Push

A Phisher's Guide to Slack

// Evade network monitoring & EDR

// Get access to a user account

// Gain persistence and move laterally

Author: Luke Jennings, VP of R&D, Push Security

Table of Contents

Introduction

In this guide, we'll demonstrate how instant messaging applications are an increasingly attractive target for a range of phishing and social engineering attacks.

We'll use Slack as our primary example and we'll explore external phishing as part of the initial access phase of the kill chain. Then, we'll look at how once an attacker has a foothold on Slack, new attack possibilities open up that allow for persistence and lateral movement to be achieved.

We'll 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?

Instant messenger (IM) apps were primarily created to improve organizations' internal communications, while traditional attacks like phishing and social engineering are often done through external communications, like email. Email remained the standards-based protocol that enabled external communication no matter what email vendor was in use.

In recent years, however, IMs have become the primary method of communication for many businesses. I wanted to focus on IMs 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 for internal communications, their high effectiveness meant there was a strong desire for there to be a way to use them to communicate with external parties as well.

This led IM vendors to create external communication features. Now we have Slack Connect (introduced in June 2020) and Microsoft Teams external access (introduced in January 2022) to support this. This external access has massively increased the attack surface of these platforms.

Despite decades of security research, email security appliances and user security training, email-based phishing and social engineering is often still 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.

In addition to the technical security shortcomings of IM platforms, there also tends to be a sense of urgency associated with IM messages due to the conversational nature of these messages. Combined with a history of increased trust, we have the ingredients for very successful IM-based social engineering attacks.

In fact, there's been an uptick 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.

In this guide, however, we'll focus on a few techniques specific to Slack.

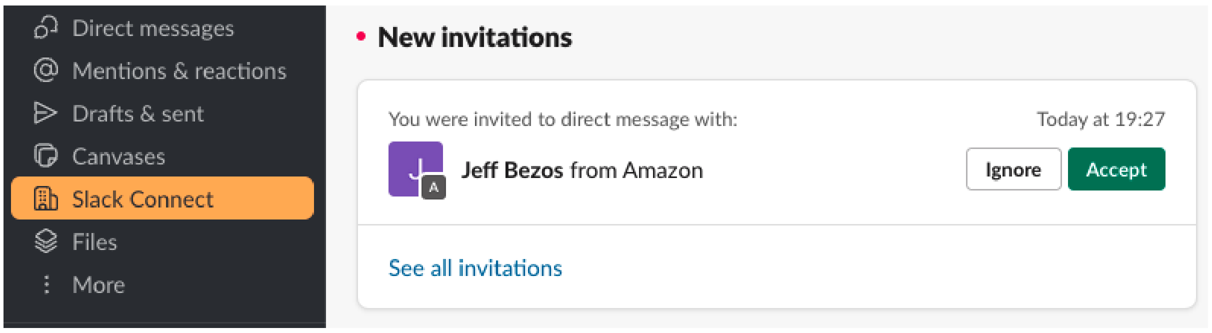
IM user spoofing

We've all seen techniques for spoofing emails, but there are many security controls like Sender Policy Framework (SPF) that can prevent direct spoofing of domains and email security gateways that can flag suspicious domains.

Those security controls don't exist for IM, so we have new options for spoofing. Let's explore a few.

External IM invites

IM applications often make use of informal display names for organization and employee names as well as user-chosen handles (like first names, nicknames, etc). These often don't need to be unique either. Consider the following Slack Connect request:



You were invited to direct message with: Today at 19:27

Jeff Bezos from Amazon

Slack connect invite from an external tenant with an attacker chosen user name and organization name

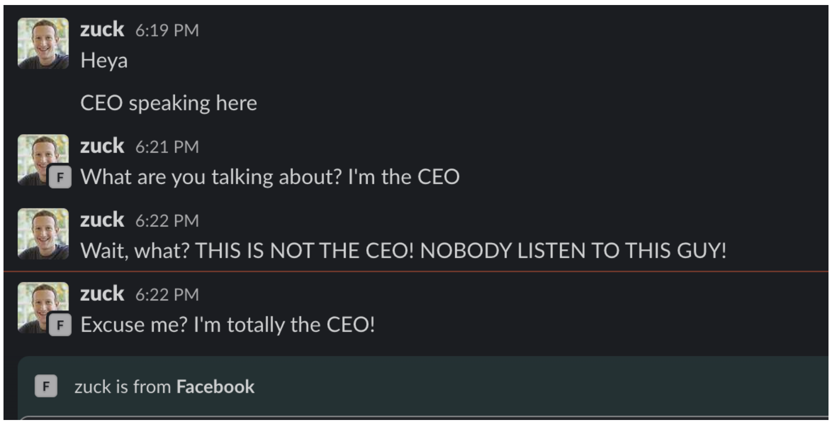
It's not easy for a target user to tell if the user or organization requesting to connect is legitimate when they first receive this invitation. There's also a curiosity incentive — you can't see a first message from the user, so it's tempting for the target user to accept in order to see the message even if they ignore it after viewing.

However, once an attacker gets a first connection the communication channel is open. They could launch an attack right away, or they launch an attack much later after the target user has forgotten they ever connected (more on this later).

Spoofing an internal user

Worse, there's nothing stopping an external attacker from impersonating internal users/employees too. This is especially a concern if an attacker can social engineer their way into being invited into a channel.

An external attacker below (the Zuck with an F on the profile to show it's an external account) in a channel impersonating an internal user (the Zuck without an F to show it's an internal account):



zuck 6:19 PM
Heya

CEO speaking here

zuck 6:21 PM
What are you talking about? I'm the CEO

zuck 6:22 PM
Wait, what? THIS IS NOT THE CEO! NOBODY LISTEN TO THIS GUY!

zuck 6:22 PM
Excuse me? I'm totally the CEO!

F zuck is from **Facebook**

An external attacker in a channel impersonating an internal user.

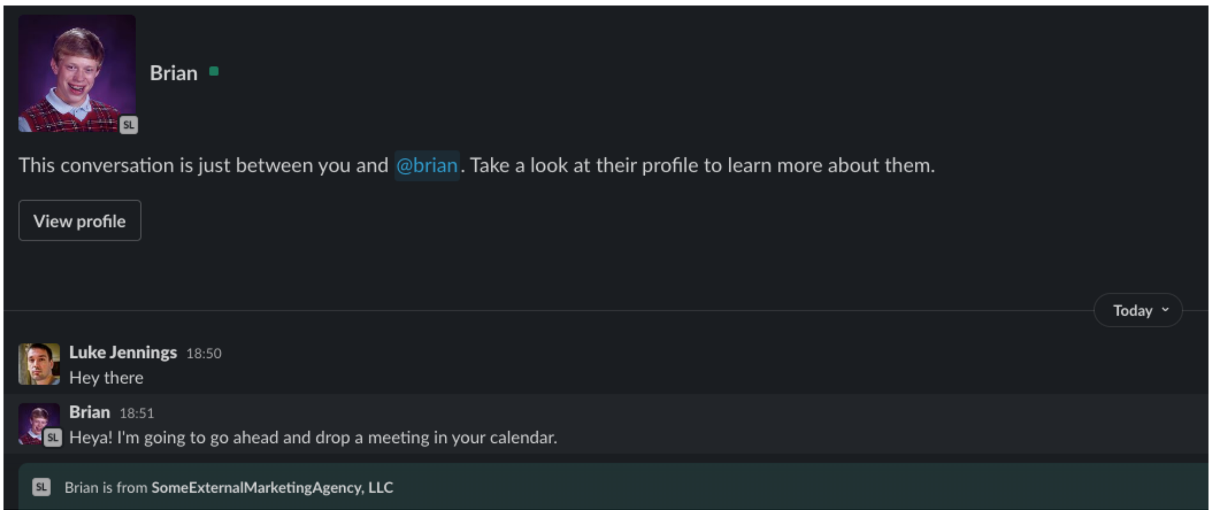
While this particular example is less likely to be successful in a small channel, it's much more of a concern if they change their user identity to replicate an internal employee or teammate, then direct message a member of the channel. DMing an individual channel member doesn't require a new Slack connect invite, so it's much easier for an unsuspecting target to fall victim to social engineering in this way.

Chameleon attack

A particularly interesting external attack capability is that an attacker can act as a chameleon and change their identity over time. Let's say an external attacker successfully connects with a potential target as an external entity. Maybe they exchange some innocuous communications and then leave the conversation to die. Perhaps the target even has Slack message retention settings enabled that delete the chat history after 90 days.

