ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 30 / 30 · ENTERPRISE EXPLOITATION

Stolen Cloud Keys, Backdoored WordPress

Inside the pax_cmd Chain

How a stolen Google Cloud service-account key becomes a WordPress backdoor, and what the operator's workspace shows

Two-week observation window ending 18 November 2026 · Prepared November 2026
Full Report · Public edition · Version 1.1

29
Code files on nine open web directories (21 unique SHA-256 digests)

2
Versions of the Python bucket-to-WordPress playbook (5,251 and 5,480 bytes)

2
Behinder-compatible PHP webshells, one for PHP 7 and one rebuilt for PHP 8

15
Third-party package files among the 29, which yield 13 strings that look like domains but are not

What Is in This Report. 11 chapters and two appendices.

How a stolen Google Cloud service-account key becomes a WordPress backdoor, and what the operator's workspace shows

01	Executive Summary	04
02	What Was Found: 29 Files on Nine Hosts	07
03	The pax_cmd Playbook: From Service-Account Key to Theme Backdoor	11
04	Behinder Twins and Workspace Utilities	14
05	Archived Crypto-Exchange Claims and the Operator Model	16
06	Who the Swarm Targeted, and Other Explanations	19
07	What Defenders Should Do	22
08	Detection Rules	24
09	Indicators	27
10	Scope and Limits	30
11	Sources	32
A	Indicators of compromise (4)	34
B	Detection rules (4 files)	35

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense recovered a two-version Python playbook from an exposed attack archive. It uses a stolen Google Cloud service-account key to overwrite WordPress theme files inside a storage bucket that feeds a live web root. A hidden `pax_cmd` parameter is then meant to turn every page into a command gate. Twin Behinder webshells show the same fix-and-retry habit. Restrict bucket writes to deployment accounts, remove workstation service-account keys, and block `pax_cmd` requests at the WAF.

CHAPTER 01

Executive Summary

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense recovered a two-version Python playbook from an exposed attack archive. It uses a stolen Google Cloud service-account key to overwrite WordPress theme files inside a storage bucket that feeds a live web root. A hidden `pax_cmd` parameter is then meant to turn every page into a command gate. Twin Behinder webshells show the same fix-and-retry habit. Restrict bucket writes to deployment accounts, remove workstation service-account keys, and block `pax_cmd` requests at the WAF.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04
The Study in Numbers 04

01 Executive Summary

Many WordPress platforms now keep theme and media files in cloud object storage and sync them into the web root. That design is convenient, but it moves the line between cloud identity and server code. This report shows how one operator exploits that line, using 29 code files left on nine open web directories.

1. **Bucket write access turns a stolen service-account key into WordPress code execution.**
   Two Python scripts from an exposed ARTEX archive, tied to the archive's iGaming task (1,231 recorded worker runs), mint a Google OAuth token from a stolen key. They are written to overwrite a theme's `404.php` and `functions.php` inside an iGaming staging bucket. The second version fixes a key-parsing bug in the first.

2. **Twin Behinder webshells show the same repair cycle for PHP 8.**
   On a separate open directory, two small PHP webshells use AES-128 with a 16-byte XOR fallback and Behinder v3's public default key. The first version breaks on PHP 8.0. The second swaps in buffered `eval` to run on PHP 8.

3. **The same archive holds unverified claims against a crypto exchange.**
   The archive's crypto-exchange task records 2,178 worker runs against a Southeast Asian exchange. Its logs claim SQL injection into a linked database server, a webshell on a backup NAS and a harvested cloud key. None of these claims is confirmed.

Three Decisions for Leaders

1. **Treat buckets that feed web roots as production code.** Only CI/CD accounts should write to theme folders. Alert on any PHP object written to such a bucket, and baseline the deployed theme files.
2. **Remove long-lived service-account keys from workstations.** Move developers to short-lived federated tokens, strip project-editor rights, and block new key creation by policy.
3. **Layer web, cloud-audit and PHP runtime controls.** Block `pax_cmd` requests, alert when PHP workers start shells, and patch PHP to 8.2.35, 8.3.36, 8.4.27 or 8.5.12.

The Study in Numbers

Number	What it is
29	Code files on nine open web directories (21 unique SHA-256 digests)
2	Versions of the Python bucket-to-WordPress playbook (5,251 and 5,480 bytes)
2	Behinder-compatible PHP webshells, one for PHP 7 and one rebuilt for PHP 8
15	Third-party package files among the 29, which yield 13 strings that look like domains but are not
1,231	Worker runs the archive records for its iGaming task, the one behind the playbook
2,178	Worker runs the archive records for its crypto-exchange task
731	Nodes (795 edges) in our analytical model of the wider operator infrastructure

How to use this report.

Security leaders can read Chapters 1, 6 and 7. Cloud, DevOps and CMS teams should read Chapters 3 and 7. SOC and detection engineers should start with Chapters 8 and 9. CTI analysts will find the workspace and infrastructure detail in Chapters 2, 4 and 5.

CHAPTER 02

What Was Found: 29 Files on Nine Hosts

IN SHORT

Nine open web directories held 29 code files. Only four are offensive scripts; fifteen are ordinary package files that would flood a watchlist with false domains if taken at face value.

INSIDE THIS CHAPTER

Five kinds of file 08
Package files are noise, not indicators 09
Owners are not always operators 09

02 What Was Found: 29 Files on Nine Hosts

Sandbox runs of compiled malware show how an implant calls home. They rarely show how an operator logs in to a cloud API or fixes a broken script. Code left in an open web directory can show both. We read the files as text. We did not run any of them, and we did not contact any system named in them.

Open directories mix custom attack scripts with whole Python virtual environments and developer agent folders. Reading every string as an indicator produces noise. So the first step is to sort the files by what they are.

