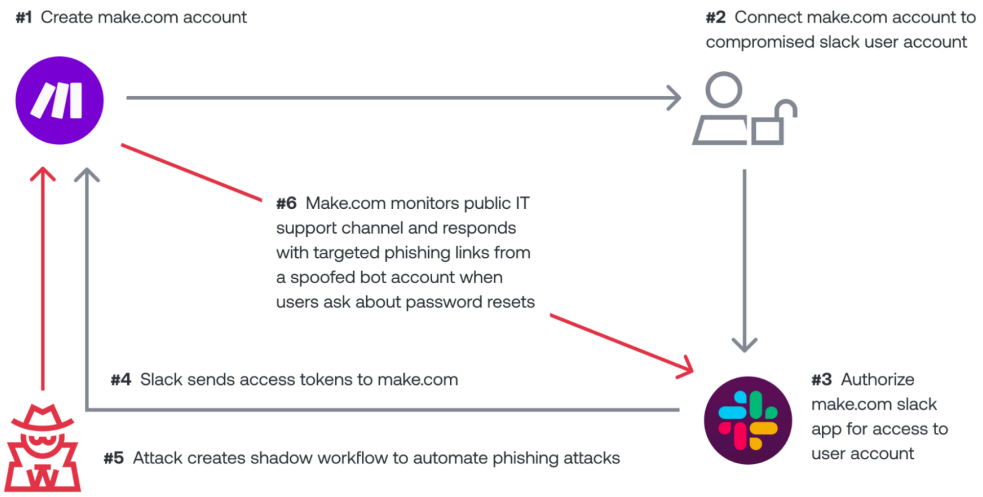


Diagram showing the connections between the attacker, compromised Slack account and make.com

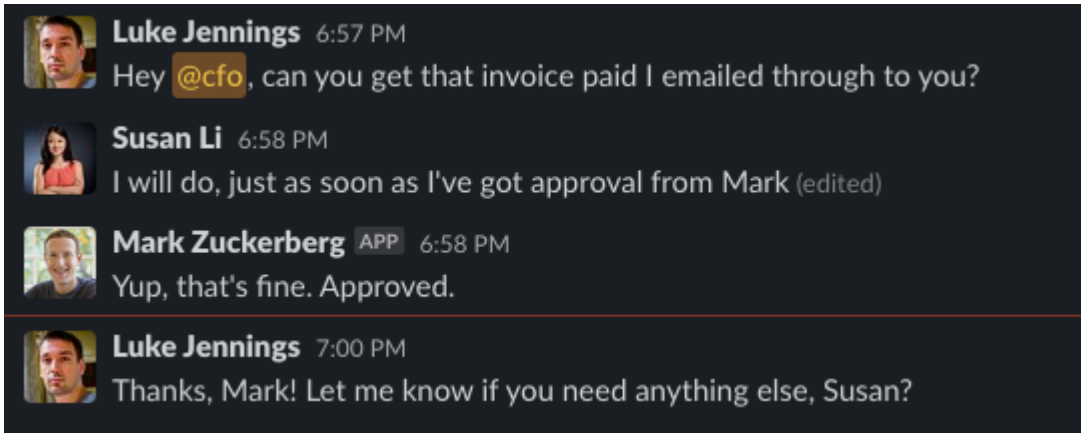
Multi-party spoofing

Another great possibility provided from using Slack apps and bot tokens for spoofing is the ability to spoof inline with existing communications as multiple parties. Ordinarily, if a Slack user kept changing their name, handle and photo for spoofing internally, Slack would change all existing messages to the latest profile data every time. That makes it hard to spoof multiple identities in short time windows and so an attacker could only really spoof one

person at a time. However, with Slack apps you can inject messages as different people at different points of a conversation using bot tokens.

Consider the following example, where I'm using my own internal account to message the CFO about paying a malicious invoice that I have hypothetically raised. Perhaps they then indicate approval is needed from another party, in this case the CEO. Similarly, this might be a common process for access requests requiring manager approval and many other business processes.

In this case, I'm able to quickly spoof a message as another user to act as the approval in a manner that is pretty sneaky. The only giveaway at first glance is the "APP" tag after the spoofed message.



Multi-party spoofing of messages for advanced social engineering internally on Slack

This is just one example but the ability to spoof multiple identities simultaneously from just one compromised account on what is usually seen as a trusted internal communications system really opens up a ton of possibilities for social engineering attacks focused on lateral movement.

See more original research and technical content from Push

Impact

After two whole posts on attacking Slack, covering both external attacks during the initial access phase and internal attacks in the persistence and lateral movement phases, we've covered a serious amount of ground!

It's worth taking a step back and considering the key impact points:

- IM apps like Slack are now external phishing and social engineering vectors, not just internal ones
- User spoofing can be used in novel ways to enhance social engineering that employees may not be familiar with
- Link spoofing techniques can make phishing links much harder to spot and so increase social engineering success
- Malicious Slack messages can be modified later to replace the phishing link to cover up the attack

- Slack apps, and especially bot tokens, can be used for very effective persistence techniques. Some examples:
    - It’s possible to read all messages even after a compromised user changes their password
    - It’s possible to send (and spoof) messages even if the compromised user account is deleted (e.g. a disgruntled employee who is fired)
- Slack apps and shadow workflows can be used to conduct some fairly advanced social engineering attacks once an attack has a foothold on a Slack tenant. Some examples:
    - Automatically phishing employees in response to common IT support questions
    - Multi-party spoofing for advanced social engineering

Conclusion

IM apps have become the default internal communication for most organizations now, but are now a common method of communication with external parties, as well. This means they’ll become a key battleground in both the initial access phase of compromises and the latter phases of lateral movement and persistence.

This also means organizations reliant on traditional email security gateways and email-based phishing training are likely to see the effectiveness of these controls decrease if attacks shift to the IM apps.

In this article, we highlighted a number of spoofing, phishing and persistence techniques that can be employed by an attacker with a foothold that has compromised an internal account on a Slack tenant in order to persist their access and perform lateral movement. In the previous article, we covered spoofing and phishing techniques that could be used by external attackers in the initial access phase to get that first foothold in the first place.

While this article focused on Slack specifically, similar attacks may be possible for other IM apps as well. Going forwards, it will be important for organizations to factor in these types of attacks into their security strategies.