The attacker bides their time and then, in the future, they completely change their Slack identity to impersonate an internal user and message the target again. The connection is already present so the message will come through like any other message, only this time it will appear as if it's coming from a completely different person. It's quite possible that the target could be fooled into believing the message is from the internal user.

This could be particularly dangerous in CEO fraud attacks. An attacker could, for example, forge connections with finance employees ahead of time for seemingly legitimate purposes and later use those connections to send Slack messages spoofing the CEO.


Brian

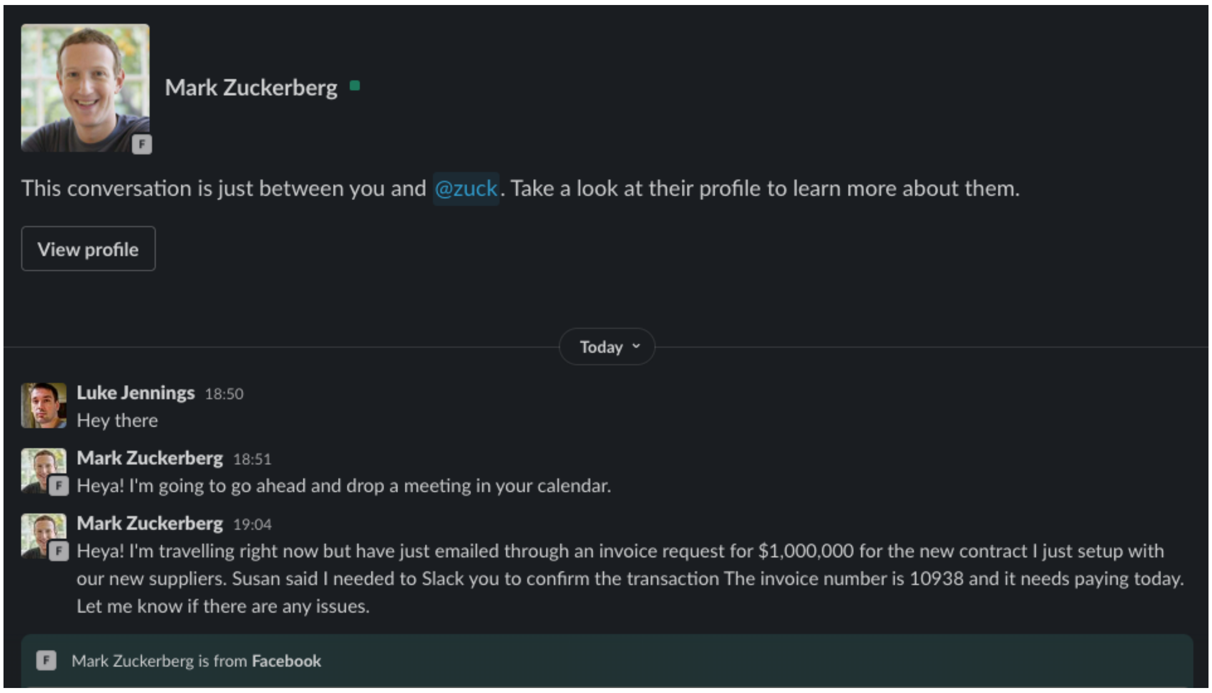
This conversation is just between you and @brian. Take a look at their profile to learn more about them.

Luke Jennings 18:50
Hey there

Brian 18:51
Heya! I'm going to go ahead and drop a meeting in your calendar.

Brian is from SomeExternalMarketingAgency, LLC

An initial message from an accepted Slack connect invite, from "Brian" at "SomeExternalMarketingAgency, LLC"


Mark Zuckerberg

This conversation is just between you and @zuck. Take a look at their profile to learn more about them.

Luke Jennings 18:50
Hey there

Mark Zuckerberg 18:51
Heya! I'm going to go ahead and drop a meeting in your calendar.

Mark Zuckerberg 19:04
Heya! I'm travelling right now but have just emailed through an invoice request for $1,000,000 for the new contract I just setup with our new suppliers. Susan said I needed to Slack you to confirm the transaction The invoice number is 10938 and it needs paying today. Let me know if there are any issues.

Mark Zuckerberg is from Facebook

A social engineering message sent in future with a change in user identity - no new Slack connection is required

All the examples given so far are possible as an external attacker making Slack connect invites, so they work as the initial access phase of the kill chain.

However, if an attacker gains control of an internal Slack user account for the target tenant, or the attacker is a malicious insider (e.g. a disgruntled employee), then they don't even need to worry about achieving an initial connection request. Under a default configuration, they could change their name and photo to impersonate the CEO immediately and message anyone they like. That's how an attacker may move into the lateral movement phase of the kill chain. More on this later.

Let's jump back for just a minute to talk about other things an attacker can do during this initial access phase.

Link preview spoofing

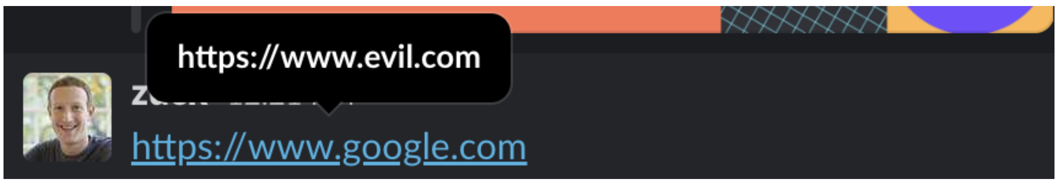
Another key issue is link preview spoofing. HTML allows a variety of ways to specify hyperlinks. In email, secure email gateways will often alert or block commonly abused types, such as forging a different URL as the link display text to what the underlying link points to. For example, an attacker could show the link as https://www.google.com but direct it to https://www.evil.com when it is clicked. Secure email gateways often perform a lot of other analysis of links, including domain analysis and active crawling to identify common phishing attacks.

On IM applications, however, this same standard of link analysis isn't always present. Additionally, the widespread introduction of link unfurling/previewing has also given more options for spoofing links to hide their true source to increase the chances of social engineering success.

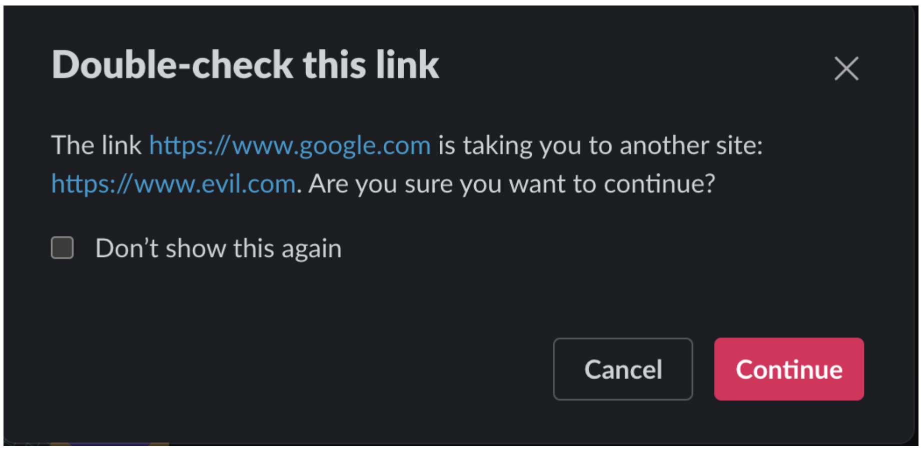
Traditional link forging

We'll start with a common traditional link forging scenario to see how Slack handles that, then show how link previews change the threat.

Here, we can see forging a link is permitted by Slack, but at least the real domain is shown to the user along with an overt warning.



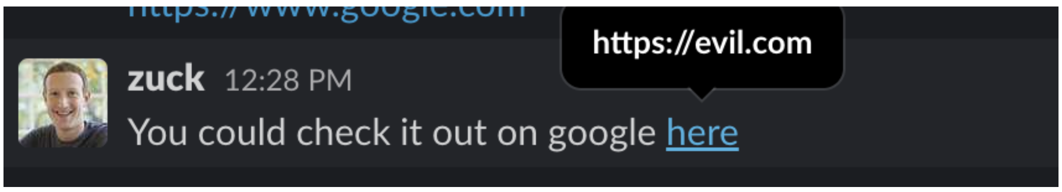
Link forging shows the real domain on a hover-over


Double-check this link

The link https://www.google.com is taking you to another site: https://www.evil.com. Are you sure you want to continue?