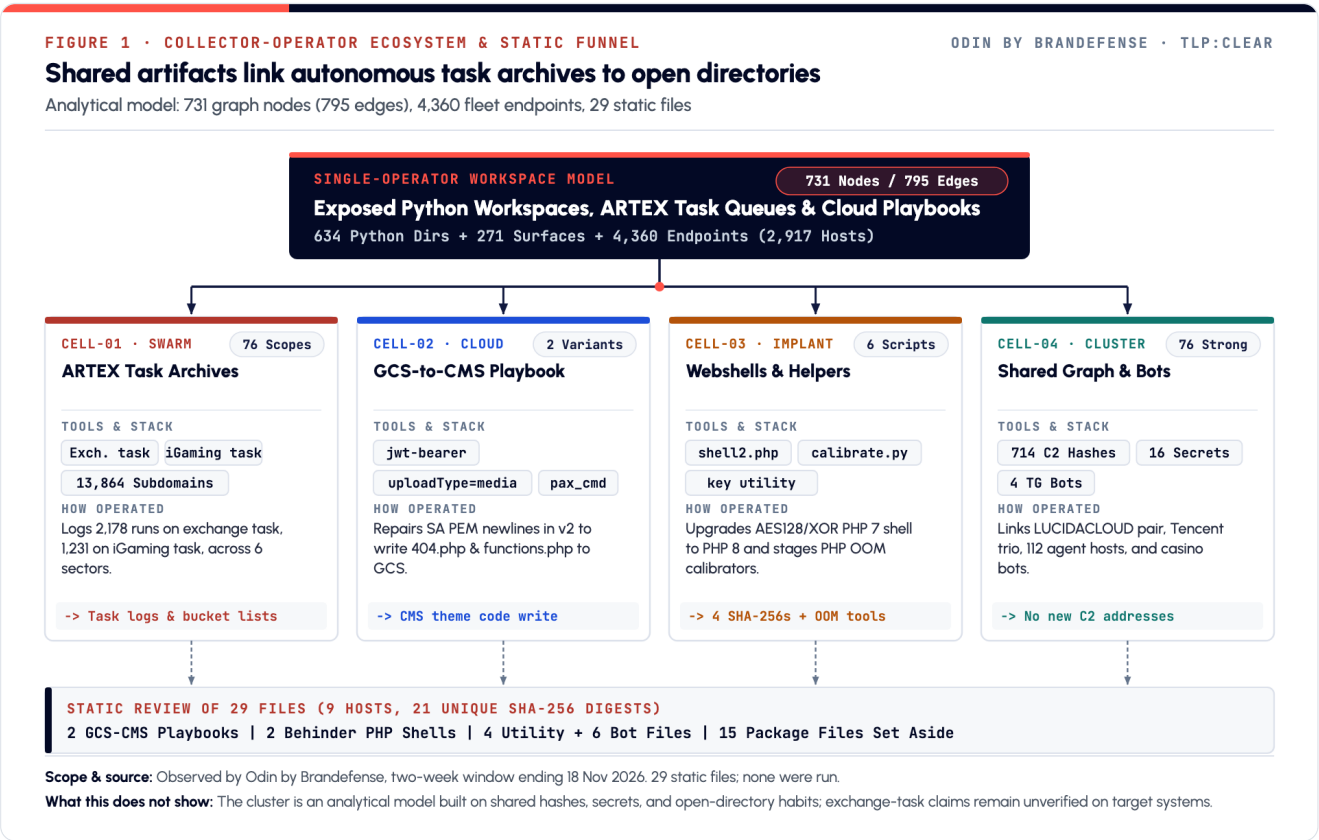


Figure 1: Shared artifacts link autonomous task archives to open directories

Five kinds of file

The 29 files are 24 Python scripts, 2 PHP files, 2 JavaScript files and 1 shell script. They hash to 21 unique SHA-256 digests. Seven duplicates come from Linux virtual environments that link `/venv/lib` to `/venv/lib64`. One more is a second copy of `calibrate.py`.

Class	Files	Unique SHA-256	Hosts	Size range (bytes)	What it is
Offense playbook	2	2	1	5,251 to 5,480	Bucket-to-WordPress `pax_cmd` deployers for the iGaming task
Webshell pair	2	2	1	569 to 755	Behinder-compatible AES-128/XOR PHP 7 and PHP 8 webshells
Operator utilities	4	3	2	1,421 to 28,367	A `CVE-2026-109618` calibration script, a key generator and a rule generator
Agent and bot code	6	6	3	3,933 to 28,633	AI coding-agent folders and chat-bot scripts
Package files	15	8	3	2,976 to 43,572	`telnetlib3`, `numpy`, `pip` and a mini-program UI component



Figure 2: Static review separates four offense scripts from fifteen package files

Package files are noise, not indicators

More than half of the files (15 of 29, 51.7%) are third-party packages. A naive domain pattern pulls 13 strings out of them: docstring words and attribute names that happen to look like domains, such as Python's `logger[.]info` call. None of them is attacker infrastructure. Keep package paths out of indicator extraction before anything reaches a SOC watchlist.

Every network address inside the 29 files was already known from Odin's research into 4,360 autonomous-offense endpoints on 2,917 hosts (Monograph 11). The files added no new command-and-control addresses.

Owners are not always operators

The nine hosts play different roles. Three of them expose AI coding-agent folders, bot scripts or a `telnetlib3` virtual environment. Nothing on those three shows offensive tooling, so we treat their owners as users, not operators. All nine host addresses are withheld.

Host role	Files	Unique SHA-256	Main contents
Python package host	10	5	Python 3.13 `telnetlib3` files in `lib` and `lib64`
Bot host	6	4	Two Python bot modules with `numpy` and `pip`
AI agent host	3	3	Hermes agent folders and an Orca terminal shell hook
Archive host	2	2	`deploy_gcs_webshell.py` and `deploy_gcs_webshell2.py`
Webshell host	2	2	`/serve/shell.php` and `/serve/shell2.php`
Utility host	2	2	A key-generation script and a rule-generation script
CVE calibration host	2	1	`/CVE-2026-109618/tools/calibrate.py` (two copies)
UI component host	1	1	One mini-program JavaScript component
Chat-bot host	1	1	One QQ chat-bot script

CHAPTER 03

The pax_cmd Playbook: From Service-Account Key to Theme Backdoor

IN SHORT

When a cloud bucket feeds a WordPress web root, the right to write to the bucket is the right to change the site's code. The two deployer scripts are built around that fact and need no login to the web server itself.

INSIDE THIS CHAPTER

Version 1 breaks the key, version 2 fixes it 11

Listing the bucket and writing the theme 12

The verification probes 12

03 The pax_cmd Playbook: From Service-Account Key to Theme Backdoor

Modern WordPress hosting often runs stateless PHP containers. Themes and media live in cloud object storage, which is synced or mounted into `wp-content/`. Teams tend to treat the bucket as passive storage. Once executable `.php` theme files live there, cloud IAM controls who can change server code.

The two scripts sit in the exposed ARTEX archive, next to the records of its iGaming task. Read side by side, they show the operator fixing a crypto bug and then widening the backdoor to every page.

