Chasing Badgers: Infrastructure Hunting with OpenCTI and Shodan



By Mario Rojas, Senior Security Researcher, ORYXLABS

Note: This post departs from our usual analytical format and adopts a more exploratory, field-journal style. The shift reflects the investigative and hands-on nature of infrastructure .hunting, where agility and narrative context are essential to conveying process and insight

A few days ago, I was reading Symantec's report on cyber operations in the Middle East and something in the Seedworm (a.k.a. MuddyWater) section caught my eye: this Advanced .Persistent Threat (APT) is using Brute Ratel C4 (BRc4) in its operations

There have been numerous reports of different threat groups deploying Brute Ratel C4 (BRc4). However, this is the first time we've seen public confirmation of MuddyWater leveraging it. The group has a history of abusing legitimate tools in its operations, so its decision to add BRc4 to its arsenal was worth a closer look. Here we share some of the passive OSINT techniques used to track "Badger" infrastructure (Brute Ratel's term for its .endpoint agents) so threat hunters can proactively identify and block it

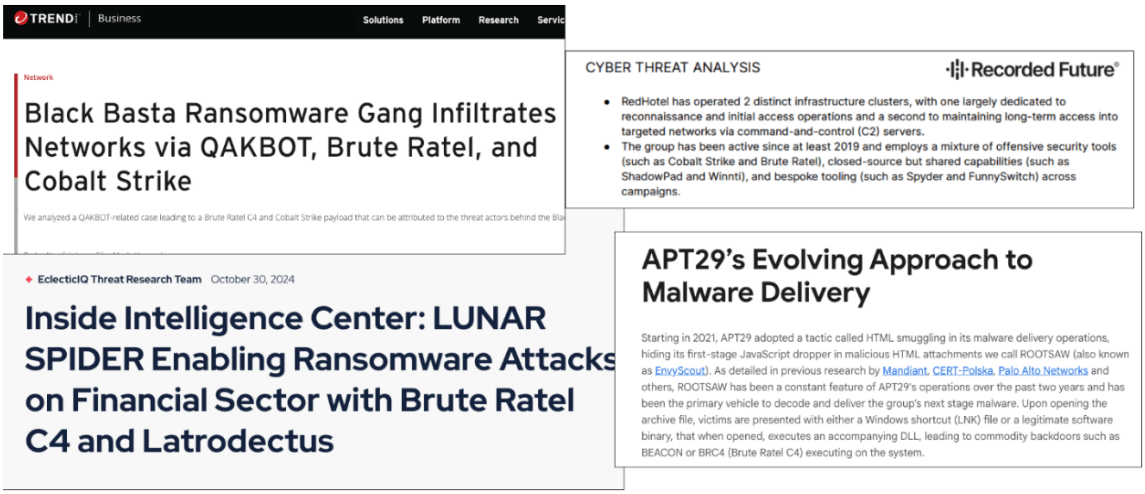
?Who is MuddyWater

MuddyWater (a.k.a. Seedworm, Earth Vetala, MERCURY, Static Kitten, TEMP.Zagros, Mango Sandstorm, TA450) is a cyber espionage group publicly attributed to Iran's Ministry of Intelligence and Security (MOIS). Active since at least 2017, the group has targeted government, telecom, energy, and other critical sectors across the Middle East, Europe and .North America

Its operations often involve exploiting known vulnerabilities for initial access, followed by the use of legitimate tools such as remote administration utilities and tunnelling software to blend in with normal network activity. The APT's toolset has included custom malware, .open-source frameworks, and now BRc4

A quick note on Brute Ratel

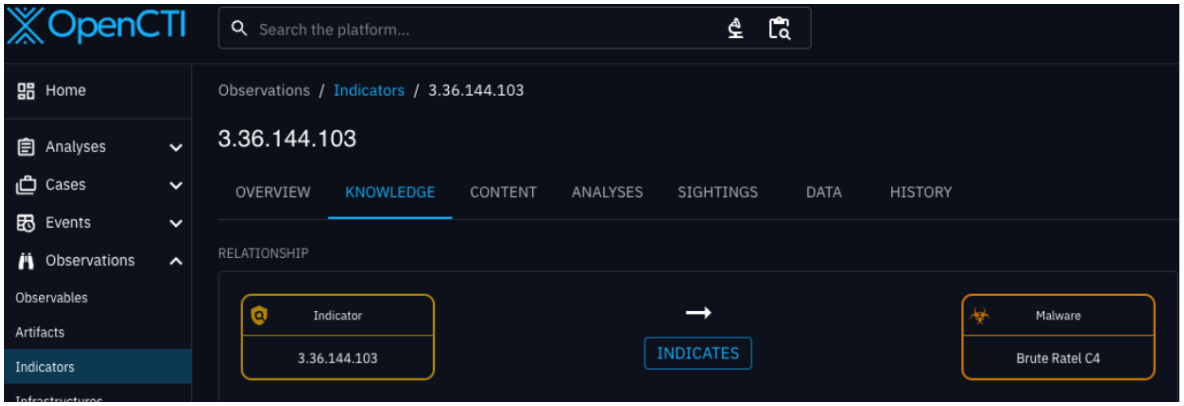
BRc4 is a legitimate red-teaming tool built for stealth. It employs techniques such as memory encryption [T1027], userland hook removal [T1562.006], and sleep masking [T1055.002] to avoid detection, features that make it attractive to both red teams and .threat actors



