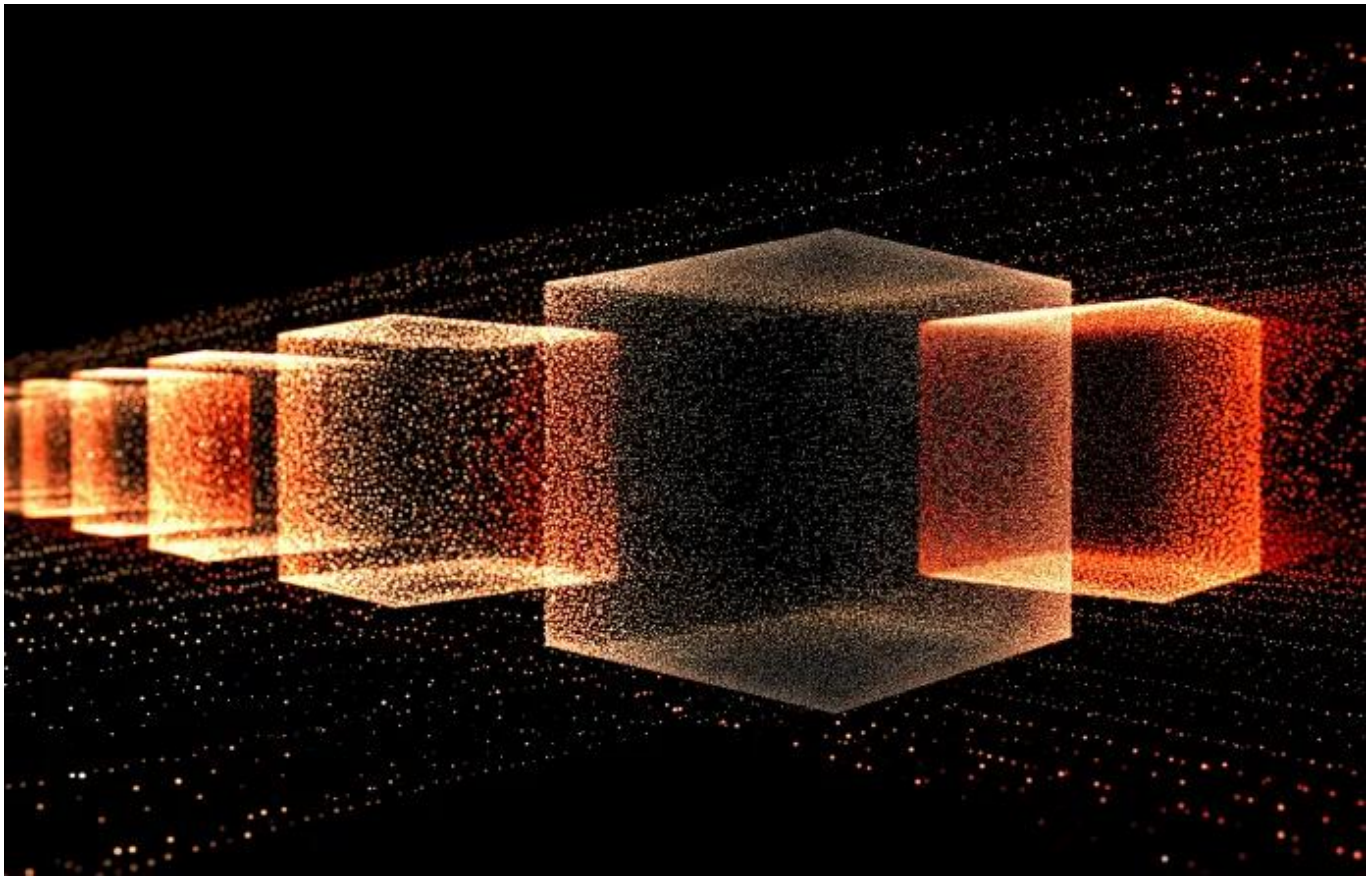
OpenCTI for Analysts: From Feeds to Findings (2 of 4)



How to Structure Threat Data in OpenCTI

By Mario Rojas, Senior Security Researcher, ORYXLABS

This post is the second in a series on OpenCTI to help anyone new to Cyber Threat Intelligence (CTI) learn how to use it as their primary Threat Intelligence Platform (TIP).