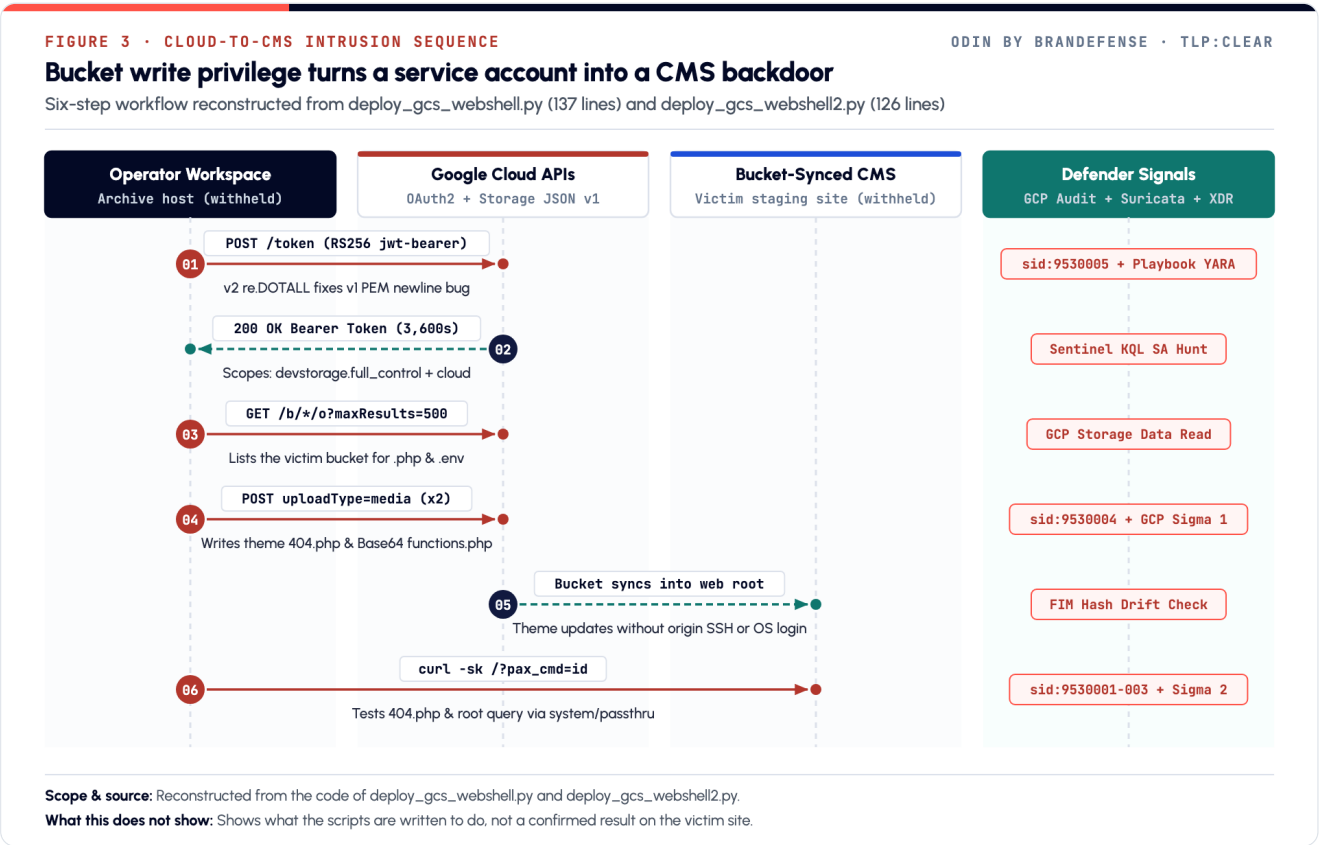


Figure 3: Bucket write privilege turns a service account into a CMS backdoor

Version 1 breaks the key, version 2 fixes it

Both scripts read a stolen Google Cloud service-account key from the task path `i21899/gcs_service_account.json`. A service-account key is a JSON file that holds an RSA private key in PEM format.

Version 1 (`deploy_gcs_webshell.py`) strips every newline from the JSON text before parsing it. PEM keys need those line breaks around the `BEGIN` and `END` markers. Without them the key is corrupt, and `RS256` JWT signing fails on the operator's machine.

Version 2 (`deploy_gcs_webshell2.py`) drops that shortcut. It pulls `private_key`, `client_email` and `private_key_id` out with `re.DOTALL` regular expressions, which keep the line breaks intact. With a valid key it builds a signed JWT assertion with a 3,600-second lifetime.

The assertion asks for two broad OAuth 2.0 scopes: `devstorage.full_control` and `cloud-platform`. The script posts it to `oauth2.googleapis[.]com/token` with grant type `urn:ietf:params:oauth:grant-type:jwt-bearer`. Google returns a one-hour bearer token. This is the standard server-to-server flow, used here with a stolen key (T1528, T1078.004).

Listing the bucket and writing the theme

With the token, both versions call the Cloud Storage JSON API at `storage.googleapis[.]com`. They list up to 500 objects in the victim's bucket. The loop looks for object names containing `.php`, `wp-config`, 404 and `.env`. Those names confirm that the bucket holds the live WordPress tree of an iGaming staging site.

The scripts then upload two files with `POST /upload/storage/v1/b/<bucket>/o?uploadType=media` and the header `Content-Type: application/x-php`. Both files go into the active theme's folder under `wp-content/themes/`. The bucket name, the theme name and the staging site are withheld because they identify the victim.

File written	What it does
Theme `404.php`	Replaces the theme's missing-page handler. Any unrouted request now checks for a `pax_cmd` value in `$_REQUEST`. A value is passed to `system`, then `passthru`, `exec` or `shell_exec` if the first is disabled.
Theme `functions.php`	WordPress loads the active theme's `functions.php` on every front-end and admin request. Version 2 ships this file Base64-encoded and keeps a stub of the original, so the theme still renders.

The gate has no IP restriction and no secret check. Anyone who finds the parameter can use it (T1505.003).

The verification probes

After the upload, each script is written to test the gate with `curl -sk` and the harmless command `id`. Version 1 tries three paths. Version 2 drops two of them and adds a root query, which relies on the `functions.php` hook.

Attribute	Version 1 (`deploy_gcs_webshell.py`)	Version 2 (`deploy_gcs_webshell2.py`)
Size and length	5,251 bytes, 137 lines	5,480 bytes, 126 lines
Key input	`i21899/gcs_service_account.json`	`i21899/gcs_service_account.json`
PEM parsing	Strips newlines, breaks `RS256`	`re.DOTALL`, keeps newlines
Token request	`jwt-bearer`, 3,600 s, storage and cloud scopes	Same
Objects uploaded	Theme `404.php` and `functions.php`	Theme `404.php` and Base64 `functions.php`
Probe paths	Theme `404.php?pax_cmd=id`, `/wp-content/404.php?pax_cmd=id`, `/pax_cmd.php?pax_cmd=id`	Theme `404.php?pax_cmd=id`, `/?pax_cmd=id`

These are scripts, not logs of a result. Nothing we saw shows whether the upload or the probe worked against the staging site.

CHAPTER 04

Behinder Twins and Workspace Utilities

IN SHORT

The fix-and-retry habit also shows up in PHP webshells. A PHP 7 webshell that breaks on PHP 8 sits next to a rebuilt PHP 8 version, both keyed with Behinder's public default.