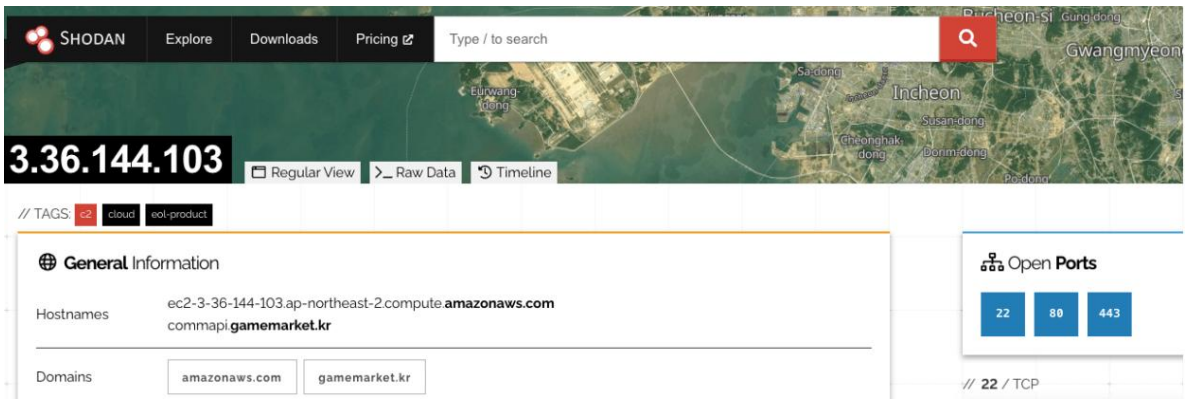
Since a cracked version was leaked in 2022, it has been abused in campaigns by threat groups such as Black Basta and LUNAR SPIDER, and in espionage operations tied to regions including the Middle East. These groups favour tools like Brute Ratel as their stealthy .payloads, which are often hard to catch. Its infrastructure, though? Not as much

Where the hunt begins

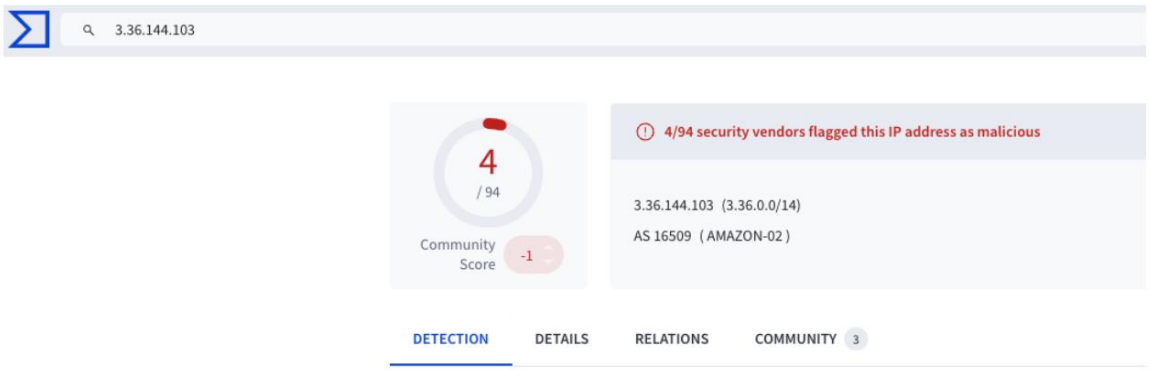
I decided to start the "hunt" by identifying a known BRc4 command-and-control (C2) instance from our OpenCTI platform, then used resources such as Shodan and VirusTotal to .track additional infrastructure