In this series' first post, we focused on getting comfortable navigating OpenCTI and building trust in what you see (markings, confidence, provenance, recency). This post is about turning that trust into structure.

In OpenCTI terms, this is modeling: taking evidence and deliberately shaping it into knowledge objects that reflect how an operation actually works.

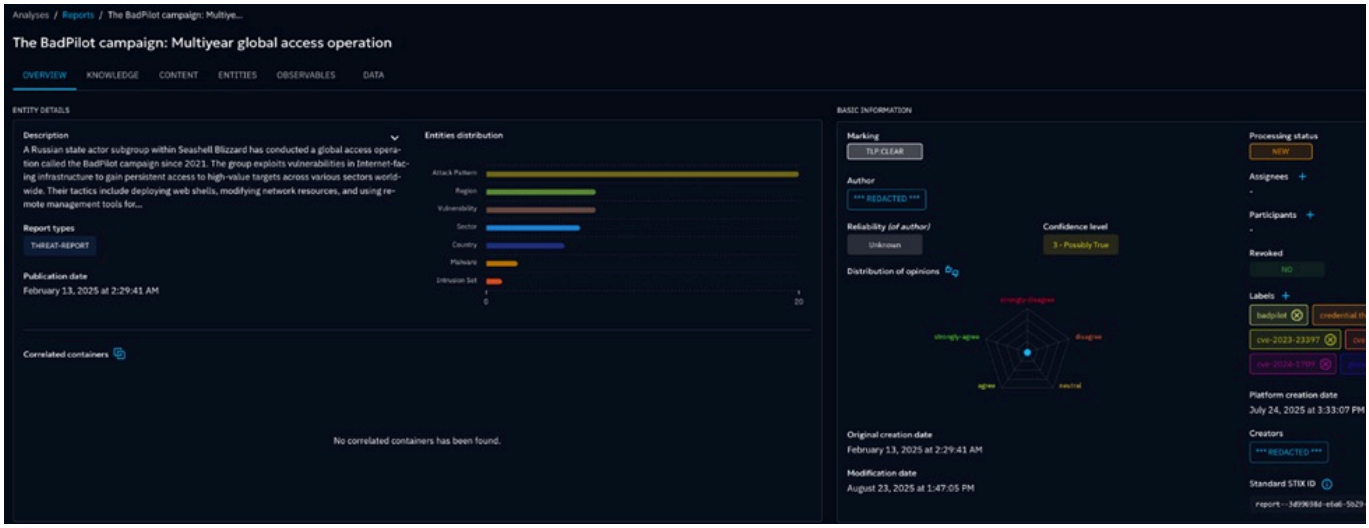
In this edition you will learn how to take raw items, whether they arrive from external feeds or your own telemetry, and decide what belongs as an observable, what deserves to become an indicator, and what should be promoted into higher-order knowledge like infrastructure, tools, vulnerabilities, or a campaign.

Technical analysis

To keep this practical, we will anchor the walkthrough in Microsoft's reporting about the BadPilot campaign, a multiyear initial access operation linked to a subgroup within Seashell Blizzard. We will also continue using the OpenCTI demo instance so you can follow along.