INSIDE THIS CHAPTER

Two webshells, one repair 14
A PHP crash-bug calibration script 14
Utilities and agent folders 14

04 Behinder Twins and Workspace Utilities

Two webshells, one repair

A separate open directory holds `/serve/shell.php` (569 bytes, 25 lines) and `/serve/shell2.php` (755 bytes, 35 lines). Both speak the Behinder v3 protocol. Behinder is a public Chinese webshell manager that encrypts its traffic.

Each file reads an HTTP POST body, Base64-decodes it and decrypts it with OpenSSL `AES128`. If the OpenSSL extension is missing, it falls back to a 16-byte rolling XOR. Both derive their 16-byte key from Behinder v3's public default: the first 16 hex characters of `MD5('rebeyond')`.

The difference is how they run the decrypted code. Version 1 passes it to `create_function`, which PHP deprecated in 7.2 and removed in 8.0. On a modern stack it fails with a fatal error. Version 2 replaces it with `ob_start` and `eval`, and encrypts the response the same way.

File	Size	Lines	How it runs code	PHP versions
`/serve/shell.php`	569 bytes	25	`create_function`	PHP 7 only
`/serve/shell2.php`	755 bytes	35	`ob_start` and `eval`, encrypted output	PHP 8.x

Because both keep the public default key, Behinder traffic signatures and the file hashes in Chapter 9 both apply.

A PHP crash-bug calibration script

Another host holds two copies of `/CVE-2026-109618/tools/calibrate.py` (3,417 bytes each). CVE-2026-109618 is a critical PHP flaw. PHP mishandles uninitialized memory while cleaning up after an out-of-memory error, which allows remote code execution. The PHP project fixed it on 21 October 2026 in 8.2.35, 8.3.36, 8.4.27 and 8.5.12. PHP 8.1 and 7.x are end-of-life and get no fix.

A calibration script of this kind tunes memory layout before an exploit attempt, so it points at web-facing PHP targets. The script's owner is not established, so its hash is withheld.

Utilities and agent folders

A fourth host holds a key-generation script (1,421 bytes) and a rule-generation script (28,367 bytes). Three more hosts expose Hermes agent folders, an Orca terminal shell hook, a QQ chat-bot script and two Python bot modules. These may belong to independent developers running AI coding agents on badly configured servers. We keep them on a watchlist and withhold their hashes and addresses.

05

CHAPTER 05

Archived Crypto-Exchange Claims and the Operator Model

IN SHORT

The archive that held the playbook also holds swarm logs for 76 anonymized scopes. Read them as a threat model for database, NAS and cloud IAM segmentation, not as proof of a breach.

INSIDE THIS CHAPTER

What the crypto-exchange task claims 16

The operator model 17

05 Archived Crypto-Exchange Claims and the Operator Model

Autonomous red-team frameworks such as ARTEX save worker transcripts, bucket listings and LLM-written test reports to disk. When an operator leaves that workspace on open HTTP, defenders can read how a swarm chains reconnaissance into cloud and database pivots.



Figure 4: Iterative script repairs recur across cloud and webshell workspaces

What the crypto-exchange task claims

The crypto-exchange task records 2,178 worker runs against a Southeast Asian cryptocurrency exchange. No names, hostnames or bucket identifiers are disclosed. LLM-driven attack tools often overstate impact or log commands that never ran. So every item below is an unverified claim from the attacker's own records.

Claim in the archive	What it says
Linked-server SQL injection	A web or API SQL injection path was used to enable `xp_cmdshell` on a linked Microsoft SQL Server holding a production exchange database
NAS backup webshell	An unauthenticated PHP webshell was placed on a NAS backup mount reachable from the database network
Workstation cloud key	A Google Cloud project-editor service-account key was taken from an internal workstation, with further attempts to mint keys for a pentest account and reach Identity-Aware Proxy (IAP) dashboards
Cached access and bucket lists	Cached VPN certificates, SSH key pairs and two exchange bucket names; sibling tasks list banking and fintech buckets

All bucket names in these records belong to targets and are withheld.

The operator model

We model the wider infrastructure as one collector-operator cluster. This is an analytical model, not a proven actor. It rests on concrete links: shared endpoint hashes, reused secrets and certificates, and exposed task queues.

The model has 731 nodes and 795 edges. Of those edges, 76 are strong links such as a shared hash, secret or certificate. Three habits support the single-operator reading.

1. **Template and key reuse.** The two webshells reuse Behinder's public key. Elsewhere in the model, three hosts on Tencent cloud share a byte-identical `shell.aspx` and 16 secrets. A pair of controllers on LUCIDACLOUD hosting shares 714 endpoint hashes.
2. **Sector focus.** The iGaming playbook fits four Russian-language casino and compliance Telegram bots and four Telegram control groups found in public sources.
3. **Chronic exposure.** Leaving the playbook and the crypto-exchange logs in an open directory matches 634 exposed Python HTTP servers and 112 exposed AI agent-session hosts elsewhere in the model.

Model metric	Count	What it means
Nodes and edges	731 / 795	Includes 76 strong cross-host links
Documented access surfaces	271	Drop sites, bot and config credentials, templates, panels and databases
Exposed Python HTTP servers	634	The same open-directory pattern as the archive host
Shared endpoint hashes, LUCIDACLOUD pair	714	Dense overlap between two paired controllers
Shared secrets, Tencent trio	16	Alongside a byte-identical `shell.aspx` on three hosts
Exposed AI agent-session hosts	112	Hosts that leak agent transcripts or workspace folders
Telegram artifacts from public sources	4 + 4	Four casino and compliance bots and four control groups

CHAPTER 06

Who the Swarm Targeted, and Other Explanations

IN SHORT

Payments, fintech and Web3 firms make up more than half of the 76 scopes in the archive. The swarm's own severity labels are unverified, and other readings of the archive remain possible.