Link forging also presents a warning dialog to the user by default if they click the link

On the other hand, if we use conversational text to mask the true URL, we no longer get a warning when clicking the link. We can just embed the link into a conversation instead of posting the link on its own. However, it's still possible to see the real URL via a mouseover, so this doesn't really differ from traditional email phishing scenarios. Without any context of the link, it's likely a security conscious user will hover-over to see what the link points to.



https://evil.com

zuck 12:28 PM

You could check it out on google here

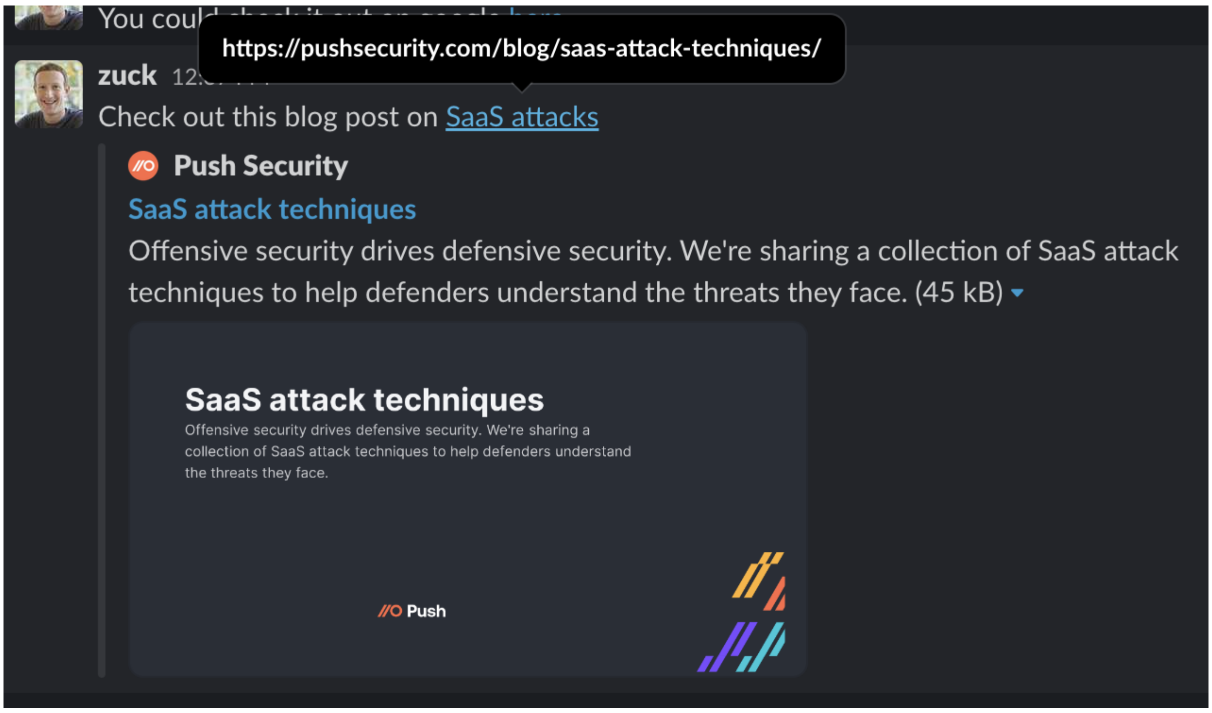
A hover-over still shows the true URL with a friendly text link but no warning dialog is given

Abusing link previews using an internal account

It gets more interesting when we use links that Slack is able to unfurl to provide a link preview. We'll show how this works with full link previews first.

By default, full previews only show for messages from internal users. To make the explanation easier, we'll show full previews first, but then we'll show the difference with limited previews in external messages afterwards and thus show how it impacts external phishing attacks in the initial access phase.

Here, we'll show a legitimate example of posting one of our own blogs where Slack helpfully unfurls the URL and gives some context to the link as a preview:



https://pushsecurity.com/blog/saas-attack-techniques/

zuck

Check out this blog post on SaaS attacks

Push Security

SaaS attack techniques

Offensive security drives defensive security. We're sharing a collection of SaaS attack techniques to help defenders understand the threats they face. (45 kB)

SaaS attack techniques

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

Push

Link unfurling resulting in a helpful link preview

This is very useful for the user and, despite the fact you can still see the real link clearly via a hover-over, a user is much less likely to check a link when they've already had a seemingly legitimate preview context displayed to them.

So, how can we use this scenario maliciously?

The obvious attack scenario is to minimize the link display text so it's not noticeable and hard to hover-over, then forge a different link preview for Slack than what is given to the user when they click the link. Then when the user clicks the link, they'll be directed to our phishing page instead.

We can do this through using a single character as the link display text and then performing user agent specific processing of web requests. For example, Slack unfurling uses a user agent like the following:

```
Unset
Slackbot-LinkExpanding 1.0 (+https://api.slack.com/robots)
```

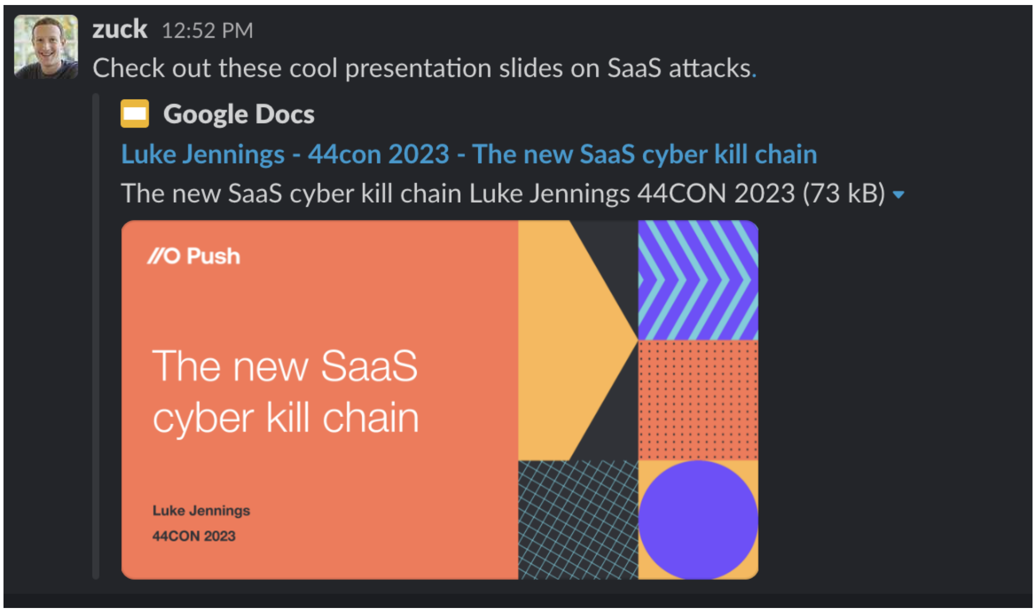
Therefore, without even requiring much sophistication, we can use some simple python code to perform a redirect to a legitimate source when our web request handler sees this user agent. However, when a target user visits using a normal web browser we instead return a malicious page:

```python
Python
from http.server import HTTPServer, SimpleHTTPRequestHandler


class MyHandler(SimpleHTTPRequestHandler):
  def do_GET(self):
    for header, val in self.headers.items():
      if header == "User-Agent":
        print(header, val)
        if val.startswith("Slackbot-LinkExpanding") or "SkypeUriPreview" in val or
"Google-PageRenderer" in val:
          self.send_response(301)
          self.send_header('Location',
'https://docs.google.com/presentation/d/1JsjD2Ro9KaHmW2vILPKJ6-7ptW89pfsAReyzCxQdpq0/edit?
usp=sharing')
          self.end_headers()
          return
      print(header, val)
    return super(MyHandler, self).do_GET()


httpd = HTTPServer(('localhost', 8000), MyHandler)
httpd.serve_forever()
```

Example Python code for redirecting to benign content for a Slack preview, while serving malicious content otherwise

The end result is that the user sees a nice friendly link preview legitimately produced by Slack and Google Docs in real time. However, if they click the link, they'll be taken to our phishing page instead.

