Cover Your SaaS

Token theft, OAuth abuse, and API exposure in SaaS incident response

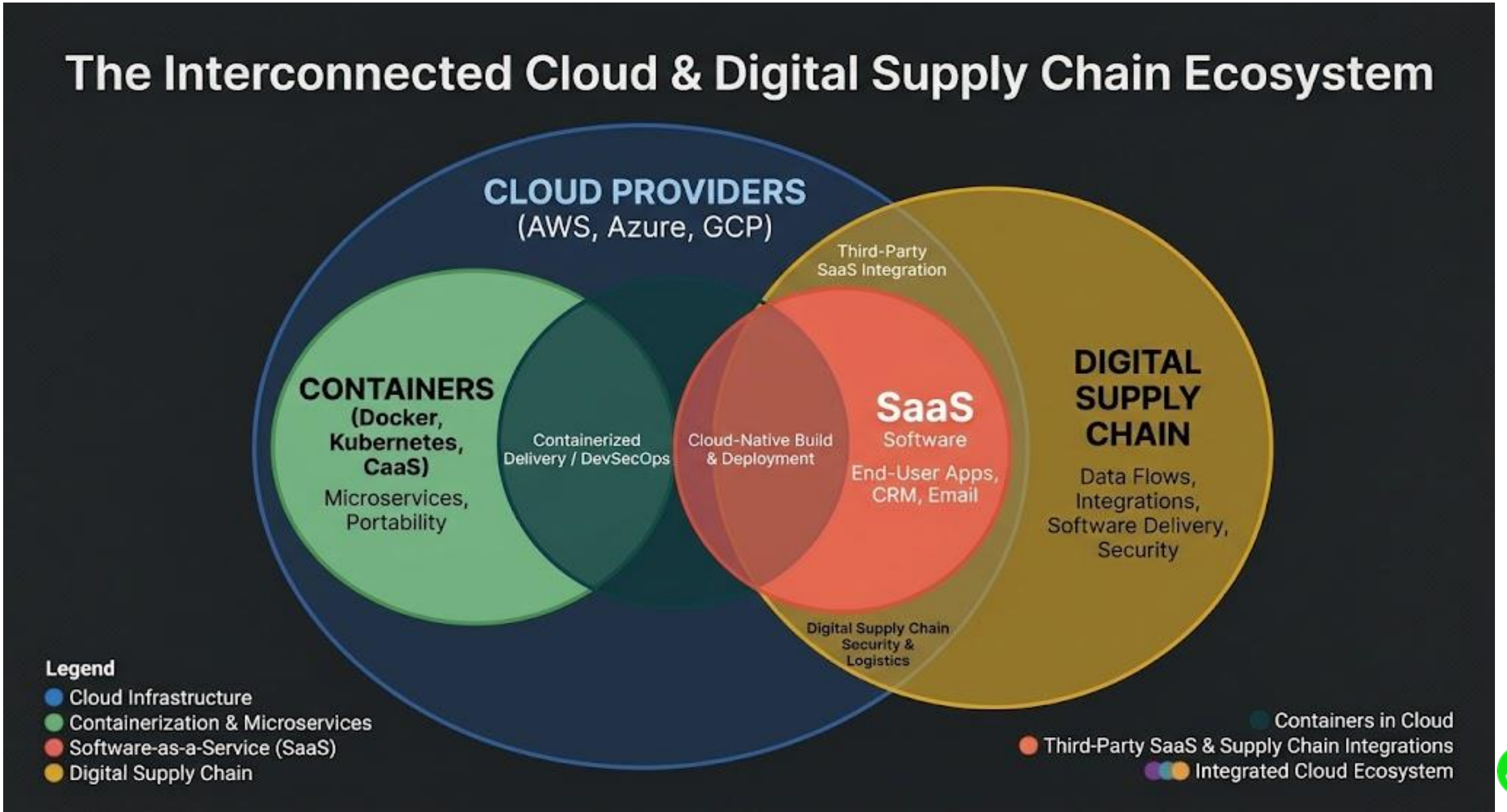
TRUSTEDSEC

Shane Hartman

Principal IR Consultant



Confused about where SaaS Sits?



When you have an Incident
Where do you even start?



No hard drives
to image



No network packets
to sniff



No physical servers
to seize

In SaaS, the identity is the host and the audit log is the disk.

Everything you examine is an API record someone else generated,
retained on someone else's schedule.

