CVE-2024-46507: Yeti Platform Server-Side Template Injection (SSTI)

Chebuya

Vulnerability Overview

Affected Product Summary

Yeti is a *Forensic Intelligence platform and pipeline for DFIR teams*. It allows threat intelligence and DFIR teams to catalog, search, and link pieces of intelligence such as IP addresses, TTPs, and threat actors. With 10,000 pulls from Dockerhub and nearly 2,000 stars on GitHub, it is a fairly popular tool for DFIR practitioners.

In this blog post, I will detail 2 security flaws that, if used in conjunction, could lead to unauthenticated remote code execution on the application server. The consequences of a successful server compromise could include attackers collecting intelligence on who/what the threat intel teams were monitoring and modifying/destroying IoCs/observables.

The proof-of-concept exploit can be found in our CVE Github repository (https://github.com/RhinoSecurityLabs/CVEs/tree/master/CVE-2024-46507).

The patch for these CVEs can be found on the yeti-platform repository: SSTI (https://github.com/yeti-platform/yeti/commit/235f70357e3f8214897072c36470f1402d490607), Insecure secret (https://github.com/yeti-platform/yeti-docker/commit/4a67458c9dcf348951e4d921e5c5af28b33cee9e)

Vendor: https://yeti-platform.io/ (https://yeti-platform.io/)

Product: YETI (Your Everyday Threat Intelligence)

Confirmed Vulnerable Version:

- *SSTI:* 2.0 – 2.1.11
- *Insecure Secret:* 2.0 (e351163 (https://github.com/yeti-platform/yeti-docker/commit/e351630e40f8d2376d72a5448e1478384566d7d#diff-9e03cadea3957ac1d036704143830a26095c219daa83015c77c3957fa51143eeR11)) – 2.1.11 (4a67458 (https://github.com/yeti-platform/yeti-docker/commit/4a67458c9dcf348951e4d921e5c5af28b33cee9e))