Start with evidence, then model outward

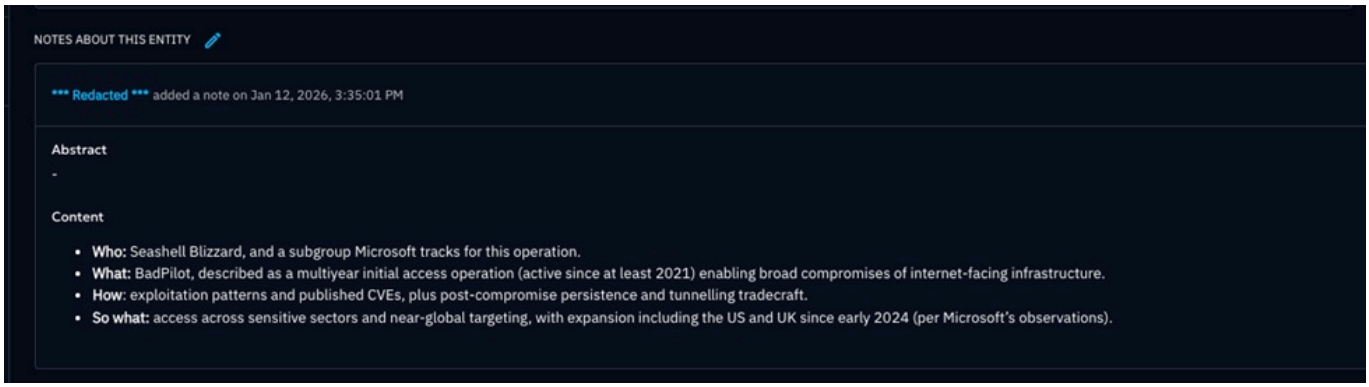
Begin in **Analysis → Reports**. Open the report entry for Microsoft's BadPilot write-up and treat it as the spine of your model.



From the report, pull out a short set of anchors (you do not need everything):

- **Who**: Seashell Blizzard, and a subgroup Microsoft tracks for this operation.
- **What**: BadPilot, described as a multiyear initial access operation (active since at least 2021) enabling broad compromises of internet-facing infrastructure.
- **How**: exploitation patterns and published CVEs, plus post-compromise persistence and tunnelling tradecraft.
- **So what**: access across sensitive sectors and near-global targeting, with expansion including the US and UK since early 2024 (per Microsoft's observations).

This is the difference between "I saw an IOC" and "I can explain an operation." Use OpenCTI notes to capture the few details that justify your decisions, so someone else can retrace your reasoning later, as shown in the image below.



Use a simple promotion ladder

Think of OpenCTI as two layers. The bottom layer is Stix Cyber Obervables (**SCOs**) – observables like domains and hashes. The top layer is Stix Domain Observables (**SDOs**) – things that explain meaning, like Threat Actors, Campaigns, Malware, Tools, Infrastructure, Vulnerabilities, and Reports.

As you pull items from the report (and from your observations), promote them deliberately:

- **Observables**
  Raw facts such as a domain, IP, URL, file hash, email address, or file. Keep them as observables when you can't defend action yet.
- **Indicators**
  Items you are willing to operationalise for detection or blocking. These must have context, time relevance, and a reason to exist.
- **Infrastructure**
  Attacker-controlled resources or stable patterns of reuse. Infrastructure is where "one-off IOC" becomes something you can track over time.
- **Tools and Malware**
  Capability families. The BadPilot report is useful here because it shows how legitimate tools can become part of an actor's persistence stack, not just "malware." Microsoft describes RMM software used to maintain access and a unique persistence method known as ShadowLink, plus OpenSSH deployed with a unique key in some cases.
- **Campaign**
  A named operation with timing and purpose supported by reporting. BadPilot is explicitly described as a tracked campaign.