TRUSTEDSEC



Agenda – Down the Rabbit Hole

01 Why SaaS IR is different

02 The new kill chain

03 Rabbit hole #1: Token theft

04 Rabbit hole #2: OAuth abuse

05 Rabbit hole #3: API exposure

06 Where to look



TRUSTEDSEC

01

Why SaaS IR is different

You don't own the evidence. You rent it, and the lease is short.



TRUSTEDSEC

Traditional IR vs. SaaS IR Methods

	Traditional IR	SaaS IR
Primary evidence	Disk images, memory, PCAP, EDR telemetry	Audit and sign-in logs pulled via vendor APIs
Pivot point	Hostname, IP, process tree	Identity, app / client ID, session and token IDs
Containment	Isolate the host, block the IP	Revoke sessions, tokens, grants, integrations
Who holds evidence	You	The vendor, on their retention schedule
Attacker tooling	Malware, C2 implants	Legitimate APIs and your own integrations
What 'normal' looks like	Baselined endpoints	Rarely baselined: integration users and apps



Three Parties hold your evidence



1. You control

- Which logs are licensed and enabled
- Export to your SIEM and how long you keep it
- Your inventory of apps, grants and integration users



2. The SaaS vendor controls

- What gets logged and at what fidelity
- Default retention and log lag
- Support-only data you must request in a case



3. The integrator controls

- Where your OAuth tokens are stored
- Their own access and infrastructure logs
- When (and whether) they tell you

The Retention Clock Starts before the incident

Log source	Default window	Watch out for
Microsoft Entra sign-in and audit logs	**30 days (P1/P2) · 7 days (Free)**	Upgrading later is not retroactive
Microsoft Purview Audit (Standard)	**180 days**	Not every Entra event lands in the UAL
Google Workspace audit logs	**6 months**	Email log search: 30 days
Salesforce Event Monitoring	**30 days (licensed)**	Free tier: a few log types, 1 day
Salesforce Setup Audit Trail	**180 days**	Config changes only
Okta System Log	**90 days**	Stream it or lose it

Defaults as documented by each vendor; confirm against your licensing.

02

The new kill chain

No exploits. No malware on your endpoints. Just valid credentials, used somewhere you didn't expect.



The Traditional Kill Chain is Dead in SaaS



The New Chain 3 doors into your SaaS



Token theft

Steal the session, skip the login: cookies, access and primary refresh tokens (PRT)s