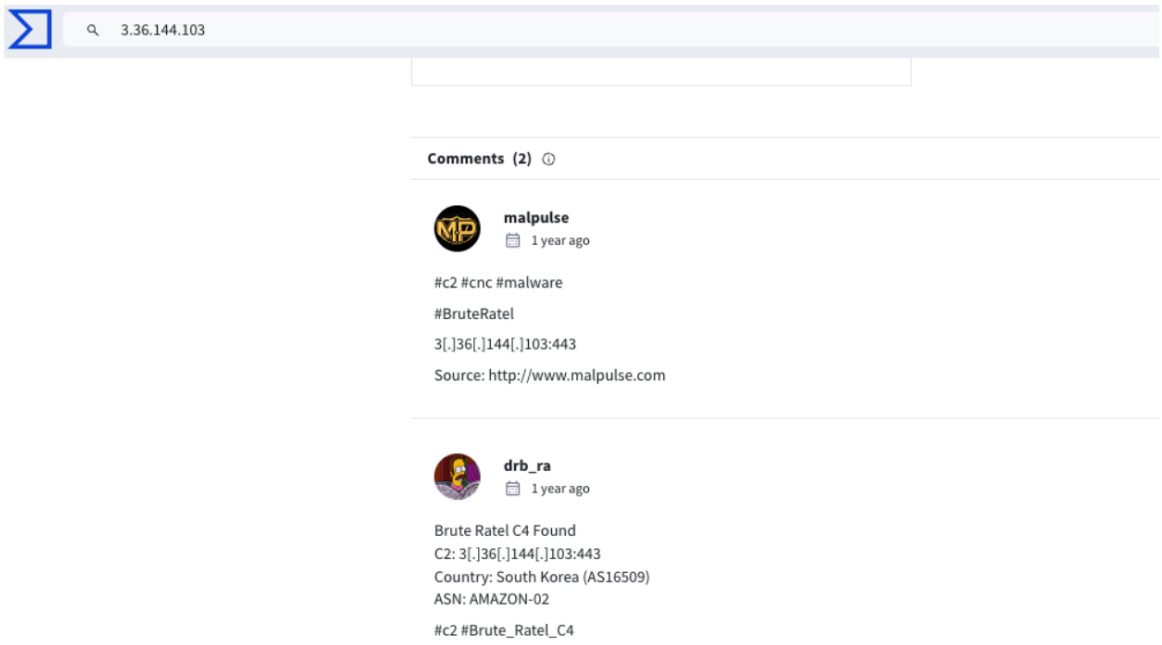
After locating the starting point (3[.]36[.]144[.]103), I turned to Shodan, which revealed a few open ports: 22 (SSH), 80 (HTTP), and 443 (HTTPS), and had already flagged the host as .a C2



Then, I moved to check VirusTotal for information on this host. Surprisingly, it showed only .(a few detections (4/94



Looking at the “Community” tab confirmed the presence of Brute Ratel on this host, which, …based on the timestamp, has been the case for at least a year. That is Persistence



The pivots

In OSINT, pivoting refers to using one piece of information to uncover related data, thereby effectively expanding the investigation beyond the initial data point. It’s time to jump back to Shodan and continue hunting. Among the myriad of data Shodan collects about the :hosts, there are a few "identifiers" we can use as pivot points

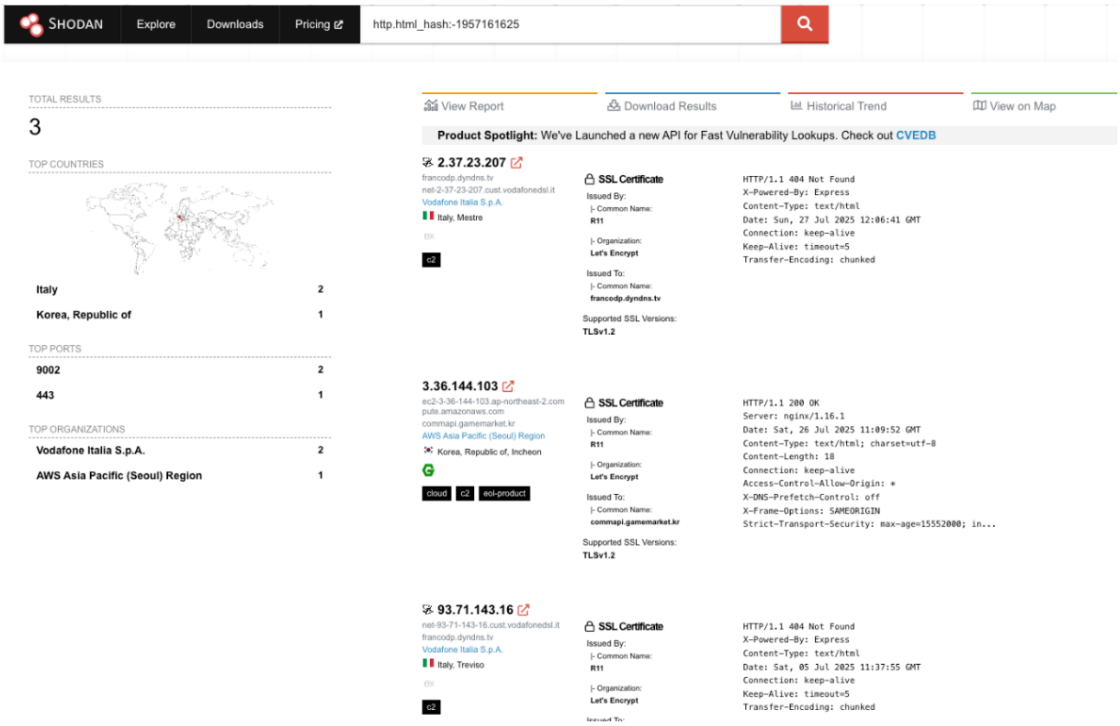
- HTTP Headers •
- HTML Hash •
- Certificate Details •
- JARM fingerprint •