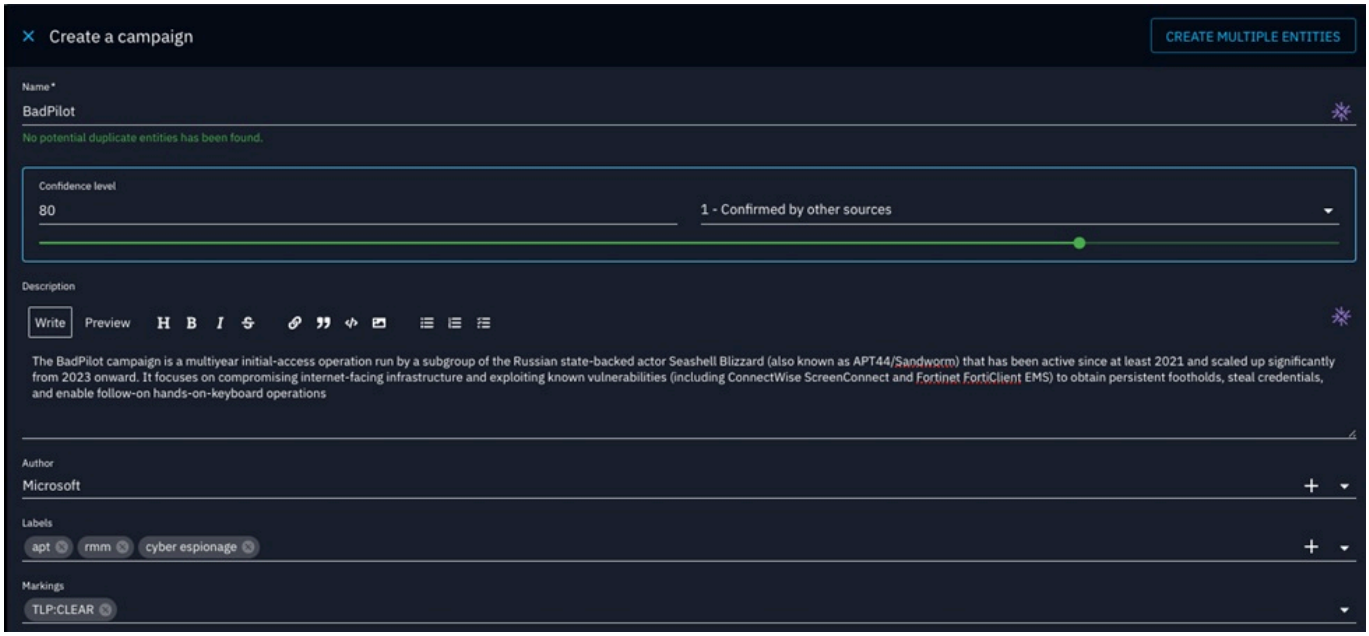
This ladder prevents the most common modelling mistake: turning everything into an indicator. Your job is to move upward only when you have enough evidence to justify it.

Model BadPilot in a clean, defensible way

Here's the minimum structure that creates real value:

1) Campaign and actor

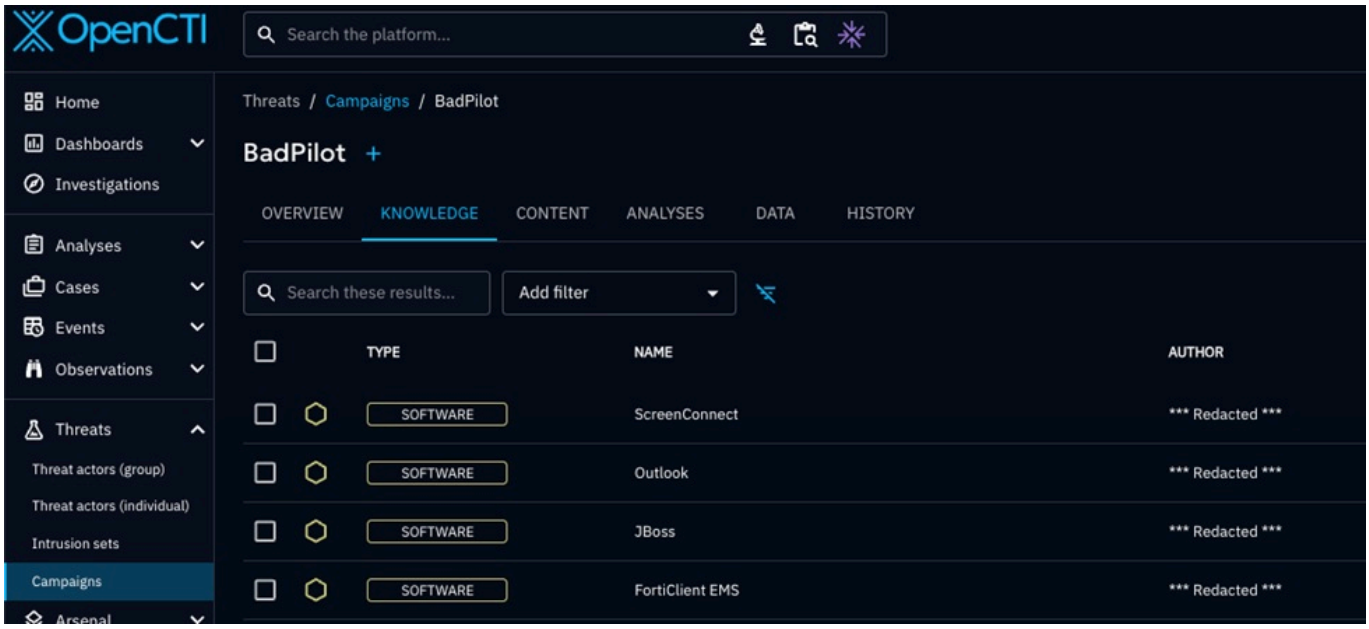
- Create or locate a **Campaign** object for "BadPilot."
- Link it to the **Seashell Blizzard** intrusion set and to the Microsoft report as the primary reference.