MITRE ATT&CK
T1528 · T1539 · T1550.001



OAuth abuse

Talk a user into consenting, or steal the tokens a trusted vendor already holds

MITRE ATT&CK
T1199 · T1528 · T1566



API exposure

Keys in tickets, over-scoped integration users, non-human identities, AI agents

MITRE ATT&CK
T1552 · T1078.004

Eighteen months of SaaS incidents

Jun 2025

UNC6040 vishing installs fake Data Loader apps in Salesforce

Aug 2025

Salesloft Drift OAuth tokens abused across hundreds of Salesforce orgs

Nov 2025

Gainsight connector tokens hit 200+ Salesforce instances

Feb 2026

EvilTokens device code phishing kit; 340+ M365 orgs by March

Apr 2026

Anodot → Snowflake; Context.ai → Vercel via Workspace OAuth

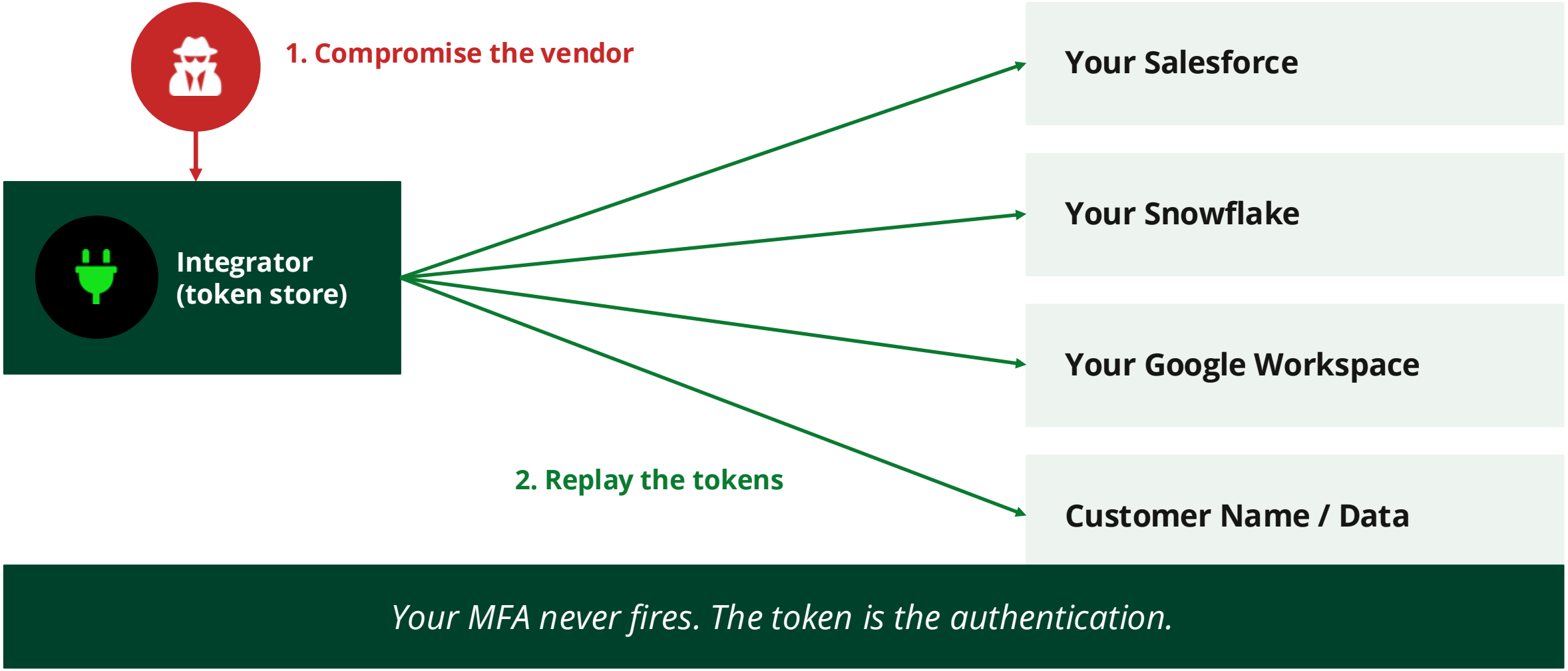
Jun 2026

Klue integration breach exposes Salesforce and Gong tokens

Aug 2026

GhostCode device code kit registers devices and grabs PRTs

One breach, hundreds of victims



03

Rabbit hole #1: Token theft

Why log in when you can just be logged in?



Know your tokens

Artifact	What it buys the attacker	Typical life	How it's stolen
Session cookie	A live browser session	Hours to days	AiTM proxy, infostealer
Access token	Direct API calls as the user	~60–90 min (Entra)	Memory, logs, proxies
Refresh token	Mints new access tokens	Days to months	Device code phishing, infostealer
Primary Refresh Token	SSO to everything as a device	14 days, rolling	Rogue device registration
Integration OAuth token	Tenant-wide API access as the app	**Often no expiry**	Vendor token store breach

Lifetimes are platform defaults and vary with configuration and policy.





How tokens walk out the door

Adversary-in-the-middle

A reverse proxy relays the real login and keeps the session cookie after MFA.

Infostealers

Lumma on a Context.ai laptop (Feb 2026) started the chain that reached Vercel.

Device code phishing

The victim types the attacker's code into the real microsoft.com/devicelogin.

Vendor token stores

Drift, Gainsight, Anodot and Klue each held tokens for many customers.

Device code phishing: MFA works anyway

1

Request

Attacker asks Entra ID for a device code via the legitimate API

2

Lure

DocuSign, voicemail or 'shared file' page shows the code; kits generate it live

3

Victim signs in

Real Microsoft page, real password, real MFA prompt

4

Harvest

Attacker's poller receives access and refresh tokens

5

Persist

Register a device, obtain a PRT, even enroll in Intune

EvilTokens (Feb 2026): 340+ orgs in a month · GhostCode (Aug 2026): PRT theft plus Intune enrollment



What token theft leaves in Entra ID

`AuthenticationProtocol = deviceCode`	Device code sign-ins by users who never use CLIs or TVs
`Non-interactive sign-in log`	Where replayed refresh tokens show up; often not forwarded to the SIEM
`SessionId / UniqueTokenIdentifier`	Linkable IDs that follow one session into Exchange, SharePoint, Teams and Graph activity logs
`Device registration / Intune`	New devices registered to the victim right after a suspicious sign-in
`IP, ASN and user agent drift`	One SessionId seen from two networks, such as cloud-hosting ranges