INSIDE THIS CHAPTER

Four ways to read the evidence 19

06 Who the Swarm Targeted, and Other Explanations

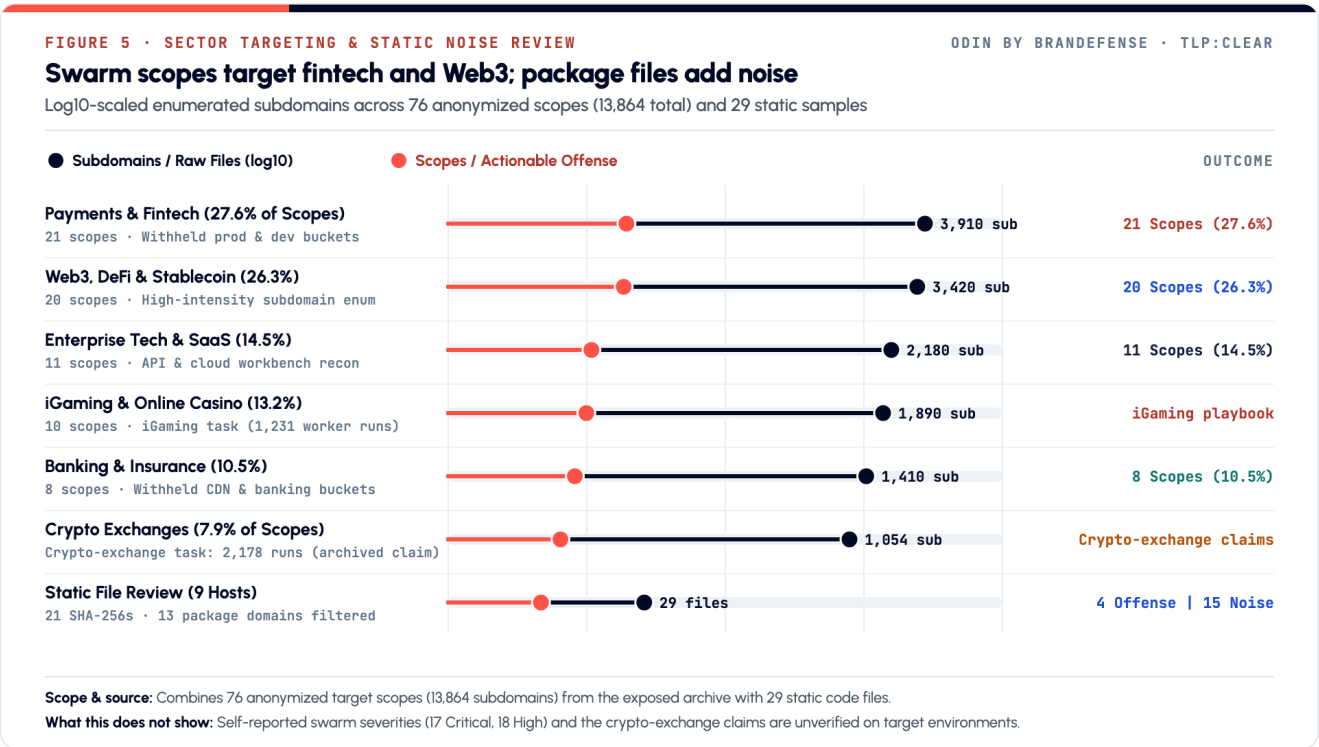


Figure 5: Swarm scopes target fintech and Web3; package files add noise

The archive lists 76 anonymized target scopes and 13,864 enumerated subdomains. Of those, 1,105 were live HTTP or TLS hosts. All target names are withheld.

Sector	Scopes	Share	Subdomains	Key task or bucket evidence
Payments and Fintech	21	27.6%	3,910	Production and development buckets (names withheld)
Web3, DeFi and Stablecoin	20	26.3%	3,420	Multi-worker subdomain and API reconnaissance
Enterprise Tech and SaaS	11	14.5%	2,180	Cloud workbench and SaaS tenant enumeration
iGaming and Online Casino	10	13.2%	1,890	The iGaming task (1,231 runs) and the bucket playbook
Banking and Insurance	8	10.5%	1,410	Banking and CDN buckets (names withheld)
Crypto Exchanges	6	7.9%	1,054	The crypto-exchange task (2,178 runs) and backup bucket claims

Four ways to read the evidence

1. **Package noise.** Raw string matching over `site-packages` and UI bundles produces false domains. The 15 package files are not indicators, and none of their 13 domain-like strings belong on a block list.
2. **An authorized test gone wrong.** Task-numbered folders, structured `penetration_test_report.md` outputs and a pentest-named service account in the crypto-exchange task fit a contracted red team misconfiguring an ARTEX server. Even so, an unauthenticated `pax_cmd` gate in a live theme file exposes the site to anyone, and needs urgent remediation.

3. **Developers, not operators.** Hosts with Hermes, Orca or bot folders may belong to developers running local AI agents. They stay on a watchlist, and their addresses are not published.
4. **Inflated LLM severity.** The archive's LLM summaries rate 22.4% of scopes critical (3 claimed remote code execution, 14 claimed data or key exposure) and 23.7% high, 46.1% combined. Only the iGaming task comes with recovered deployment code.

07

CHAPTER 07

What Defenders Should Do

IN SHORT

This chapter lists the actions defenders should take, in priority order.

07 What Defenders Should Do

Cloud and WebOps

- [ ] List every Cloud Storage, S3 or R2 bucket synced or mounted into a web root (`wp-content/themes/`, `wp-content/uploads/`). Turn off PHP execution on media mounts and turn on `DATA_WRITE` audit logs for those buckets.
- [ ] Allow theme-folder writes only from named CI/CD service accounts. Alert on any `application/x-php` object written by anyone else.

Cloud IAM

- [ ] Revoke long-lived user-managed service-account keys on workstations. Enforce `constraints/iam.disableServiceAccountKeyCreation`.
- [ ] Remove `roles/editor` from CI and developer accounts. Review OAuth `jwt-bearer` token requests from hosts outside Google Cloud.

CMS and platform

- [ ] Record hashes of the deployed `404.php` and `functions.php` for every theme, and alert on drift. Mount core and theme code read-only on origin containers.
- [ ] Set `disable_functions = system,passthru,exec,shell_exec` in `php.ini` where the site allows it.
- [ ] Upgrade every internet-facing and staging PHP runtime to 8.2.35, 8.3.36, 8.4.27 or 8.5.12 or later for CVE-2026-109618. Retire PHP 7.x, 8.0 and 8.1 pools.

Detection and network

- [ ] Block `pax_cmd` in query strings and POST bodies at the WAF, and deploy the rules in Chapter 8 on staging and production tiers.
- [ ] Check MSSQL linked servers for `xp_cmdshell`, and separate NAS backup mounts from database and web networks. These are the paths the crypto-exchange task claims.

CHAPTER 08

Detection Rules

IN SHORT

This chapter describes each published rule and where to deploy it.

INSIDE THIS CHAPTER

How the Suricata signatures work 24
How the other rules work 25
MITRE ATT&CK mapping 25

08 Detection Rules

The full rules are in Appendix B. Every rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma, Kusto.Language); the rules were not replayed against live traffic. Test them in your own environment before production use.