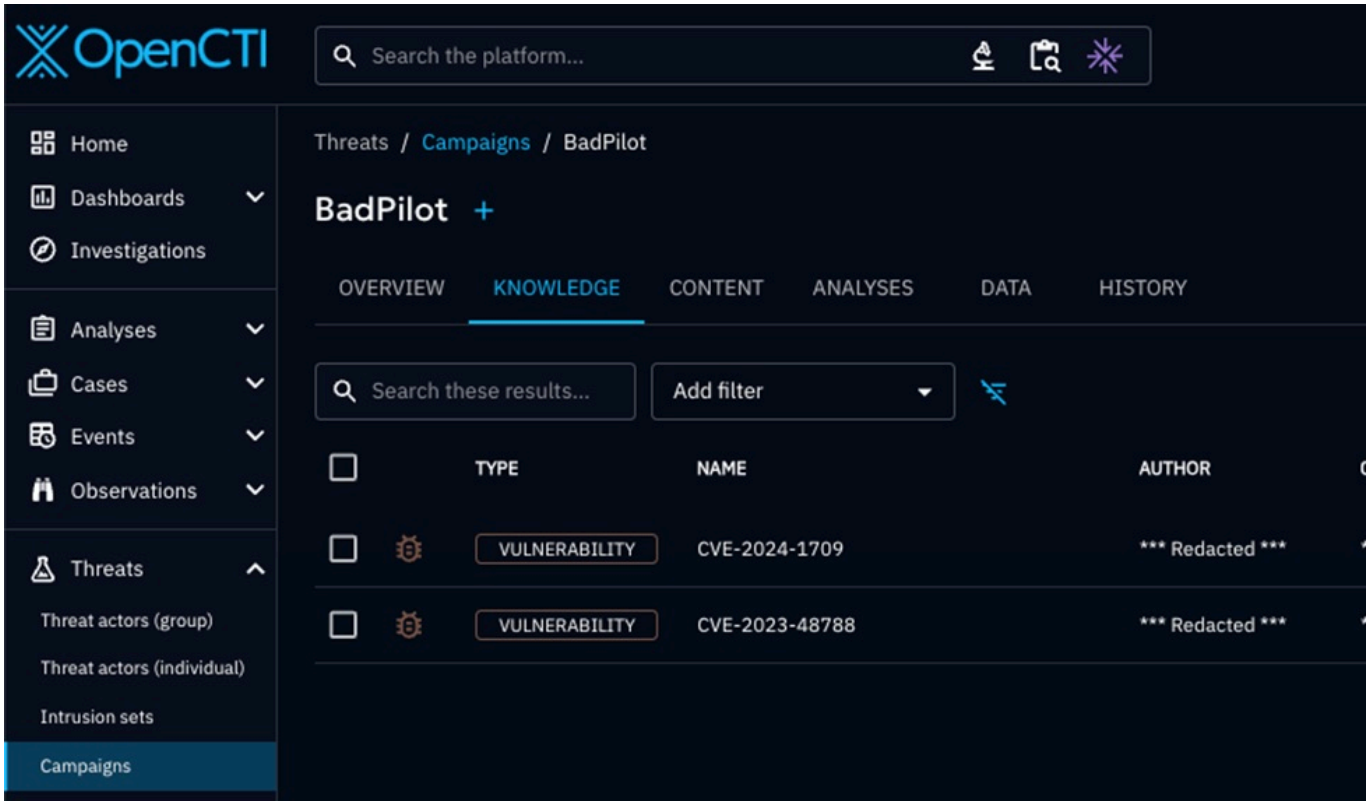
In the example above, we created the campaign using its original name “BadPilot” and adjusted the confidence level to match our internal policies.