Hunt: one session, many networks

```
union SigninLogs,
AADNonInteractiveUserSignInLogs
| where TimeGenerated > ago(30d)
| where isnotempty(SessionId)
| summarize IPs = make_set(IPAddress),
    ASNs = make_set(AutonomousSystemNumber),
    Apps = make_set(AppDisplayName),
    First = min(TimeGenerated),
    Last = max(TimeGenerated)
    by SessionId, UserPrincipalName
| where array_length(ASNs) > 1
| order by array_length(ASNs) desc
```

Read the results

- Two ASNs for one session is a lead, not a verdict: mobile and VPN users flap
- Pivot the SessionId into the UAL and Graph activity logs
- Swap in AuthenticationProtocol == "deviceCode" to find device code abuse

```
let DeviceCodeSessions =
    SigninLogs
    | where TimeGenerated > ago(30d)
    | where AuthenticationProtocol ==
"deviceCode"
    | where isnotempty(SessionId)
    | distinct SessionId;
```




Hunt Resources to get Started

Azure Sentinel (Microsoft): https://github.com/Azure/Azure-Sentinel
Entra ID solution: https://github.com/Azure/Azure-Sentinel/tree/master/Solutions/Microsoft%20Entra%20ID
Hunting queries: https://github.com/Azure/Azure-Sentinel/tree/master/Hunting%20Queries
SigmaHQ: https://github.com/SigmaHQ/sigma
Cloud rules: https://github.com/SigmaHQ/sigma/tree/master/rules/cloud
Correlation rule spec: https://github.com/SigmaHQ/sigma-specification
Elastic detection rules: https://github.com/elastic/detection-rules
Splunk security content: https://github.com/splunk/security_content

Community KQL collections
Bert-JanP: https://github.com/Bert-JanP/Hunting-Queries-Detection-Rules
reprise99 (Matt Zorich): https://github.com/reprise99/Sentinel-Queries
SlimKQL: https://github.com/SlimKQL/Hunting-Queries-Detection-Rules
MPreimanis Entra detections: https://github.com/MPreimanis/entra-detections
pirateape Azure-Security: https://github.com/pirateape/Azure-Security

For testing your hunts
Stratus Red Team (Datadog): https://github.com/DataDog/stratus-red-team
Microsoft Extractor Suite (Invictus-IR): https://github.com/invictus-ir/Microsoft-Extractor-Suite



A password reset is not eviction

What teams do

- Reset the password
- Re-register MFA
- Close the ticket

What eviction requires

- Revoke all sign-in sessions and refresh tokens (***twice***)
- Delete devices the attacker registered kills the *Primary Refresh Token* (PRT) path
- Remove MFA methods and passkeys the attacker added
- Review OAuth grants, inbox rules and app registrations created during the session
- Confirm Continuous Access Evaluation covers the apps involved

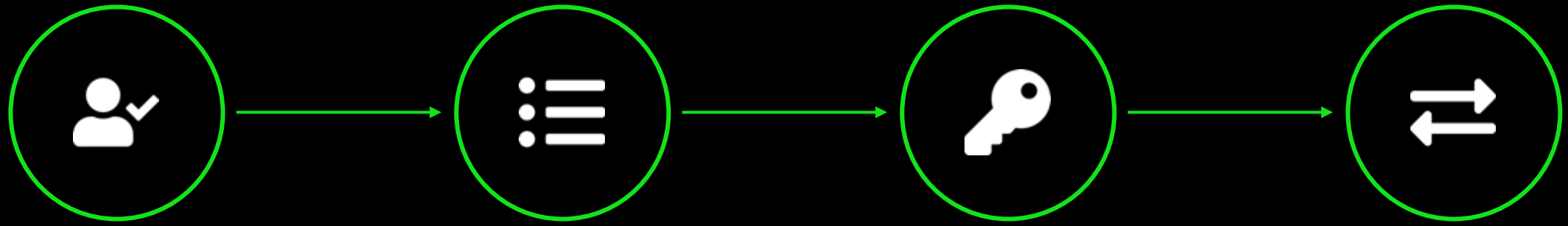
04

Rabbit hole #2: OAuth abuse

Consent is a credential. So is every token your vendors hold for you.



OAuth, From the Investigator's Chair



Consent

A user or admin approves an app

Scopes

The app receives a set of permissions

Tokens

The app stores a refresh token, not you

API calls

The app acts as the user or an integration user

Five questions for every grant: Who consented? When? Which scopes? Where is the token stored? Who else could have it?