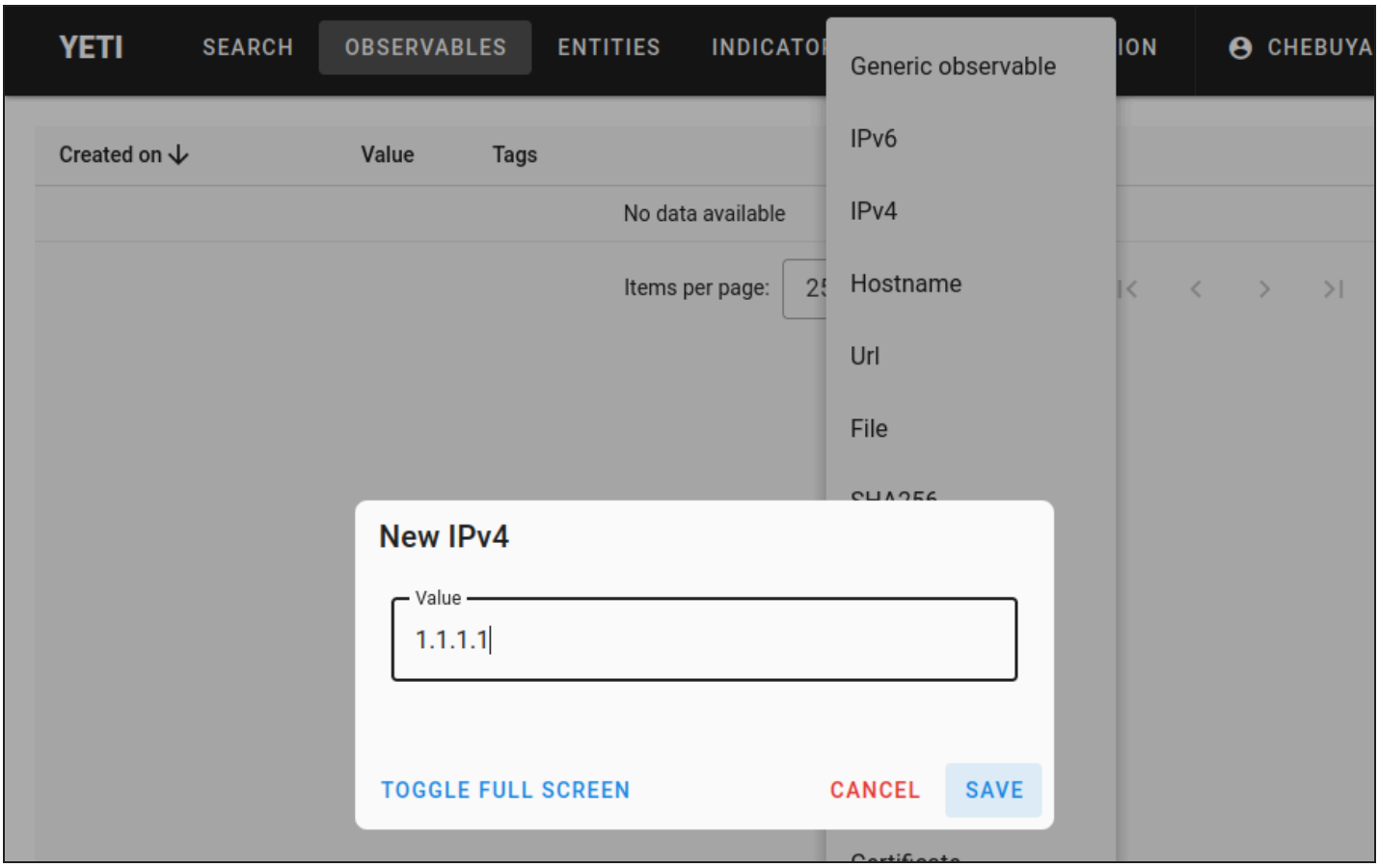
Fixed Version: 2.1.12

Product Link: https://github.com/yeti-platform/yeti (https://github.com/yeti-platform/yeti)

CVE-2024-46507: SSTI Leads to RCE

Yeti allows its users to create custom report templates that users can export their IoCs/intel into a custom format. The template content is not sanitized before being processed on the backend, leading to the possibility of code execution.

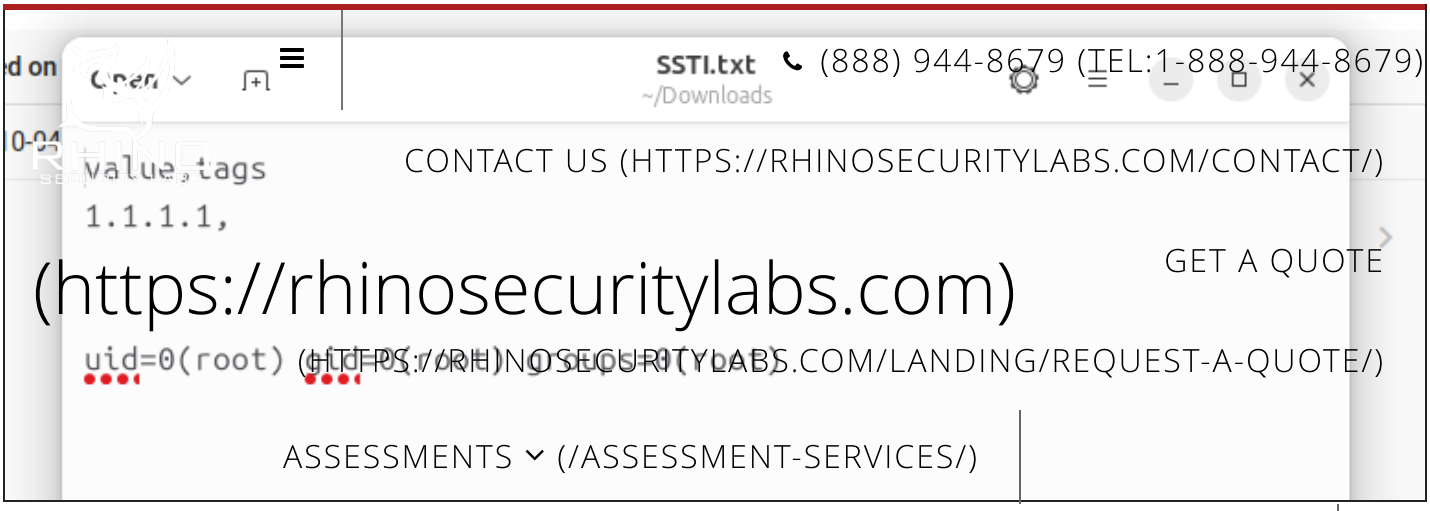
Templates can be used to export observables (hashes, domains, IP addresses). To successfully instantiate an export, there needs to be an observable too export. To create one, can go to Observables -> New Observable -> Save



To trigger the rendering of our malicious template, we need to select our newly created observable to export it with our malicious template.

After exporting the observables, a .txt file is downloaded.

In the .txt file, we see the output of the command that we chose in the malicious template (in this case, id).