2) Access vectors as vulnerability knowledge

Microsoft notes exploitation activity since at least 2021 and lists vulnerabilities used by the subgroup, including (among others) Exchange, Zimbra, OpenFire, TeamCity, Outlook, ScreenConnect, and FortiClient EMS, and notes JBOSS without an exact CVE.



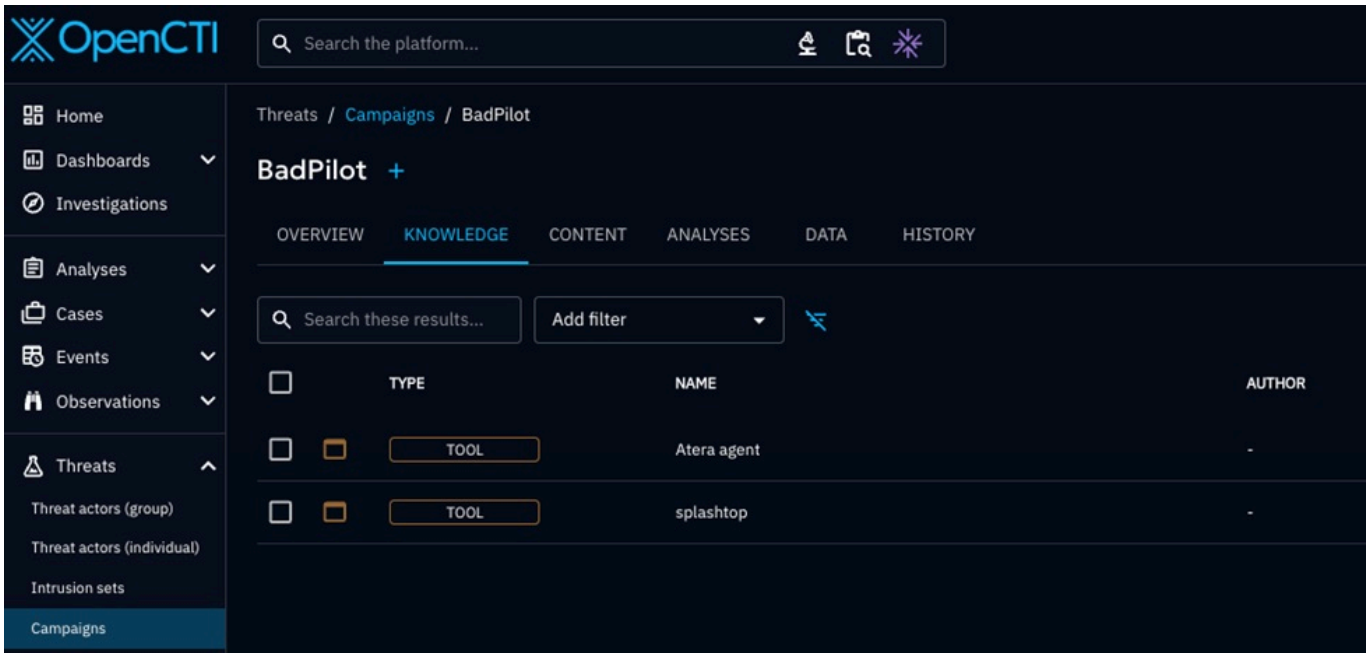
It also highlights that since early 2024, exploitation of ScreenConnect (CVE-2024-1709) and FortiClient EMS (CVE-2023-48788) expanded targeting to include the US and UK.



Modelling tip: Vulnerabilities are a high-value way to connect intelligence to action. When you link a campaign to exploited CVEs, your audience immediately understands what to prioritise in exposure management.

3) Post-compromise tooling and persistence

Microsoft describes the subgroup deploying RMM software (Atera Agent, Splashtop Remote Services) after exploitation to retain command and control while blending in as legitimate utilities. It also describes a subset of cases with OpenSSH and a persistence method Microsoft calls ShadowLink.