Case study: Salesloft Drift

Mar–Jun 2025	Salesloft GitHub accessed: repos downloaded, guest user added, workflows created
Then	Pivot into Drift's AWS environment; customer OAuth tokens stolen
Aug 8–18	Systematic exports from Salesforce orgs; records searched for AWS keys, passwords, Snowflake tokens
Anti-forensics	Query jobs deleted, but the logs were not affected
Aug 20	Salesloft and Salesforce revoke Drift tokens; app pulled from AppExchange

`Drift user agents seen in Salesforce logs: Salesforce-Multi-Org-Fetcher/1.0 · Salesforce-CLI/1.0`



Case study: Context.ai to Vercel

Feb 2026	Lumma infostealer on a Context.ai employee laptop
Mar 2026	Context.ai's AWS accessed; OAuth tokens stolen
Mar–Apr	Token used on a Vercel employee's Google Workspace; the app held broad 'Allow All' scope
Apr 2026	Lateral movement into Vercel internal systems
Apr 19	Disclosure; malicious OAuth client ID published as an IOC

Your move

- Search your Workspace OAuth Token log for the published client ID
- Ask: can one user's consent speak for the whole tenant?
- Require admin approval for unverified apps and broad scopes

Where OAuth abuse shows up in Salesforce

Artifact	What to look for
LoginHistory / LoginEvent	Login sub-type 'OAuth Refresh Token', application name, source IP
API, RestApi, BulkApi events	connected_app_name, row counts, QueryMore paging, failed-request spikes
User agents	Script and CLI agents from an app that normally runs from the vendor's servers
Setup Audit Trail	Connected apps installed, permissions and profiles changed
Connected Apps OAuth Usage	Who is authorized right now: screenshot it before you revoke

Without Event Monitoring you get a handful of log types with one day of retention.



"Our vendor was breached": the first 24 hours

1. Find every grant to that vendor across all your SaaS, not just the one named
2. Export the relevant logs now, before revocation and before retention expires
3. Snapshot grants, scopes, integration users and IP restrictions
4. Revoke tokens and grants; disable or lock the integration user
5. Hunt the full retention window by client ID, app name, vendor IPs and user agents
6. Search exposed objects for secrets, then rotate and hunt each secret's use
7. Brief legal and comms: extortion and downstream phishing are coming



05

Rabbit hole #3: API exposure

Your SaaS data is full of keys, and your integrations don't do MFA.



TRUSTEDSEC

Your tickets are a secrets vault

AKIA

The AWS access key prefix

UNC6395 searched for other credentials in stolen Salesforce records

A breach that started in Salesforce can spread into AWS, Snowflake or any other co-mingled system.

Search, rotate, then hunt

- Scan case descriptions, notes, comments, chat, tickets and wiki pages
- Start with known prefixes:
- AKIA – AWS Access key
- ghp – GitHub Personal token
- xoxb – Slack bot token
- sk – API key often used in OpenAI
- And connection strings
- Rotate every secret found in exposed data, whether or not you see abuse
- Hunt each secret's use in its own platform: CloudTrail, Snowflake, GitHub
- Assume the stolen dataset gets mined again later



Non-human identities skip the front door

Integration users, service accounts and API tokens authenticate without a human, so MFA never enters the picture.



Baseline

Source IPs, user agents, objects touched, volume and time of day for each integration



Detect

Anything off-baseline from a known client ID is your best lead



Scope

One integration's permissions define the blast radius; know them in advance



AI agents hold tokens now

What changed

- Agents and MCP servers hold OAuth tokens to Jira, GitHub, Drive and CRM
- One grant can span code, documents, email and tickets
- 2026 research: a tampered agent config captured SaaS tokens and survived rotation