HTML Hash

I decided to start with HTML hash - Shodan’s fingerprint of the actual HTML response body from the server. This is particularly useful when the same C2 toolkit serves a static or .templated page. You’ll find it under the 443 tab



Clicking on it runs a new query using the http.html_hash filter, which, in our case, returned .two more Badgers. Both were already tagged as C2, so no surprises there



Even though this was useful, I was hoping to find infrastructure that hadn't been flagged .yet, so I continued exploring other pivot points

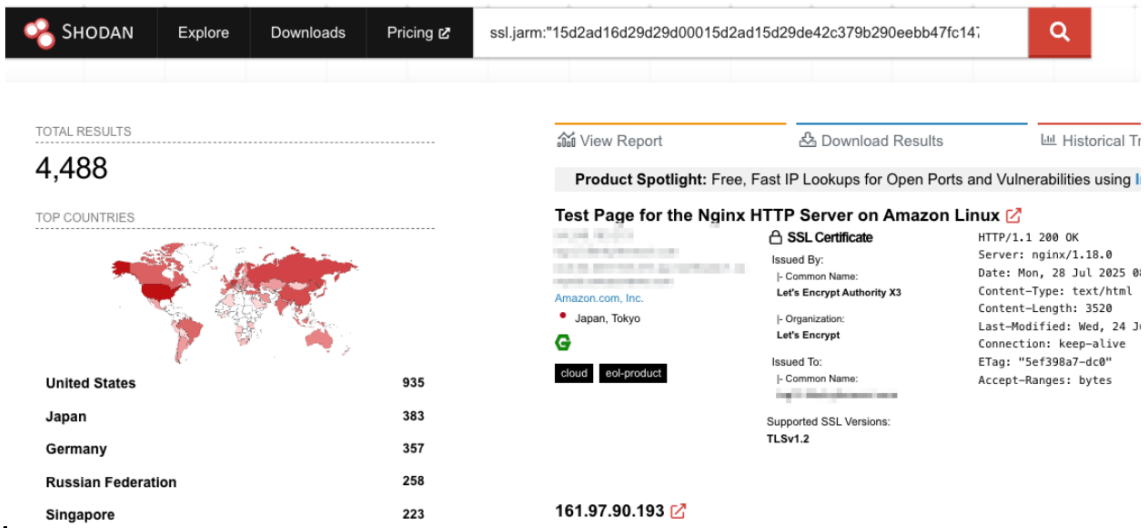
JARM

JARM is a TLS fingerprinting technique that hashes a server's behaviour across multiple handshake attempts. The idea is that similar configurations generate similar hashes; this is .especially true for tools such as Brute Ratel, which use embedded Go-based HTTP servers



:Shodan lists the JARM under raw data

.Clicking the JARM hash will run a new query



This hash returned close to 4,500 results. As you can see, JARM alone is too noisy, but when .combined with other data points, it can help filter down candidate infrastructures

HTTP Headers

From the same host, I pulled the HTTP headers. These can be surprisingly helpful. Brute Ratel often serves minimal content (same length, same type) with no real templating or .randomness