Rule file	What it catches
`odin_pax_cmd_gcs_wordpress_backdoor.rules`	Suricata 7: eight signatures for `pax_cmd` gates, direct theme template requests, PHP uploads to Cloud Storage, `jwt-bearer` token requests, the deployer scripts, the twin webshells and exposed workspace listings
`sentinel_gcs_theme_overwrite_and_pax_cmd_hunt.kql`	Microsoft Sentinel and Defender XDR: PHP workers starting shells, deployer command lines, and non-CI calls to Google OAuth and Storage APIs
`sigma_gcs_sa_key_abuse_and_cms_webshell_deploy.yml`	Sigma (3 rules): PHP theme writes in Cloud Storage audit logs, PHP workers spawning shells, and deployer or probe command lines on Linux
`odin_pax_cmd_deployer_and_behinder_twin_artifacts.yar`	YARA (4 rules): the deployer scripts, injected `pax_cmd` theme files, the Behinder twins and workspace files naming the stolen key or the CVE calibrator

How the Suricata signatures work

SID	Looks for	Why
9530001	`pax_cmd=` in an inbound request URI	Catches use of the injected gate through a query string (block candidate)
9530002	`pax_cmd=` in an inbound POST body	Catches the same gate through a form body (block candidate)
9530003	A direct request for any theme's `404.php` or `functions.php`	Normal visitors never request these files by path
9530004	Outbound `POST /upload/storage/v1/b/` with `uploadType=media`, a `wp-content` path and `Content-Type: application/x-php`	Catches a PHP theme write to Cloud Storage (needs TLS inspection)
9530005	Outbound `POST` to `oauth2.googleapis[.]com` with a `jwt-bearer` grant and an assertion	Hunt only: flags token requests from hosts where cloud SDKs should not run
9530006	Outbound `GET` for `deploy_gcs_webshell.py` or `deploy_gcs_webshell2.py`	Catches retrieval of the playbook
9530007	Inbound request for `/serve/shell.php` or `/serve/shell2.php`	Catches access to the twin webshell paths
9530008	A directory listing that names the deployer, the key file or `CVE-2026-109618`	Catches an exposed workspace of this kind on your own network

Use SIDs 9530001 and 9530002 in inline block mode once approved CMS proxies have `pass` rules. Keep SID 9530005 as a hunt signal on workstations and web tiers.

How the other rules work

The KQL query looks back 7 days. It joins two views of each device. The first finds a PHP or web worker (`php-fpm`, `apache2`, `httpd`, `w3wp.exe` and similar) starting `id`, `whoami`, `uname`, a shell, `curl` or `wget`. It also finds command lines naming the deployer, the key file, `pax_cmd=` or `uploadType=media`. The second finds Python, curl or PHP processes calling Google OAuth or Storage APIs.

Each match gets one of three labels: isolate a cloud key deployer, isolate a `pax_cmd` webshell, or hunt an unapproved API call. Exact runner hostnames go in two named allowlists. There are no wildcards.

The first Sigma rule reads Google Cloud audit logs. It flags `storage.objects.create` or `update` on PHP files under `wp-content/themes/` or `wp-content/plugins/`, or on `404.php`, `functions.php`, `pax_cmd.php` or the `/serve/` webshell names. Two exact deployment service accounts are excluded. The second Sigma rule flags PHP or web workers that spawn shells or recon tools. The third flags Python or curl command lines that name the deployer, the key file, `pax_cmd=id` or the CVE calibrator.

YARA rule	Fires when
`ODIN_GCS_WordPress_PaxCmd_Deployer_Playbook`	A Google OAuth token marker appears with `devstorage.full_control`, `uploadType=media` and either a theme path or `pax_cmd`
`ODIN_WordPress_PaxCmd_Injected_Theme_Gate`	`pax_cmd` and `_REQUEST` appear with two of `system`, `passthru` and `shell_exec`
`ODIN_Behinder_AES128_XOR_Twin_PHP_Webshell`	`AES128`, `openssl_decrypt` and the XOR index marker appear with `create_function` or `ob_start`
`ODIN_GCS_SA_Workspace_and_Probe_Artifact`	Two of the key file name, the deployer name and `application/x-php` appear, or the CVE name appears with `tools/calibrate.py`

MITRE ATT&CK mapping

Tactic	Technique	How the chain uses it	Detection
Defense Evasion, Persistence	T1078.004 Valid Accounts: Cloud Accounts	A stolen service-account key signs in to Google Cloud	SID 9530005; KQL; Sigma rule 1
Credential Access	T1528 Steal Application Access Token	The key is traded for a one-hour bearer token with full storage rights	SID 9530005; Sigma rule 3
Persistence	T1505.003 Server Software Component: Web Shell	`pax_cmd` theme gate and the Behinder twin webshells	SIDs 9530001 to 9530004 and 9530007; Sigma rules 1 and 2; YARA

09

CHAPTER 09

Indicators

IN SHORT

This chapter lists the published indicators and explains what stays withheld.

INSIDE THIS CHAPTER

File hashes 27

Other indicators 27

09 Indicators

Appendix A holds 4 published indicators: 4 file hashes (SHA-256) and no addresses. Eight more indicators appear in this report only: the `pax_cmd` parameter, three probe URIs, three cloud API patterns and the stolen key's task path.

Type	Count	Where to use it
Network addresses	0	All host addresses are withheld
File hashes (SHA-256)	4	EDR and file-integrity scanning
Web request parameter and probe URIs	4	WAF rules and web access logs
Cloud API patterns	3	Cloud audit logs and TLS egress proxies
Workspace file path	1	Endpoint command-line and file hunting

File hashes

File	Size (bytes)	SHA-256	Use
`deploy_gcs_webshell.py` (version 1)	5,251	`20745360f904db566a373ce39bb05ea52fee24c5f6a404db756e9c5aecb1981e`	Hunt on workstations and servers
`deploy_gcs_webshell2.py` (version 2)	5,480	`17c96f2ab239b6fd5b5454aae51301bee2021f0b8d9726c567c48b8ab167fbdc`	Hunt on workstations and servers
`/serve/shell.php`	569	`d0c6e2776390e00684969931b003047b5f5b7c483e1cf263320d8181bbab94fc`	Block in web roots
`/serve/shell2.php`	755	`67e9658391a17bc1e8dd23250bd5b99d4ef5c2e0f63d08f3c3a8abe61925c8bc`	Block in web roots

Other indicators

Indicator	Type	Use
`pax_cmd`	Query or POST parameter	Block at the WAF
`/wp-content/404.php?pax_cmd=id`	Probe URI (version 1 fallback)	Block at the WAF
`/pax_cmd.php?pax_cmd=id`	Probe URI (version 1 root fallback)	Block at the WAF
`/?pax_cmd=id`	Probe URI (version 2 `functions.php` check)	Block at the WAF
`oauth2.googleapis[.]com/token` with grant type `jwt-bearer`	Cloud API pattern	Watch, do not block the domain
`devstorage.full_control` scope	OAuth scope	Watch in IAM logs
`storage.googleapis[.]com/upload/storage/v1/b/*/o?uploadType=media`	Cloud API pattern	Hunt for PHP writes
`i21899/gcs_service_account.json`	Workspace file path	Hunt on endpoints

The Google domains are legitimate services. Never block them at the firewall. Look for unusual callers instead.