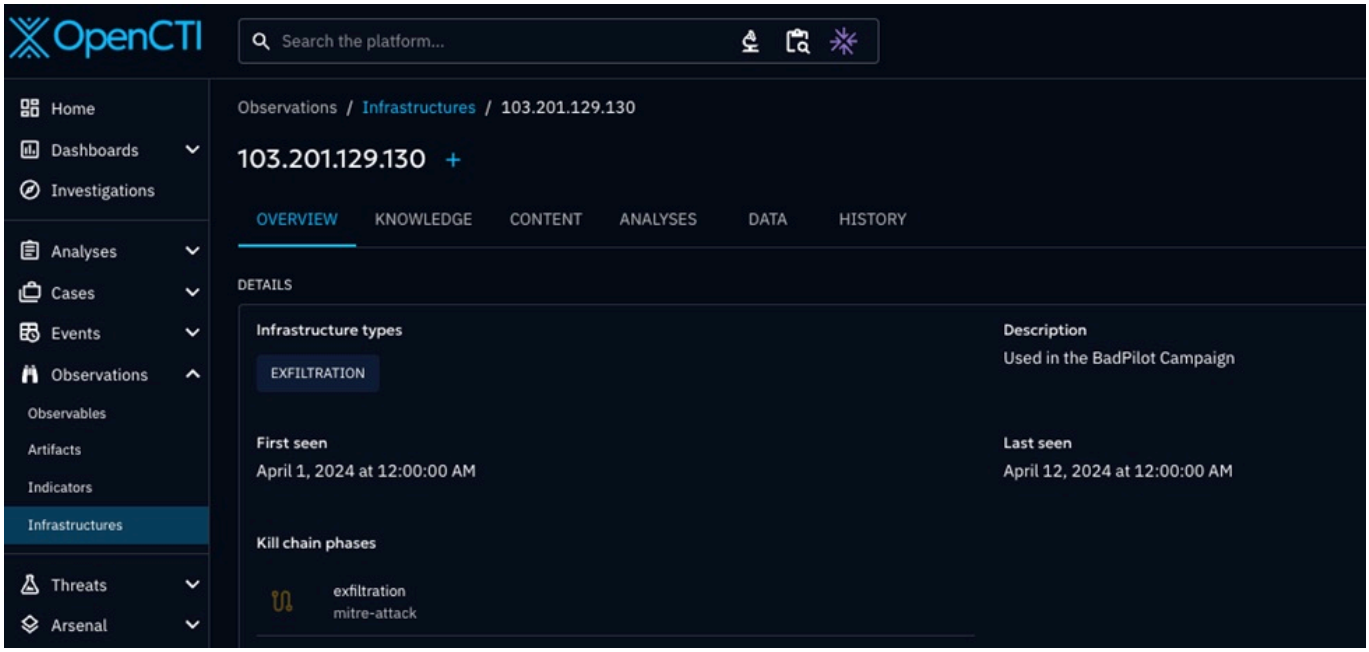
This is where you can model “Tooling used for persistence” without overclaiming. You’re not saying “Atera equals compromise.” You’re saying “Atera was used in this campaign’s

post-exploitation pattern, so unexpected installs in sensitive zones are worth scrutiny.”

4) Infrastructure and tunnelling patterns
Microsoft reports that when the actor identifies strategically valuable targets, it deploys tunnelling utilities such as Chisel, plink, and rsockstun, and provides example naming conventions used for Chisel deployments.

Microsoft also provides a table of tunnelling IP infrastructure with first/last observed windows and explicitly notes that some IPs may no longer be used and are provided for historical and forensic understanding.

This is the exact place analysts go wrong if they are not careful. The right move is to preserve the time window in your notes and avoid treating historical infrastructure as “block now.”