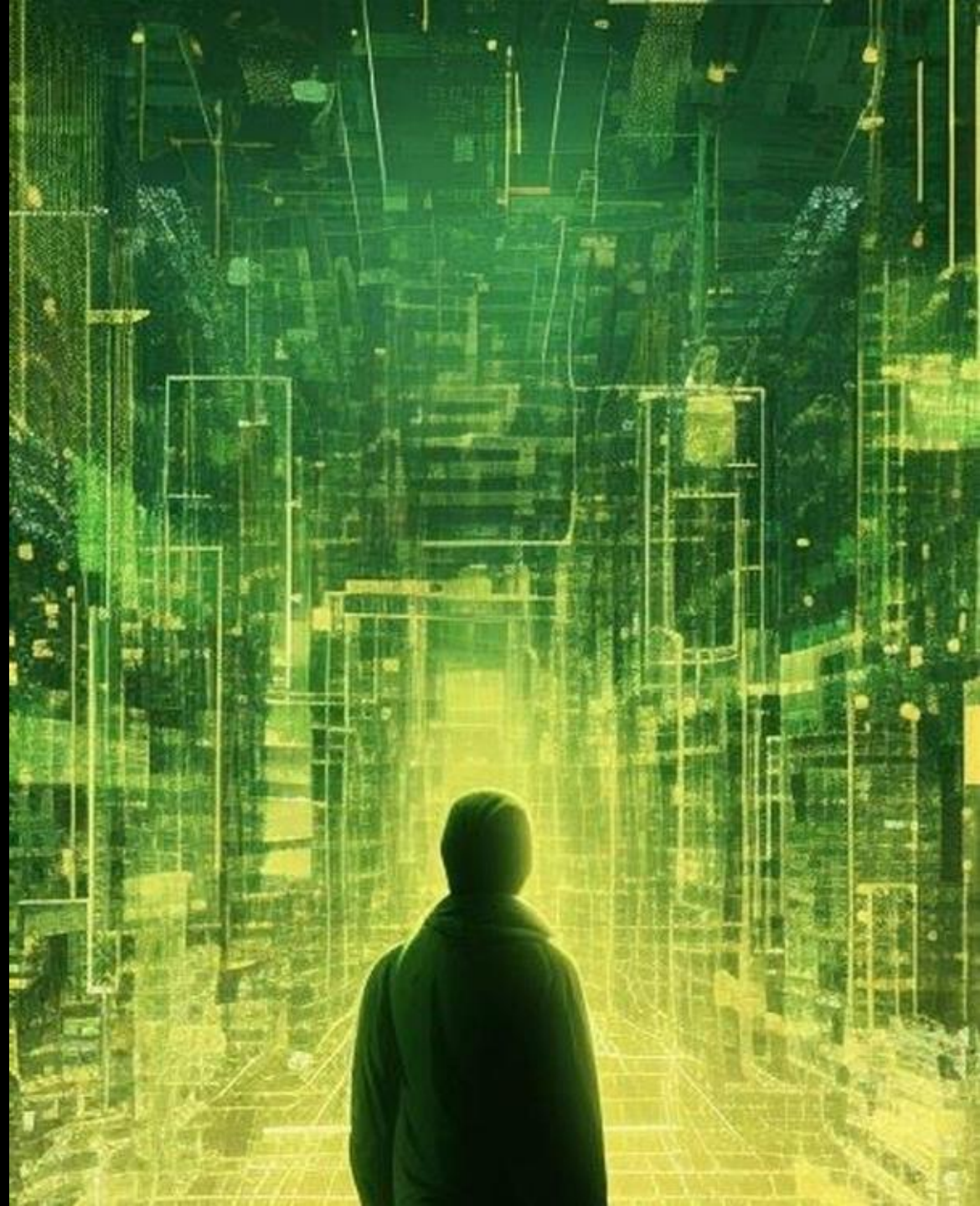
Questions for your investigation

- Was it the user, the agent, or whoever holds the agent's token?
- Which client ID does the SaaS log attribute it to?
- What agent and MCP configs exist on the endpoint? Rotate the grant, not just the token

06

Where to look

A map for when the playbook runs out.



TRUSTEDSEC

The SaaS evidence map

Platform	Primary logs	Pivot fields
Entra ID / M365	Sign-ins (interactive + non-interactive), Entra audit, Purview UAL, Graph activity	SessionId, UniqueTokenIdentifier, AppId, AuthenticationProtocol, IP / ASN
Google Workspace	OAuth Token, User (login), Admin, Drive	Client ID, scopes, IP, actor
Salesforce	LoginHistory / LoginEvent, API and Bulk API events, Setup Audit Trail	Connected app name, login sub-type, user agent, IP
Okta	System Log	External session ID, client IP, user agent, outcome
Snowflake	ACCOUNT_USAGE: LOGIN_HISTORY, QUERY_HISTORY, ACCESS_HISTORY	User, client type, IP, query ID
GitHub	Organization and enterprise audit log	Actor, token or OAuth app, workflow events