Withheld: all nine host addresses, every bucket name, the victim staging site, the theme name in the overwritten paths, and the hashes of the nine utility, agent and bot files whose owners are not established.

10

CHAPTER 10

Scope and Limits

IN SHORT

This chapter sets out what the evidence can show and what it cannot.

10 Scope and Limits

- **Window.** The findings cover a two-week observation window ending 18 November 2026.
- **Code, not outcomes.** The playbook and webshells were read as text. Nothing we saw confirms that the bucket upload, the theme overwrite or the `pax_cmd` probe worked on the staging site.
- **The archive speaks for the attacker.** Worker counts, the crypto-exchange claims and severity labels come from the swarm's own logs and LLM reports. We confirmed none of them on a target system.
- **One operator is a model.** The 731-node cluster is an analytical model built on shared hashes, secrets and certificates. It is not a named actor.
- **An authorized test is possible.** Some archive features fit a contracted red team. The unauthenticated backdoor still puts the victim site at risk.
- **Users are not operators.** Hosts with only agent, bot or package files are treated as users, and their files are not published as indicators.
- **Rule testing.** Every published rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma and Microsoft's Kusto.Language analyzer, checked on 20 November 2026). The rules were not replayed against live traffic, so test and tune them in your own environment before production use.

11

CHAPTER 11

Sources

IN SHORT

This chapter lists the public references used in the report.

11 Sources

1. Google Cloud. Best practices for managing service account keys. https://cloud.google.com/iam/docs/best-practices-for-managing-service-account-keys (verified 20 November 2026)
2. Google Cloud Storage. JSON API v1, Objects: insert. https://cloud.google.com/storage/docs/json_api/v1/objects/insert (verified 20 November 2026)
3. Google Identity. Using OAuth 2.0 for server to server applications. https://developers.google.com/identity/protocols/oauth2/service-account (verified 20 November 2026)
4. WordPress Developer Resources. Theme Handbook: Template Hierarchy. https://developer.wordpress.org/themes/basics/template-hierarchy/ (verified 20 November 2026)
5. MITRE ATT&CK. T1078.004 Valid Accounts: Cloud Accounts. https://attack.mitre.org/techniques/T1078/004/ (verified 20 November 2026)
6. MITRE ATT&CK. T1505.003 Server Software Component: Web Shell. https://attack.mitre.org/techniques/T1505/003/ (verified 20 November 2026)
7. MITRE ATT&CK. T1528 Steal Application Access Token. https://attack.mitre.org/techniques/T1528/ (verified 20 November 2026)
8. OWASP Cheat Sheet Series. File Upload Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html (verified 20 November 2026)

APPENDICES

Indicators and Detection Rules

FOR DEFENDERS

Appendix A lists the indicators published for this study. Appendix B carries every published detection rule in full.

INSIDE

A. Indicators of compromise (4) 34

B. Detection rules (4 files) 35

A Indicators of Compromise

Values are defanged for display. The CSV, JSON and STIX files on the online edition carry the raw values. Values the report marks as withheld are not listed.

ID	Type	Indicator	Port	Context	Action
R30-IOC-01	SHA-256	20745360f904db566a373ce39bb05ea52fee24c5f6a404db756e9c5aecb1981e		deploy_gcs_webshell.py · GCS-to-WordPress pax_cmd deployer, iteration 1	HUNT
R30-IOC-02	SHA-256	17c96f2ab239b6fd5b5454aae51301bee2021f0b8d9726c567c48b8ab167fbdc		deploy_gcs_webshell2.py · GCS-to-WordPress pax_cmd deployer, iteration 2	HUNT
R30-IOC-03	SHA-256	d0c6e2776390e00684969931b003047b5f5b7c483e1cf263320d8181bbab94fc		shell.php · Behinder-compatible AES-128/XOR web shell, version 1 for PHP 7 (569 B)	BLOCK
R30-IOC-04	SHA-256	67e9658391a17bc1e8dd23250bd5b99d4ef5c2e0f63d08f3c3a8abe61925c8bc		shell2.php · Behinder-compatible AES-128/XOR web shell, version 2 for PHP 8 (755 B)	BLOCK

B Detection Rules

Each rule file loads in its own engine (YARA-X, Suricata 7, Kusto, pySigma). Tune thresholds and allow-lists to your environment before alerting or blocking. The same files can be downloaded from the online edition.

odin_pax_cmd_deployer_and_behinder_twin_artifacts.yar

YARA

```
/*
   Odin by Brandefense - Monograph 30 / 30 YARA Artifact Suite
   Scope: Structural detection rules for Python GCS-to-WordPress pax_cmd deployers,
          injected WordPress theme templates (404.php / functions.php), Behinder-compatible
          AES128/XOR PHP 7/8 twin webshells, and stolen-key / PHP OOM workspace manifests.
   Classification: TLP:CLEAR
*/

rule ODIN_GCS_WordPress_PaxCmd_Deployer_Playbook
{
    meta:
        author = "Odin by Brandefense"
        monograph = "30 / 30"
        classification = "TLP:CLEAR"
        disposition = "HUNT_PRIMARY_OFFENSE_PLAYBOOK"
        description = "Matches Python GCS service-account JWT minting and WordPress pax_cmd theme overwrite
deployers"
    strings:
        $oa1 = "oauth2.googleapis.com/token" ascii
        $oa2 = "urn:ietf:params:oauth:grant-type:jwt-bearer" ascii
        $sc1 = "devstorage.full_control" ascii
        $up1 = "uploadType=media" ascii
        $wp1 = "wp-content/themes/" ascii
        $px1 = "pax_cmd" ascii
    condition:
        filesize < 128KB and ($oa1 or $oa2) and $sc1 and $up1 and ($wp1 or $px1)
}

rule ODIN_WordPress_PaxCmd_Injected_Theme_Gate
{
    meta:
        author = "Odin by Brandefense"
        monograph = "30 / 30"
        classification = "TLP:CLEAR"
        disposition = "BLOCK_INJECTED_CMS_TEMPLATE"
        description = "Matches WordPress theme files containing the pax_cmd parameter gate and multi-function
execution fallback identifiers"
    strings:
        $gate = "pax_cmd" ascii
        $req = "_REQUEST" ascii
        $fn1 = "passthru" ascii
        $fn2 = "shell_exec" ascii
        $fn3 = "system" ascii
    condition:
        filesize < 256KB and $gate and $req and 2 of ($fn*)
}

rule ODIN_Behinder_AES128_XOR_Twin_PHP_Webshell
{
    meta:
        author = "Odin by Brandefense"
        monograph = "30 / 30"
        classification = "TLP:CLEAR"
        disposition = "BLOCK_ENCRYPTED_PHP_WEBSHELL"
        description = "Matches Behinder-compatible PHP 7 create_function and PHP 8 ob_start/eval encrypted twin
webshells"
    strings:
        $c1 = "AES128" ascii
        $c2 = "openssl_decrypt" ascii
        $x1 = "15]" ascii
```

```
        $e1 = "create_function" ascii
        $e2 = "ob_start" ascii
    condition:
        filesize < 64KB and $c1 and $c2 and $x1 and ($e1 or $e2)
}

rule ODIN_GCS_SA_Workspace_and_Probe_Artifact
{
    meta:
        author = "Odin by Brandefense"
        monograph = "30 / 30"
        classification = "TLP:CLEAR"
        disposition = "WATCHLIST_OPERATOR_WORKSPACE"
        description = "Matches operator workspace artifacts referencing stolen GCS service-account paths or
CVE-2026-109618 calibration tools"
    strings:
        $w1 = "gcs_service_account.json" ascii
        $w2 = "deploy_gcs_webshell" ascii
        $w3 = "application/x-php" ascii
        $c1 = "CVE-2026-109618" ascii
        $c2 = "tools/calibrate.py" ascii
    condition:
        filesize < 512KB and ((2 of ($w*)) or ($c1 and $c2))
}
```