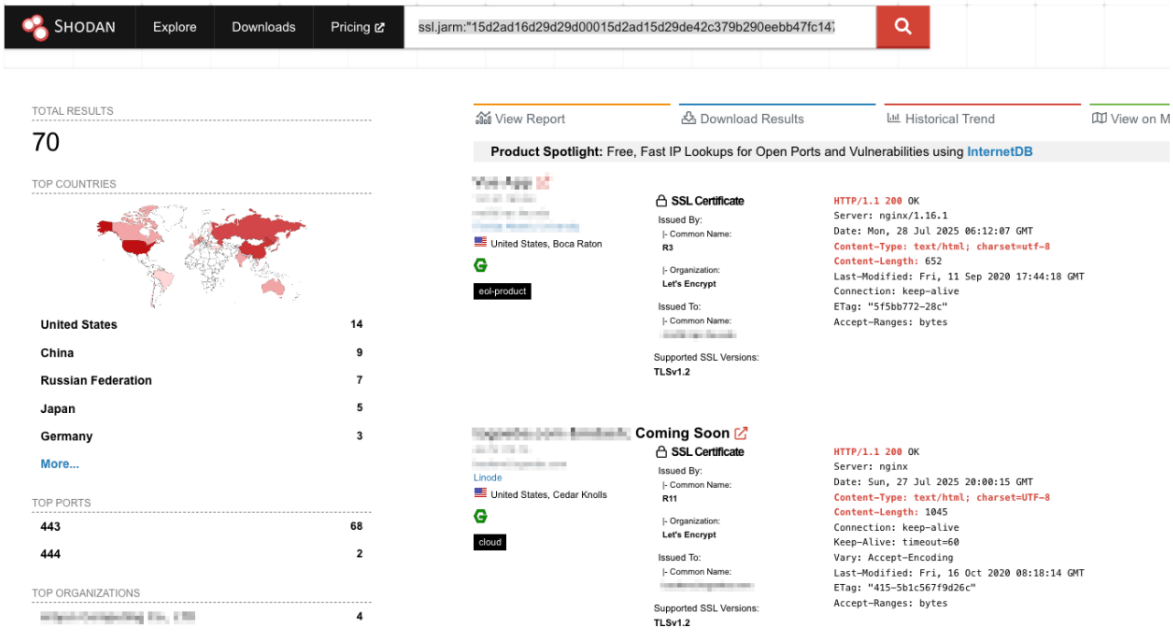
:Here is what this one returned

Instead of using the full range of Headers, I decided to take a subset that has worked in :previous hunts



- HTTP/1.1 200 OK
- Content-Type: text/html; charset=utf-8
- Content-Length: 18

That content length is interesting. I've seen multiple C2 servers return responses with exactly 14 or 18 bytes, which is an uncommon length for an HTTPS 200 OK response and can be used to easily spot suspicious instances. These Headers aren't unique on their own, .but paired with a JARM, they help filter noise significantly



:Here is the full search I used

```
ssl.jarm:"15d2ad16d29d29d00015d2ad15d29de42c379b290eebb47fc14746f97199ce"
HTTP/1.1 200 OK Content-Type: text/html; charset=utf-8 Content-Length: 18
```

This reduced potential Brute Ratel instances to around 70, a much better baseline than the

4,500 we started with. This is where combining pivot points really pays off, significantly .improving the quality of your findings

?What is next

As shown above, these simple pivots can help surface active C2 infrastructure that often :evades automated detection. From here, you could

- Cross-check in VirusTotal or ThreatFox to validate hits or identify overlaps with known .campaigns
- .Look for reused certificates or domains across other hosts
- .Drop IPs into passive DNS tools to check for rotation patterns or staging behaviour

These techniques aren't exclusive to Brute Ratel; you can apply the same approach to other C2 frameworks such as Cobalt Strike, Sliver, or Mythic. Go and try it against a known .IOC and see how far you can pivot

I hope you found the walkthrough useful. Let me know if you discover other Badger-related .infrastructure

.**Note:** All of this was passive. No interaction with suspected C2s

Confirmed BRc4 C2s

:Here is a short list of confirmed Brute Ratel servers found during this hunt

Conclusion

Brute Ratel continues to appear in active campaigns, including those tied to targeted operations in the Middle East. Its infrastructure doesn't always hide well, and passive hunting can expose far more than most feeds will reveal: starting from a single known C2, you can uncover dozens more using signals like HTML hashes, JARM fingerprints, and .response headers

If you're defending a network, blocking these C2s before they hit makes a significant difference. You can also use solutions such as DNS FIREWALL to block outbound connections before they reach malicious infrastructure, while DISCOVERY will track credential leaks and domain abuse, helping identify this type of infrastructure and complete .the picture

.Remember, there is always more intelligence to uncover