Revoke the right thing

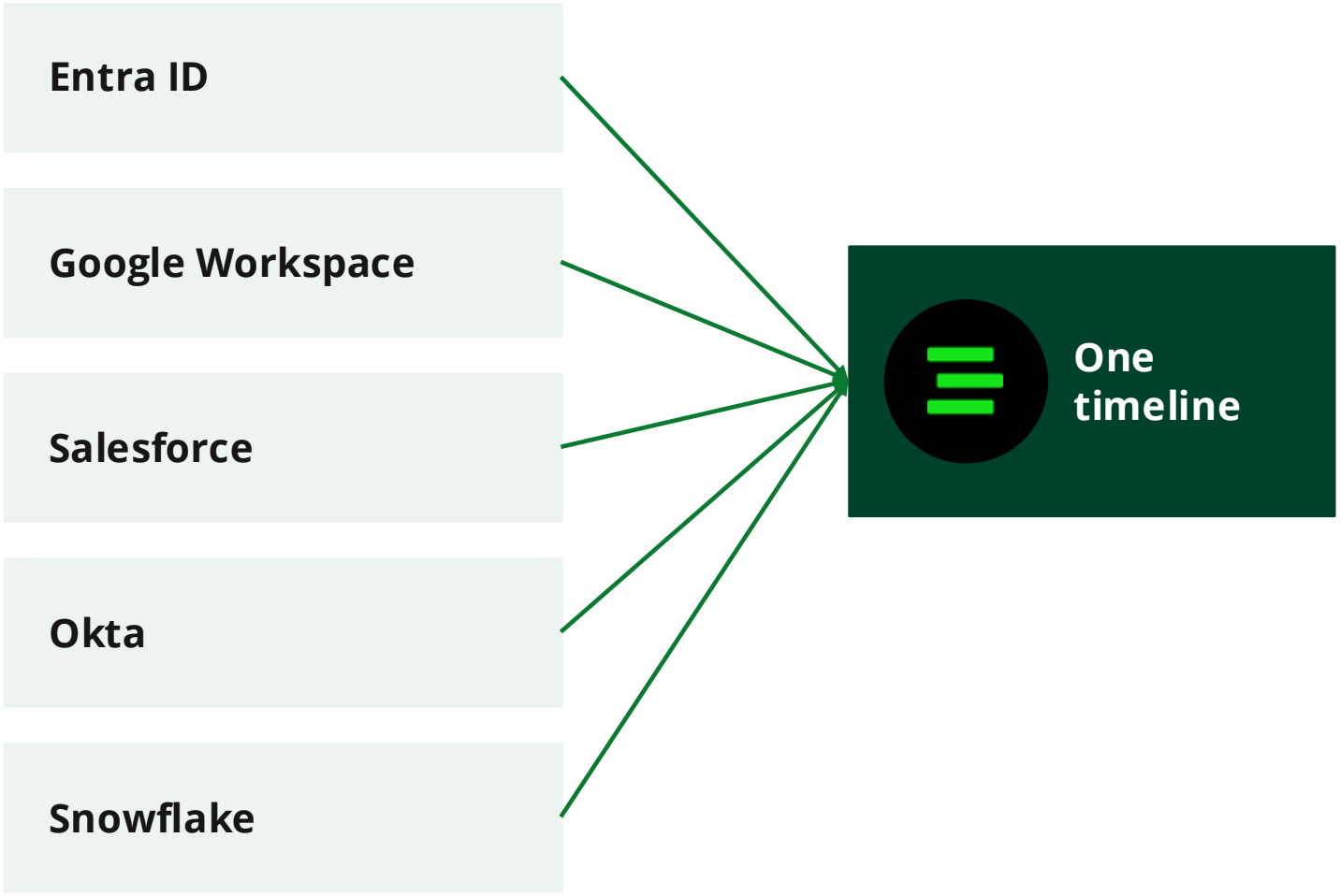
What was stolen	What actually evicts it
Session cookie	Revoke the user's sign-in sessions
Refresh token	Revoke sessions; confirm CAE-capable apps honor it
PRT / registered device	Delete the device, then revoke sessions
User OAuth consent	Remove the grant and block the app tenant-wide
Integrator tokens	Revoke the app's tokens, disable the integration user, reconnect with smaller scopes
API key or secret	Rotate, then hunt its use in the target platform



Preserve in parallel

Revocation is urgent, but some consoles stop showing what you just removed. Snapshot grants and export logs in the same hour.