We've shown a Google style phishing page as an example for harvesting credentials. Hopefully, the user will assume their Google Docs session expired and then re-enter their credentials. See what the target user would see below:


zuck 12:52 PM

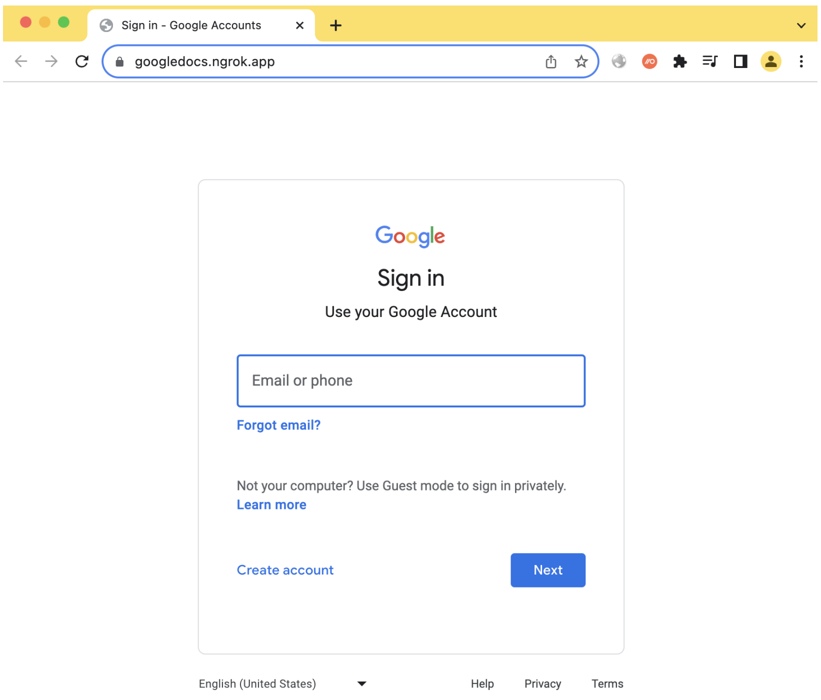
Check out these cool presentation slides on SaaS attacks.

Google Docs

Luke Jennings - 44con 2023 - The new SaaS cyber kill chain

The new SaaS cyber kill chain Luke Jennings 44CON 2023 (73 kB)

Phishing message leveraging user spoofing and link preview spoofing to make the link seem legitimate, so the user won't notice the true URL. A small period is used to hide the URL.



The fake Google phishing page the user is directed to when clicking the link, hosted on a custom ngrok domain

Using a small period as the display text for the hyperlink means it is difficult for the user to notice and hover-over to see Slack pop-up the true domain as we saw earlier. While they can still hover over the link preview itself, this only shows the real domain in the taskbar in the bottom left, which is only noticeable if you intentionally look for it.

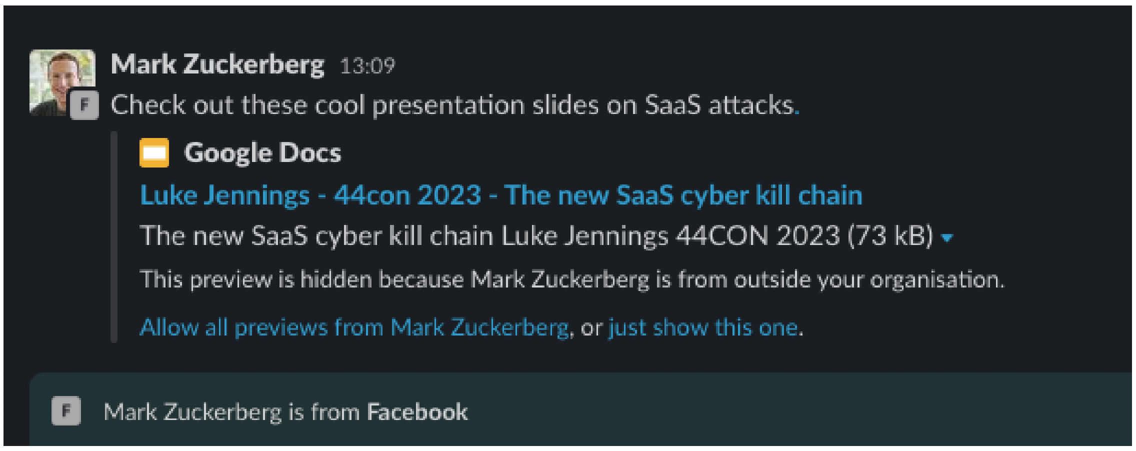
Given normal links in Slack show the domain above the mouse, users aren't used to looking for the link here and, combined with the friendly link preview, it's much less likely a target user will realize this is a phishing attack.

Abusing link previews with an external account

What we've just shown is the behavior for a message from an internal user. Slack doesn't fully unfurl a link by default, however, if this was combined with external messaging as we saw earlier. it does still show a partial link preview, so this attack is still possible.

The only real difference is it doesn't show the image part of the preview. Instead, it shows a notice to the user that it's external and gives them the option to click to show the image preview as well.

If the user clicks to show the image preview, it converts to the same full preview with the image we saw above. We can see an example of chaining the original external user spoofing attack with a link preview spoofing attack below:

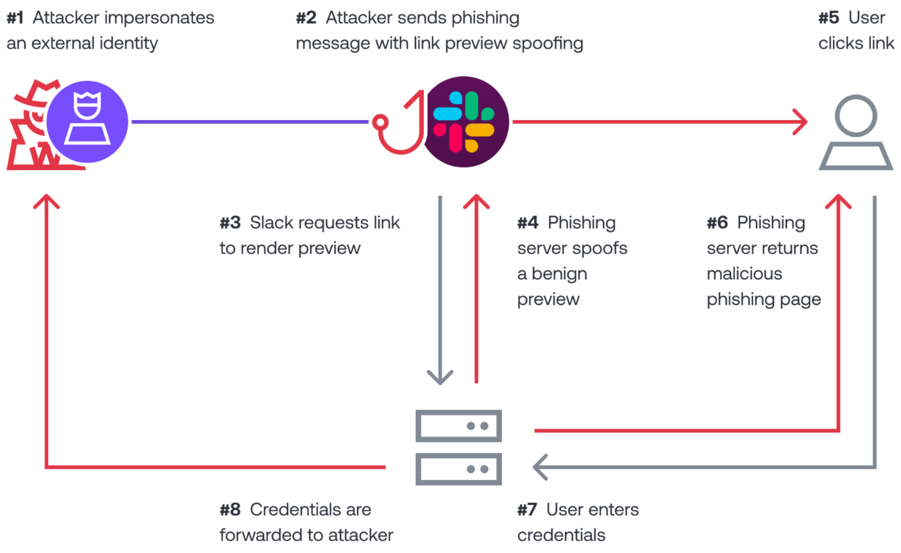


Mark Zuckerberg 13:09
Check out these cool presentation slides on SaaS attacks.
Google Docs
Luke Jennings - 44con 2023 - The new SaaS cyber kill chain
The new SaaS cyber kill chain Luke Jennings 44CON 2023 (73 kB)
This preview is hidden because Mark Zuckerberg is from outside your organisation.
Allow all previews from Mark Zuckerberg, or just show this one.

Mark Zuckerberg is from Facebook

Link previews from external messages do not show the image by default, but allow the user to override this

While this is slightly more problematic for an attacker than the internal functionality for link previews, it's still very useful as a social engineering technique and arguably the option to click "just show this one" adds to the legitimacy. The user may use this as a way to get context on what the link is, instead of looking for the underlying URL. Otherwise, clicking the link still takes the user to the phishing page without any other warnings the same as for internal messages.