CVE-2024-46508: Use of Static Insecure Secret

Yeti makes use of JWTs for authentication. The application is most commonly deployed using a docker-compose file. I was surprised during the installation process, there is no need to ever actually open the .env file to change anything, it was as simple as cloning the repo, running docker-compose, and the application was up in a few seconds. Furthermore, the documentation does not refer to the .env file anywhere. Concerningly, in the .env file the JWT secret was configured as “SECRET” and did not seem to be replaced at runtime. Given the fact that the .env file is not required to be modified in the installation process or mentioned in documentation, there was a good chance that instances deployed in the wild would have not changed the static secret. Furthermore, the variable naming did not make it readily apparent that it was being used for a JWT secret.

```
→ prod git:(main) cat .env
YETI_DOCKER_ENVFILE=.env

# Yeti database / internals
YETI_REDIS_HOST=yeti-redis
YETI_REDIS_PORT=6379
YETI_REDIS_DATABASE=0
YETI_ARANGODB_HOST=yeti-arangodb
YETI_ARANGODB_PORT=8529
YETI_ARANGODB_DATABASE=yeti
YETI_ARANGODB_USERNAME=root
YETI_ARANGODB_PASSWORD=
YETI_AUTH_SECRET_KEY=SECRET
YETI_AUTH_ALGORITHM=HS256
YETI_AUTH_ACCESS_TOKEN_EXPIRE_MINUTES=30
YETI_AUTH_ENABLED=True
YETI_SYSTEM_PLUGINS_PATH=./plugins

# Timesketch configuration
YETI_TIMESKETCH_ENDPOINT=http://timesketch-dev:5000
YETI_TIMESKETCH_USERNAME=dev
YETI_TIMESKETCH_PASSWORD=dev
→ prod git:(main)
```

With a known JWT secret, attackers can generate valid tokens and bypass authentication. Combined with the ability to execute commands from an authenticated context, this was more concerning.

Conclusion

We found Yeti interesting as the primary users would be blue teams. While reviewing the codebase, we discovered CVE-2024-46507 allowing authenticated users to execute code on the application server. While looking for other ways to exploit this vulnerability besides brute forcing/obtaining valid credentials for the application, we discovered a static JWT secret being used in the docker deployment process (CVE-2024-46508). Combining both of these issues, it would be possible to obtain unauthenticated code execution on applications in the default configuration. We want to thank the Yeti team for working with us to remediate this vulnerability.

We have added a proof-of-concept to our CVE Github repository (https://github.com/RhinoSecurityLabs/CVEs/tree/master/CVE-2024-46507) that demonstrates this vulnerability. As always, feel free to follow us on Twitter or LinkedIn and join our Discord server for more releases and blog posts.

Twitter: https://twitter.com/rhinosecurity (https://twitter.com/rhinosecurity)

LinkedIn: https://www.linkedin.com/company/rhino-security-labs/ (https://www.linkedin.com/company/rhino-security-labs/)

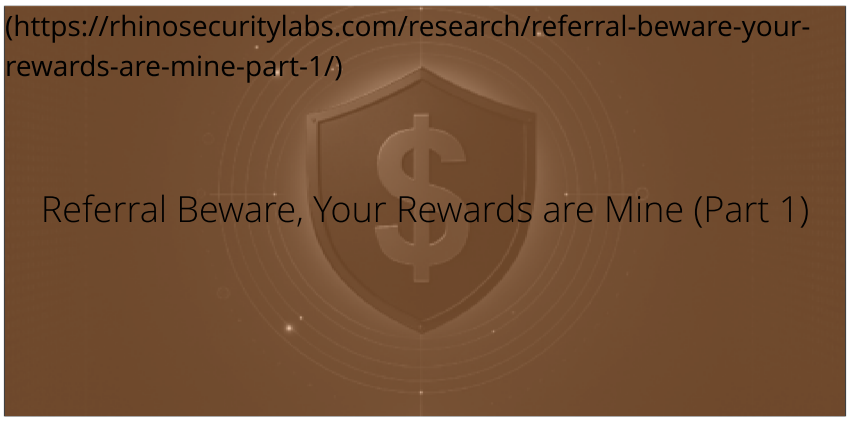
Discord: https://discord.gg/TUuH26G5 (https://discord.gg/TUuH26G5)

Researcher/Author: https://x.com/_chebuya (https://x.com/_chebuya)

Vulnerability Disclosure Timeline

Date	Note
8/28/2024	Issue reported to Yeti.
9/8/2024	Yeti acknowledged the vulnerability.
9/9/2024	Rhino provided Yeti with additional vulnerability information.
9/19/2024	Rhino obtains a CVE from MITRE.
10/1/2024	Yeti patches (https://github.com/yeti-platform/yeti/commit/235f70357e3f8214897072c36470f1402d490607) the vulnerability.

Related Resources



(https://rhinosecuritylabs.com/research/infoblox-multiple-cves/)
Multiple CVEs in Infoblox NetMRI: RCE, Auth Bypass, SQLi, and File Read Vulnerabilities
(https://rhinosecuritylabs.com/research/cve-2025-26147-authenticated-rce-in-denodo/)
CVE-2025-26147: Authenticated RCE In Denodo Scheduler

Interested in more information?