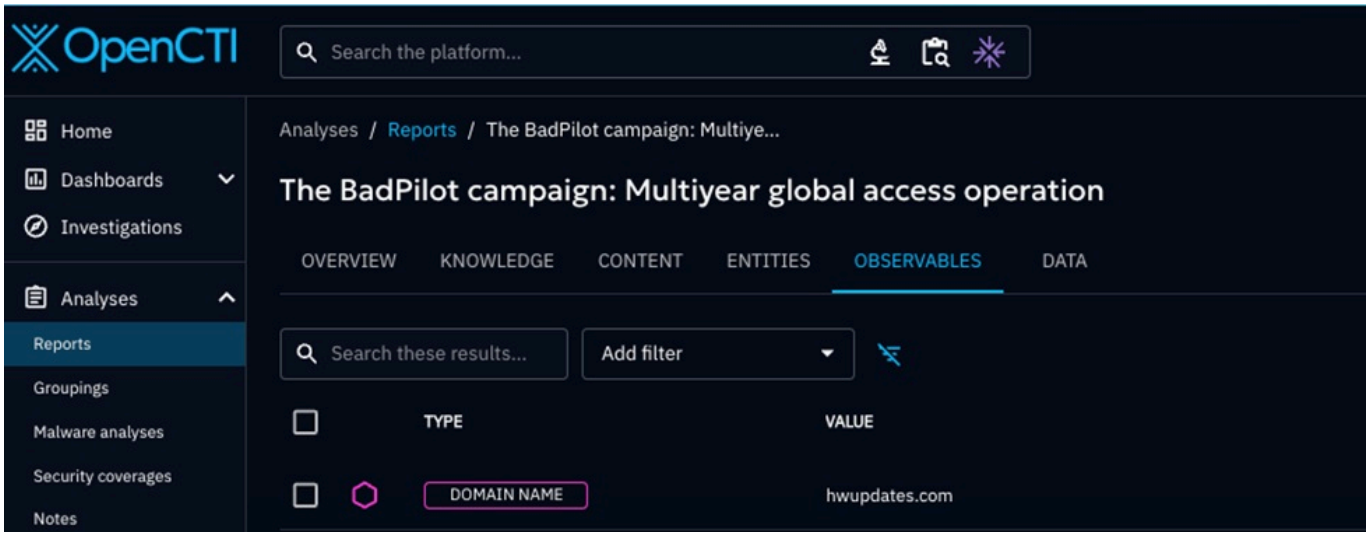
The image above shows why **last seen** matters. An IP might be mentioned in a report from years ago, but if OpenCTI shows it was **seen again recently**, that changes the priority. In this example, the underlying reporting is older, but the last seen value is from a few days ago, so it is worth a closer look.

5) Credential collection infrastructure

Microsoft also tracks actor-controlled infrastructure linked to credential collection via modified OWA sign-in pages, including hwupdates[.]com and cloud-sync[.]org.

This is a clean example of promotion logic:

- The domain or IP begins as an observable.
- It becomes an indicator only if you have a relevant detection use case and a time window.
- It becomes infrastructure when it is described as actor-controlled and tied to a technique across victims.



Normalise like a pro

As you model, keep these habits:

- Add a **one-sentence rationale** to anything you promote (“Promoted to Infrastructure because Microsoft links it to credential collection via modified OWA pages in this campaign.”).
- Capture **time bounds** when a source provides them (first observed, last observed, or at least a review date).
- Keep duplicates under control by consolidating references rather than creating parallel objects.

This is what keeps your OpenCTI environment useful six months from now.

TIP, not PIT

If your threat intelligence platform (TIP) feels noisy and unreliable, it has most likely become a pit where information goes to die. OpenCTI is not meant to hold everything you ingest. It is meant to hold what you understand. The fastest way to break a TIP is to promote without context. The fastest way to fix it is to promote slowly and deliberately, and to require a one-sentence explanation for why anything belongs.

Conclusion

The goal of modelling is not to create more objects. It is to create clarity. Start from a report you trust, build outward in small steps, and promote only what you can defend. When you do this consistently, your OpenCTI environment becomes a decision engine: a place where a campaign, an exploited CVE, a persistence tool, and a small set of infrastructure leads all connect to the same story.

In the next post, we will take this BadPilot model and pivot it into an investigation that leads to a clean action note for SOC and IR.

Remember, there’s always more intelligence to uncover.