#1 Attacker impersonates an external identity
#2 Attacker sends phishing message with link preview spoofing
#5 User clicks link
#3 Slack requests link to render preview
#4 Phishing server spoofs a benign preview
#6 Phishing server returns malicious phishing page
#8 Credentials are forwarded to attacker
#7 User enters credentials

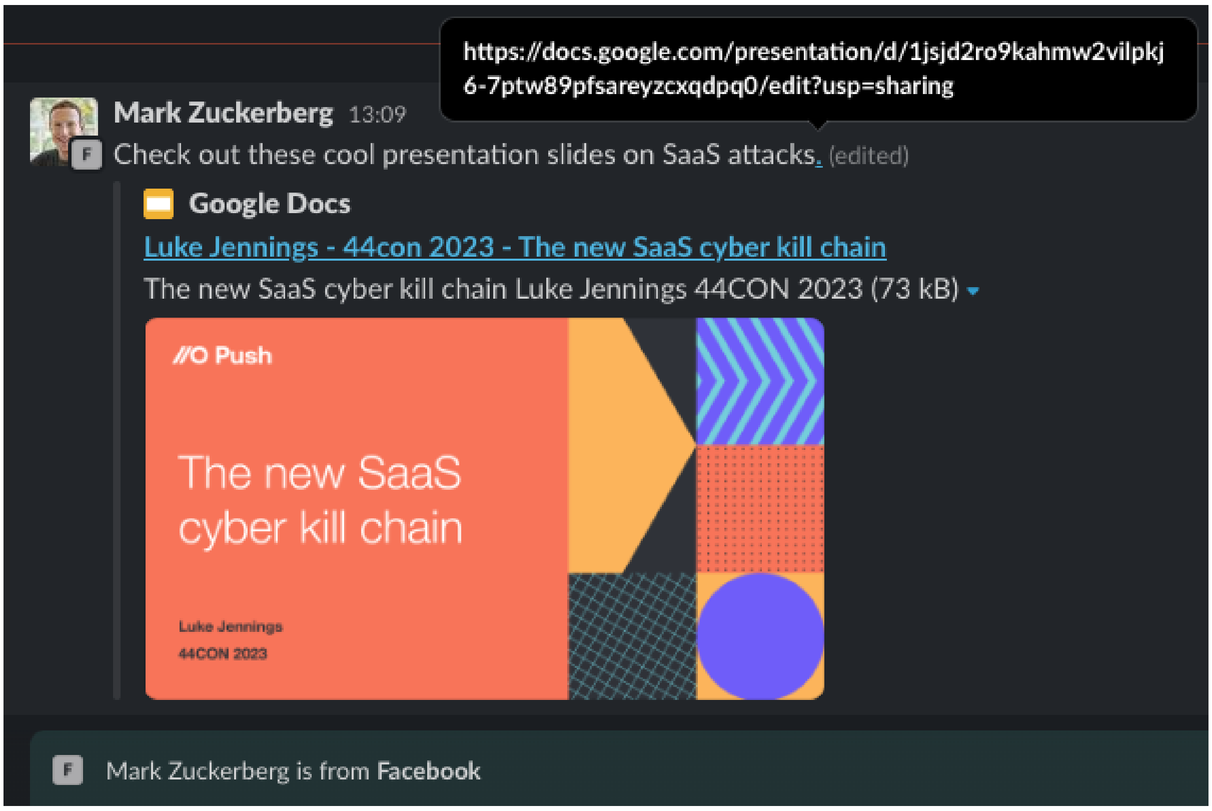
A diagram to show the combination of external spoofing and link preview spoofing in action

Cleaning your tracks

Let's say an attacker has either successfully phished the target user or perhaps now the user is suspicious and likely contacting Security or IT. Since Slack lets you edit and delete messages, an attacker can abuse this functionality to hide their attack.

An attacker can make a tiny change to the original message to replace the malicious link with the legitimate link they were spoofing for the link preview if they got the sense the target was getting suspicious. Then, if an incident responder comes to investigate, the malicious link is now gone and the message itself appears identical, covering their tracks. Other than being able to see the message has been edited, it's no longer easy to see this was a phishing attack or where the phishing link pointed to. It's definitely a useful capability that isn't usually possible with email phishing!

See this minor change reflected below, making the original phishing message appear innocuous due to the replacement of the phishing URL with a legitimate URL:



https://docs.google.com/presentation/d/1jsjd2ro9kahmw2vilpkj6-7ptw89pfsareyzcxqdpq0/edit?usp=sharing

Mark Zuckerberg 13:09
Check out these cool presentation slides on SaaS attacks. (edited)

Google Docs
Luke Jennings - 44con 2023 - The new SaaS cyber kill chain
The new SaaS cyber kill chain Luke Jennings 44CON 2023 (73 kB)

Mark Zuckerberg is from Facebook

An edited message to remove the malicious link and replace it with the same link used for spoofed link preview.

Next, we'll look at how once an attacker has a foothold on Slack, new attack possibilities open up that allow for persistence and lateral movement to be achieved.

Gaining persistence and moving laterally

In this section, 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.

Using app integrations to gain persistence

We've 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 an attacker compromises a Slack account and wants to persist and/or move laterally? Ordinarily, they can maintain access until session expiry or a password change is forced or the account is deactivated/deleted. An attacker could silently read messages as the user continues to operate their account. But if they start sending out malicious links to other targets in an attempt to move laterally, 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 dig into next.

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

Previously, we wrote 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. We're going to use make.com as our automation app example.

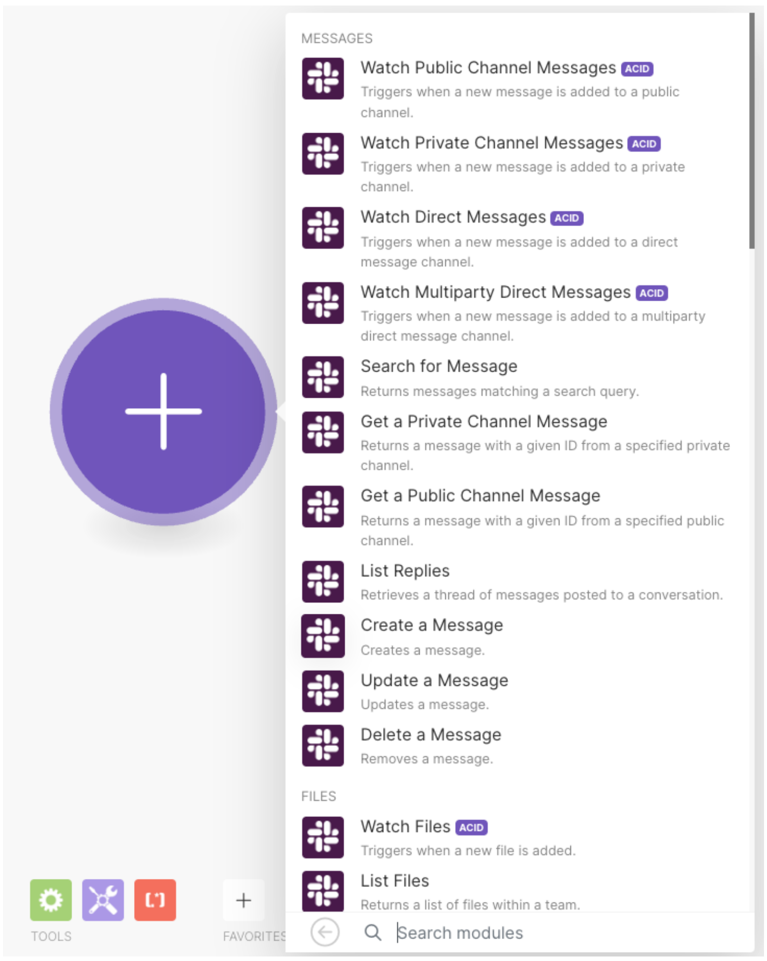
Persistent spoofing using bot tokens

There are two options available for us to play with to connect make.com to a Slack account you control to maintain persistence. Here are the pros and cons of each approach:

1. **User token** — gives full persistence as the user and can do anything a user can do, but you lose access if the user account is deactivated/deleted.
2. **Bot token** — less privileges, can't automatically read all public channels, but it can easily spoof messages that are separated from the user. Another benefit is that even if the user account gets deleted/deactivated, you can still send spoofed messages to other users (though you might not be able to see replies).

For this first example, we're going to use a bot token.

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.



MESSAGES

Watch Public Channel Messages ACID
Triggers when a new message is added to a public channel.

Watch Private Channel Messages ACID
Triggers when a new message is added to a private channel.

Watch Direct Messages ACID
Triggers when a new message is added to a direct message channel.

Watch Multiparty Direct Messages ACID
Triggers when a new message is added to a multiparty direct message channel.

Search for Message
Returns messages matching a search query.

Get a Private Channel Message
Returns a message with a given ID from a specified private channel.

Get a Public Channel Message
Returns a message with a given ID from a specified public channel.

List Replies
Retrieves a thread of messages posted to a conversation.

Create a Message
Creates a message.

Update a Message
Updates a message.

Delete a Message
Removes a message.