Build one timeline across tenants



Normalize and pivot on

- Timestamps in UTC, with each source's lag noted
- Source IP and ASN
- User agent string
- Client, app or connected app ID
- Session and token identifiers
- Integration user and service account names

Get ready before the incident



Buy the logs

Event Monitoring, Audit Premium, Entra P1+ with diagnostics on



Keep the logs

Stream to your SIEM; keep at least a year



Inventory grants

Every connected app, scope and integration user, reviewed quarterly



Gate consent

Admin approval for apps and broad scopes



Pin integrations

IP-restrict connected apps; remove API access from broad profiles



Close old doors

Block device code flow where unneeded; shorten token lifetimes



Key Takeaways

The identity is the host

Investigate sessions, tokens and client IDs the way you used to investigate machines.

Tokens are the new credentials

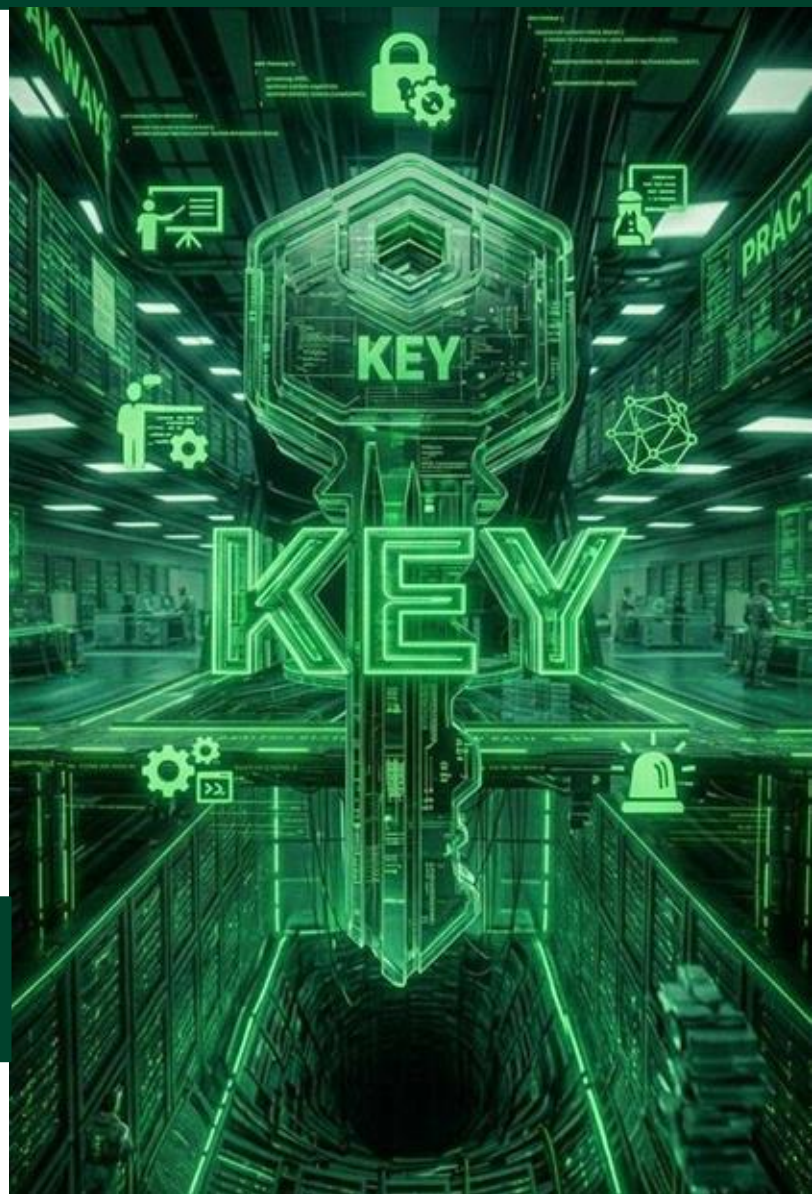
MFA protects the login; stolen tokens skip it entirely.

Your integrations are your attack surface

Every vendor holding a token can become your incident.

Logs expire; preserve in the first hour

Export and snapshot alongside containment, not after.



"If your organization runs SaaS, the incident is coming. This is how you investigate it."

Questions?

Cover Your SaaS

FOLLOW US @TRUSTEDSEC

The information security industry is constantly evolving – keep up by following TrustedSec's active social media, blogs, podcasts and webinars.

TRUSTEDSEC

https://www.linkedin.com/in/shanehartman/
https://x.com/H3xxEdit
https://www.youtube.com/@h3xxedit