20603

Contact Us Today »

ASSESSMENT SERVICES (HTTPS://RHINOSECURITYLABS.COM/ASSESSMENT-SERVICES/)
Network Penetration Test (https://rhinosecuritylabs.com/assessment-services/network-penetration-testing/)
Webapp Penetration Test (https://rhinosecuritylabs.com/assessment-services/web-penetration-testing/)
AWS Cloud Penetration Testing (https://rhinosecuritylabs.com/assessment-services/aws-cloud-penetration-testing/)
GCP Cloud Penetration Testing (https://rhinosecuritylabs.com/assessment-services/gcp-penetration-testing/)
Azure Penetration Testing (https://rhinosecuritylabs.com/assessment-services/azure-penetration-testing/)
Mobile App Assessment (https://rhinosecuritylabs.com/assessment-services/mobile-app-assessment/)
Secure Code Review (https://rhinosecuritylabs.com/assessment-services/secure-code-review/)
Social Engineering / Phishing Testing (https://rhinosecuritylabs.com/assessment-services/social-engineering/)
Vishing (Voice Call) Testing (https://rhinosecuritylabs.com/assessment-services/social-engineering/vishing-assessments/)
Red Team Engagements (https://rhinosecuritylabs.com/assessment-services/red-team-engagement/)

INDUSTRIES (HTTPS://RHINOSECURITYLABS.COM/INDUSTRY/)

Healthcare (https://rhinosecuritylabs.com/industry/healthcare/)

Finance (https://rhinosecuritylabs.com/industry/financial/)

Technology (https://rhinosecuritylabs.com/industry/technology/)

Retail (https://rhinosecuritylabs.com/industry/retail/)

RESOURCES (HTTPS://RHINOSECURITYLABS.COM/RESOURCES/)

Technical Blog (https://rhinosecuritylabs.com/blog-technical/)

Strategic Blog (https://rhinosecuritylabs.com/blog-strategic/)

Example Pentest Report (https://rhinosecuritylabs.com/landing/penetration-test-report/)

Technical Research (https://rhinosecuritylabs.com/research-and-vulnerability-disclosure/)

Vulnerability Disclosures (https://rhinosecuritylabs.com/research-and-vulnerability-disclosure/)

Disclosure Policy (https://rhinosecuritylabs.com/company/vulnerability-disclosure-policy/)

Penetration Testing FAQ (https://rhinosecuritylabs.com/assessment-services/penetration-testing-faq/)

Support: AWS Pentest Form (https://rhinosecuritylabs.com/assessment-services/support-aws-penetration-testing-form/)

()

COMPANY (HTTPS://RHINOSECURITYLABS.COM/COMPANY/)

Leadership (https://rhinosecuritylabs.com/company/leadership/)

Blog (https://rhinosecuritylabs.com/blog/)

Careers (https://rhinosecuritylabs.com/careers/)

Company Principles (https://rhinosecuritylabs.com/careers/rhino-company-principles/)

Contact Us (https://rhinosecuritylabs.com/contact/)

Get a Quote (https://rhinosecuritylabs.com/request-a-quote/)

RSS Feed (https://rhinosecuritylabs.com/blog/feed/)

ABOUT US

Rhino Security Labs is a top penetration testing and security assessment firm, with a focus on cloud pentesting (AWS, GCP, Azure), network pentesting, web application pentesting, and phishing.
With manual, deep-dive engagements, we identify security vulnerabilities which put clients at risk.

Endorsed by industry leaders, Rhino Security Labs is a trusted security advisor to the Fortune 500.

info@rhinosecuritylabs.com (mailto:info@rhinosecuritylabs.com)

(888) 944-8679 (tel:1-888-944-8679)

Rhino Security Labs, Inc

(888) 944-8679 (TEL:1-888-944-8679)

CONTACT US (HTTPS://RHINOSECURITYLABS.COM/CONTACT/)

GET A QUOTE

(https://rhinosecuritylabs.com)

(HTTPS://RHINOSECURITYLABS.COM/LANDING/REQUEST-A-QUOTE/)

ASSESSMENTS (/ASSESSMENT-SERVICES/)

INDUSTRIES (HTTPS://RHINOSECURITYLABS.COM/INDUSTRY/)

RESOURCES (HTTPS://RHINOSECURITYLABS.COM/RESOURCES/)

SECURITY BLOG (HTTPS://RHINOSECURITYLABS.COM/BLOG/)

COMPANY (HTTPS://RHINOSECURITYLABS.COM/COMPANY/)