FILES

Watch Files ACID
Triggers when a new file is added.

List Files
Returns a list of files within a team.

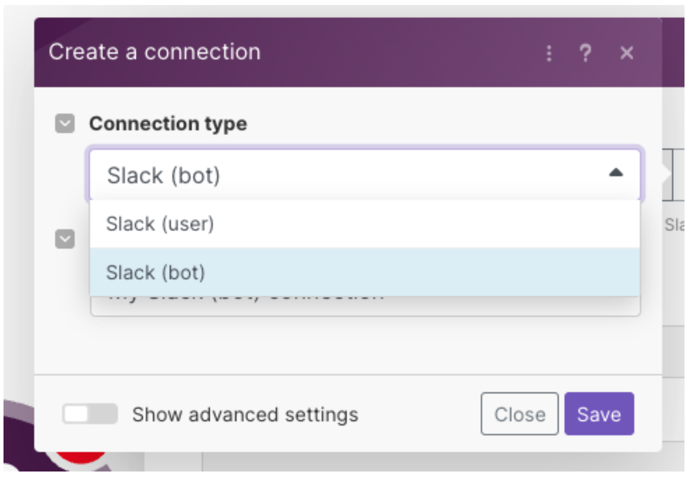
Creating a new scenario in make.com and picking a Slack module

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.

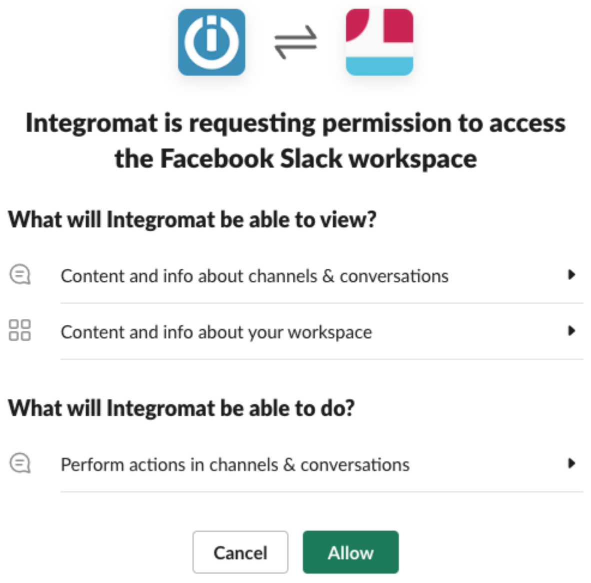
For this module, we have the option of creating either a user token or a bot token. A user token has full capabilities and will continue to operate in the event of a password change. However, if the user account is deactivated or deleted then it will cease to work.

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



Integromat is requesting permission to access the Facebook Slack workspace

What will Integromat be able to view?

- Content and info about channels & conversations
- Content and info about your workspace

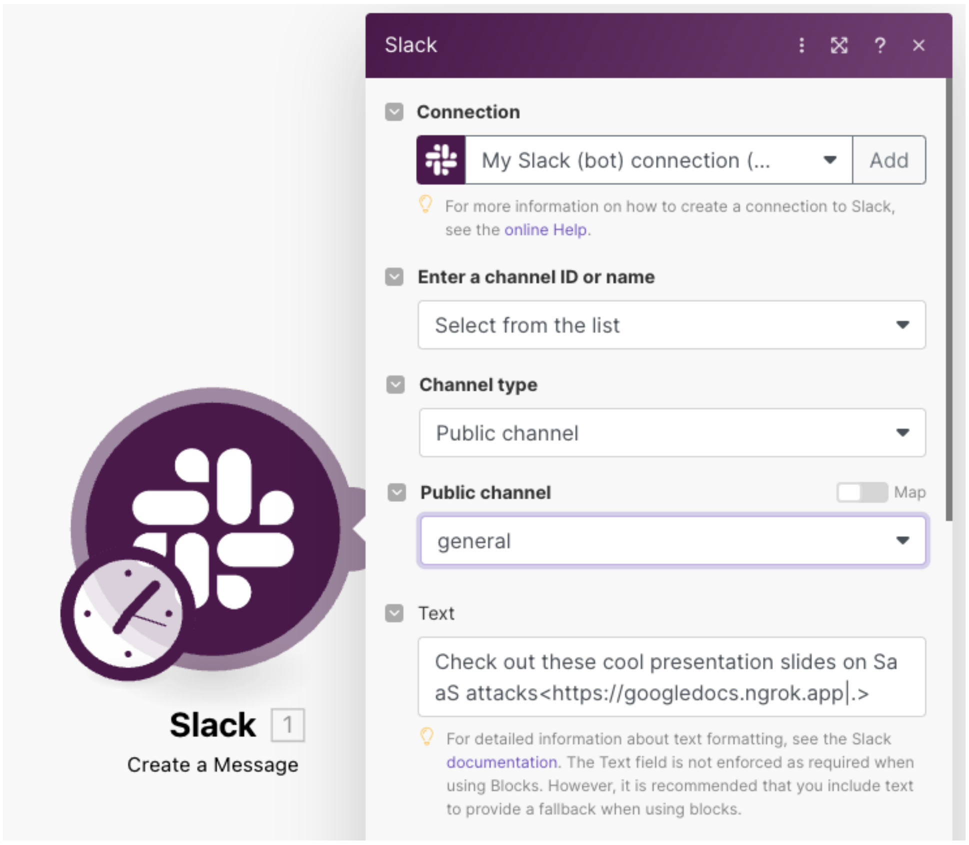
What will Integromat be able to do?

- Perform actions in channels & conversations

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. We'll demonstrate using it to send the same type of spoofed message we covered earlier, 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'll do 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.



Connection: My Slack (bot) connection (... Add

For more information on how to create a connection to Slack, see the online Help.

Enter a channel ID or name: Select from the list

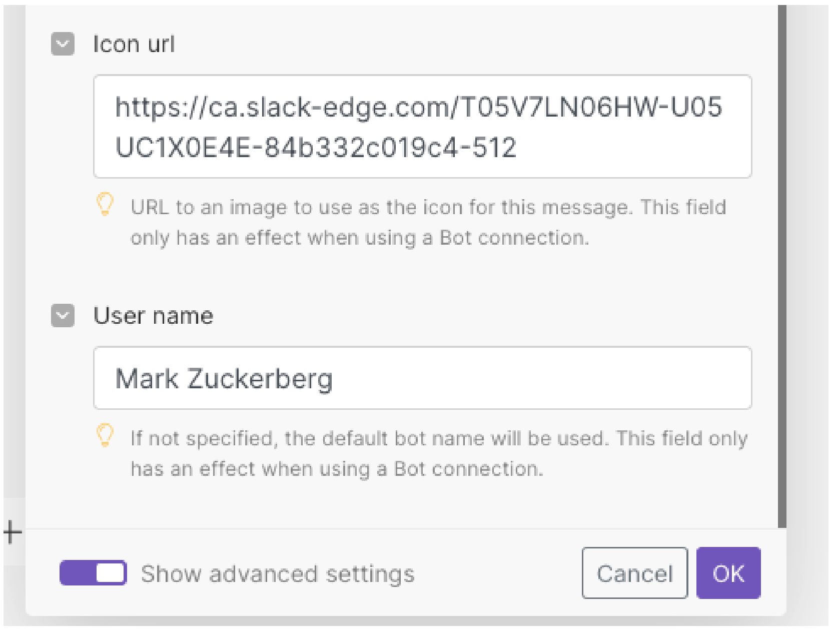
Channel type: Public channel

Public channel: general

Text: Check out these cool presentation slides on SaaS attacks<https://googledocs.ngrok.app|.>

For detailed information about text formatting, see the Slack documentation. The Text field is not enforced as required when using Blocks. However, it is recommended that you include text to provide a fallback when using blocks.

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



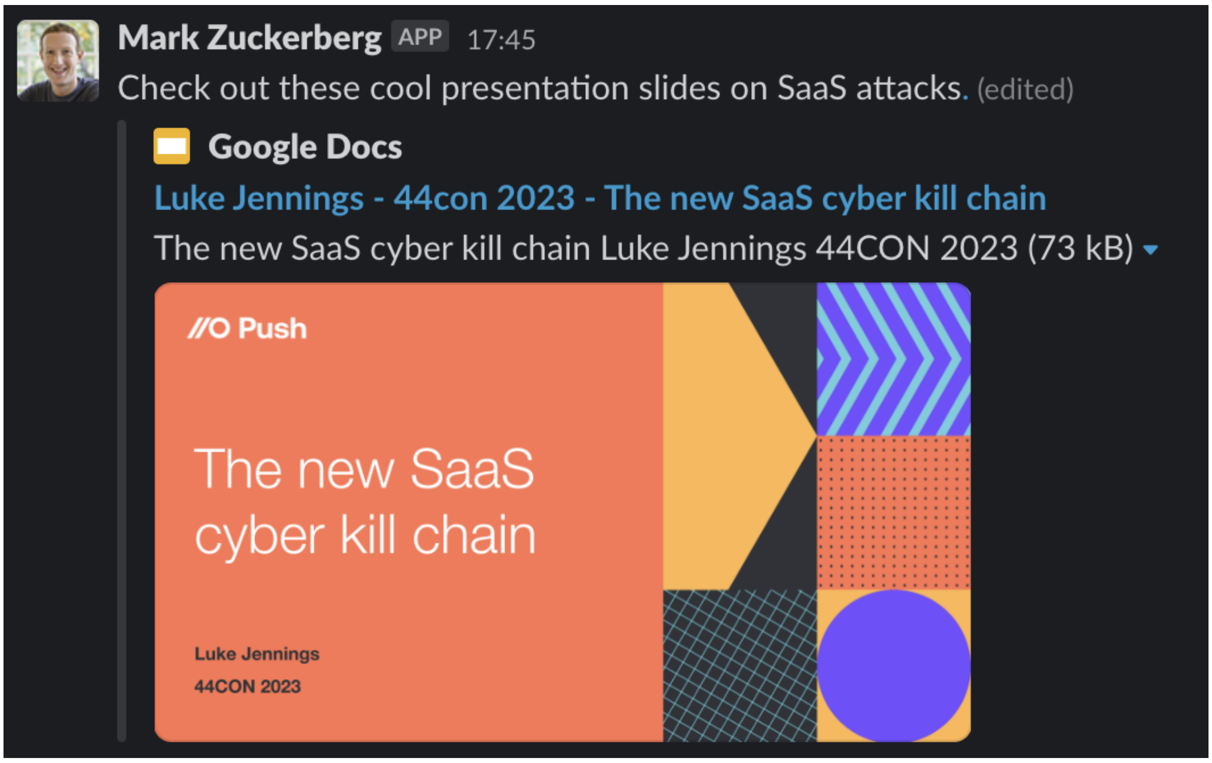
Icon url: https://ca.slack-edge.com/T05V7LN06HW-U05UC1X0E4E-84b332c019c4-512

URL to an image to use as the icon for this message. This field only has an effect when using a Bot connection.

User name: Mark Zuckerberg

If not specified, the default bot name will be used. This field only has an effect when using a Bot connection.

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


Mark Zuckerberg APP 17:45

Check out these cool presentation slides on SaaS attacks. (edited)

Google Docs

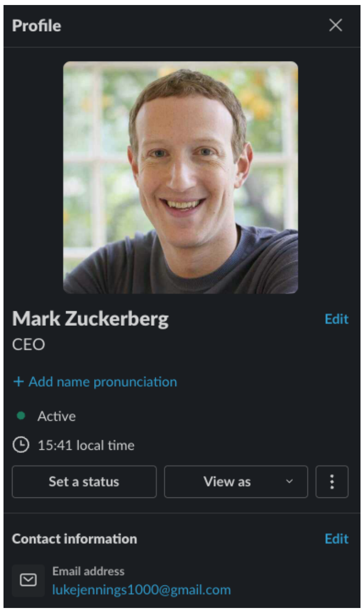
Luke Jennings - 44con 2023 - The new SaaS cyber kill chain

The new SaaS cyber kill chain Luke Jennings 44CON 2023 (73 kB)

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*:

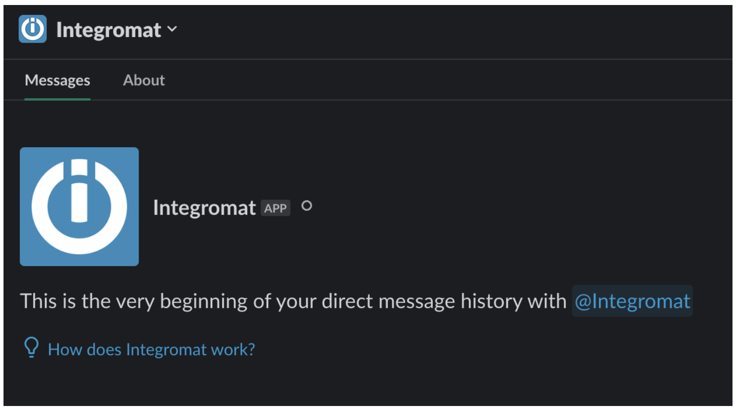

Profile

Mark Zuckerberg
CEO

Active
15:41 local time

Contact information
Email address
lukejennings1000@gmail.com

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


Integromat

Messages About

Integromat APP

This is the very beginning of your direct message history with @Integromat

How does Integromat work?

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 using both user and bot tokens

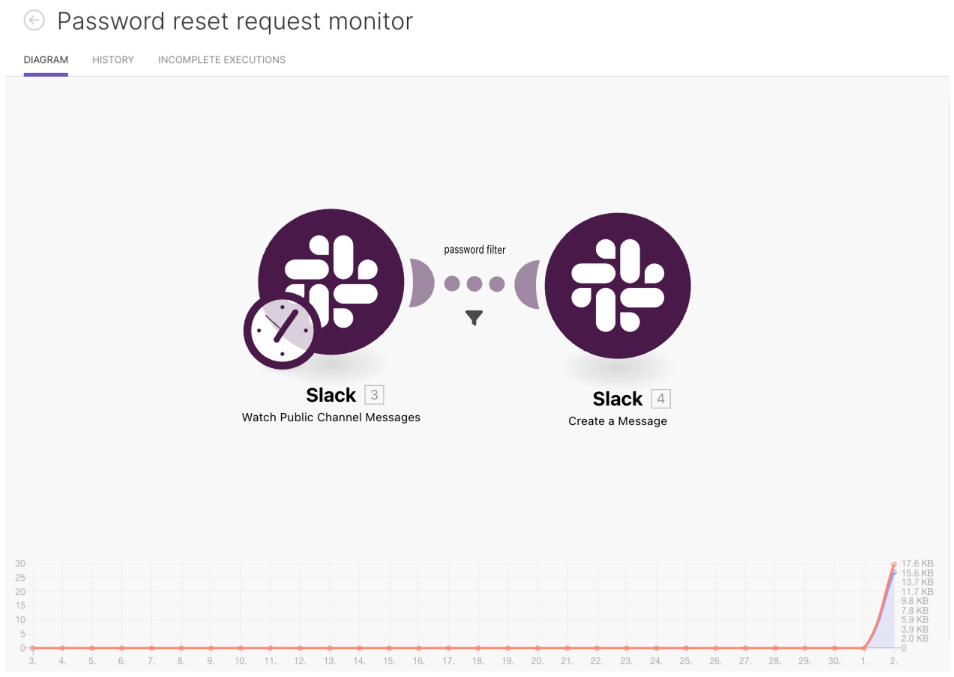
We've just seen how you can internally spoof a message via a Slack app in a way that's harder to track back to the original compromised user account and also achieve persistence at the same time.

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.

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.

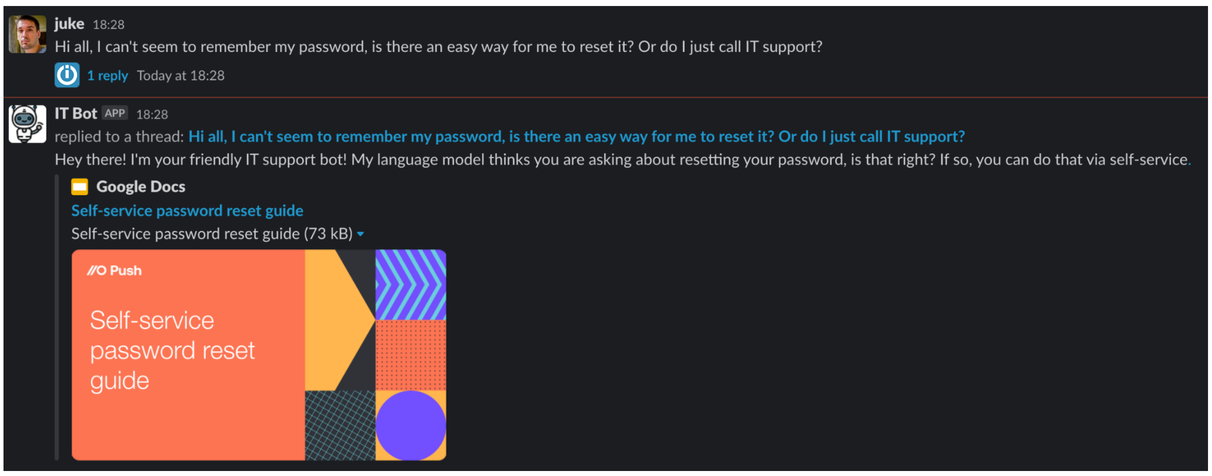
One caveat here is make.com requires we use a user token for the message monitoring part and therefore this attack couldn't survive a deactivated/deleted Slack user account. However, it will still survive password changes and so is still a useful persistence option too. Additionally, the bot token can 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

We've 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's 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 earlier in this guide and the actual link will present a fake Google login page to harvest credentials.

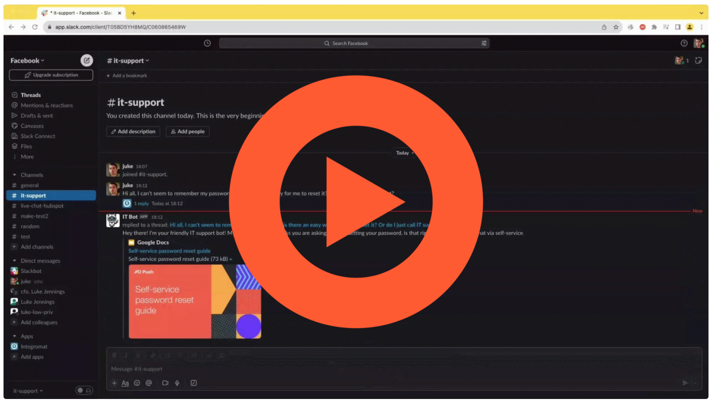


juke 18:28
Hi all, I can't seem to remember my password, is there an easy way for me to reset it? Or do I just call IT support?
1 reply Today at 18:28

IT Bot APP 18:28
replied to a thread: Hi all, I can't seem to remember my password, is there an easy way for me to reset it? Or do I just call IT support?
Hey there! I'm your friendly IT support bot! My language model thinks you are asking about resetting your password, is that right? If so, you can do that via self-service.
Google Docs
Self-service password reset guide
Self-service password reset guide (73 kB)

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:



it-support

You created this channel today. This is the very beginni

juke 18:07
joined #it-support.

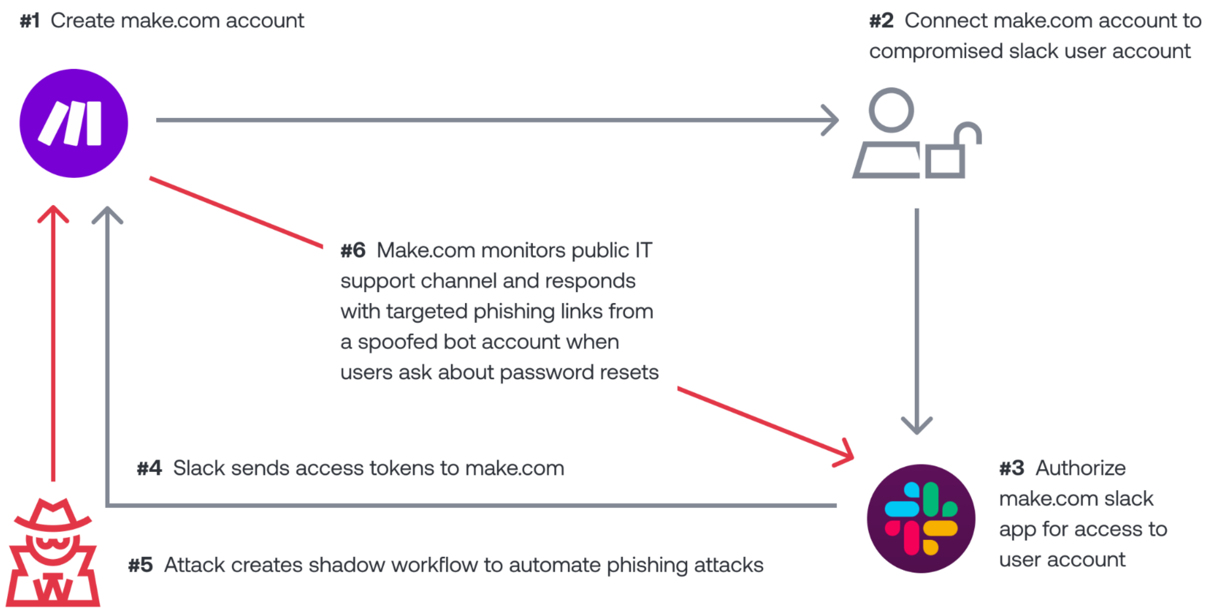
juke 18:12
Hi all, I can't seem to remember my passwor … y for me to reset it? … ?
1 reply Today at 18:12

IT Bot APP 18:12
replied to a thread: Hi all, I can't seem to rem … is there an easy w … et it? Or do I just call IT su
Hey there! I'm your friendly IT support bot! M … ks you are asking … etting your password, is that ri … hat via self-service.

Google Docs
Self-service password reset guide
Self-service password reset guide (73 kB)

Push
Self-service password reset guide

To summarize, here's a diagram to show how this all fits together:



#1 Create make.com account
#2 Connect make.com account to compromised slack user account
#3 Authorize make.com slack app for access to user account
#4 Slack sends access tokens to make.com
#5 Attack creates shadow workflow to automate phishing attacks
#6 Make.com monitors public IT support channel and responds with targeted phishing links from a spoofed bot account when users ask about password resets

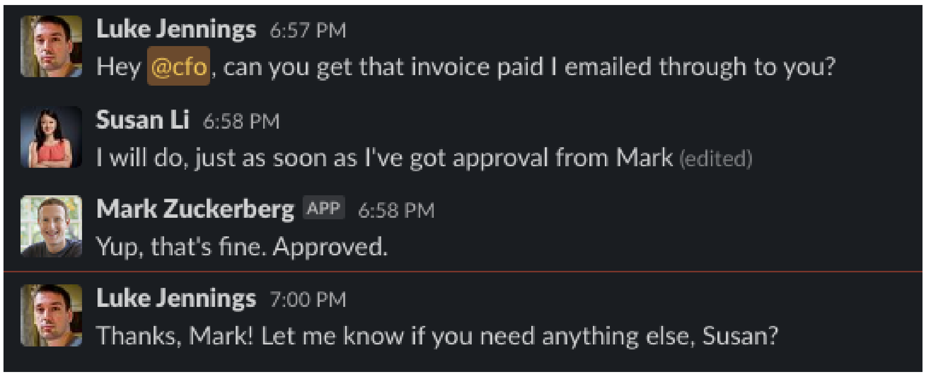
Diagram showing the connections between the attacker, compromised Slack account and make.com

Multi-party spoofing

Another great option provided by Slack apps and bot tokens 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. This might be a common process for access requests requiring manager approval and many other business processes.

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.


Luke Jennings 6:57 PM
Hey @cfo, can you get that invoice paid I emailed through to you?

Susan Li 6:58 PM
I will do, just as soon as I've got approval from Mark (edited)

Mark Zuckerberg APP 6:58 PM
Yup, that's fine. Approved.

Luke Jennings 7:00 PM
Thanks, Mark! Let me know if you need anything else, Susan?

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.

Impact

- 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 guide, we covered spoofing and phishing techniques that could be used by external attackers in the initial access phase to get a foothold. Then we highlighted a number of spoofing, phishing and persistence techniques that can be employed by an attacker once they've gotten that foothold by compromising an internal account on a Slack tenant, so they can persist their access and perform lateral movement.

While this guide focused on Slack specifically, similar attacks may be possible for other IM apps as well. We'll be exploring this possibility in future research, so make sure to follow us on LinkedIn and X to get the latest intel.

It's clear that organizations need to start factoring these types of attacks into their security strategies for the future.