odin_pax_cmd_gcs_wordpress_backdoor.rules

SURICATA

```
# Odin by Brandefense - Monograph 30 / 30 Suricata 7.x Detection Suite
# Coverage: WordPress pax_cmd request gates, direct theme template probes, GCS uploadType=media PHP writes,
#           OAuth2 jwt-bearer exchanges, deploy_gcs_webshell playbooks, and Behinder /serve/shell*.php twins
# Classification: TLP:CLEAR

alert http any any -> $HOME_NET any (msg:"ODIN [R30] WordPress Backdoor pax_cmd Query Parameter Execution Gate";
flow:established,to_server; http.uri; bsize:>8; content:"pax_cmd="; fast_pattern; classtype:web-application-
attack; sid:9530001; rev:1; metadata:created_at 2026_11_20, deployment perimeter_and_waf, tactic execution;)

alert http any any -> $HOME_NET any (msg:"ODIN [R30] WordPress Backdoor pax_cmd HTTP POST Body Parameter Gate";
flow:established,to_server; http.method; content:"POST"; http.request_body; bsize:>8; content:"pax_cmd=";
fast_pattern; classtype:web-application-attack; sid:9530002; rev:1; metadata:created_at 2026_11_20, deployment
perimeter_and_waf, tactic execution;)

alert http any any -> $HOME_NET any (msg:"ODIN [R30] WordPress Theme 404.php or functions.php Direct Request";
flow:established,to_server; http.uri; content:"/wp-content/themes/"; fast_pattern; pcre:"/\/wp-content\/themes\/
[^\/]+\/(404|functions)\.php/i"; classtype:web-application-attack; sid:9530003; rev:2; metadata:created_at
2026_11_20, deployment perimeter_and_waf, tactic persistence;)

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN [R30] GCS JSON API uploadType=media PHP Upload to
WordPress Path"; flow:established,to_server; http.method; content:"POST"; http.uri; bsize:>32; content:"/upload/
storage/v1/b/"; content:"uploadType=media"; fast_pattern; content:"wp-content"; http.content_type; bsize:17;
content:"application/x-php"; classtype:web-application-attack; sid:9530004; rev:1; metadata:created_at
2026_11_20, deployment tls_egress_proxy, tactic persistence;)

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN [R30] HUNT Non-Browser OAuth2 jwt-bearer Service-Account
Token Mint"; flow:established,to_server; http.method; content:"POST"; http.host; bsize:21;
content:"oauth2.googleapis.com"; http.request_body; bsize:>48; content:"urn%3Aietf%3Aparams%3Aoauth%3Agrant-
type%3Ajwt-bearer"; fast_pattern; content:"assertion="; classtype:policy-violation; sid:9530005; rev:2;
metadata:created_at 2026_11_20, deployment tls_egress_proxy, policy behavioral_hunt;)

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN [R30] Open-Directory GCS-to-WordPress Playbook Script
Retrieval"; flow:established,to_server; http.method; content:"GET"; http.uri; bsize:>22;
content:"deploy_gcs_webshell"; fast_pattern; pcre:"/\/deploy_gcs_webshell2?\.py$/"; classtype:trojan-activity;
sid:9530006; rev:1; metadata:created_at 2026_11_20, deployment perimeter, tactic resource_development;)

alert http any any -> $HOME_NET any (msg:"ODIN [R30] Behinder-Compatible Twin PHP Webshell Access (/serve/
shell.php)"; flow:established,to_server; http.uri; bsize:>15; content:"/serve/shell"; fast_pattern; pcre:"/\/
serve\/shell2?\.php(\?|$)/i"; classtype:web-application-attack; sid:9530007; rev:1; metadata:created_at
2026_11_20, deployment perimeter_and_waf, tactic command_and_control;)
```

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN [R30] Open-Directory Offense Workspace Listing (GCS Key
or PHP OOM Calibrator)"; flow:established,from_server; http.stat_code; content:"200"; http.response_body;
bsize:>32; content:"Directory listing for /"; fast_pattern; pcre:"/(deploy_gcs_webshell|
gcs_service_account\.json|CVE-2026-109618)/"; classtype:attempted-recon; sid:9530008; rev:1; metadata:created_at
2026_11_20, deployment perimeter, tactic reconnaissance;)
```

sentinel_gcs_theme_overwrite_and_pax_cmd_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Monograph 30 / 30 Microsoft Sentinel & Defender XDR Hunt
// Correlates non-CI GCP OAuth2/Storage API calls, stolen service-account JSON reads,
// deploy_gcs_webshell execution, and PHP-FPM/web-worker child shell commands (pax_cmd)
// Classification: TLP:CLEAR

let Lookback = 7d;
let ApprovedCicdHosts = dynamic(["CMS-CICD-RUNNER-01", "GCP-TERRAFORM-RUNNER-01"]);
let ApprovedSecurityHosts = dynamic(["SEC-VALIDATION-LNX-01", "SEC-VALIDATION-WEB-01"]);
let CloudAuthAndStorageDomains = dynamic(["oauth2.googleapis.com", "storage.googleapis.com"]);
let WebWorkerBinaries = dynamic(["php-fpm", "php-cgi", "php", "apache2", "httpd", "lsphp", "w3wp.exe"]);
let SuspiciousChildBinaries = dynamic(["id", "whoami", "uname", "sh", "bash", "dash", "curl", "wget",
"python3"]);
let SuspiciousHostProcs = DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedCicdHosts)) and not(DeviceName in~ (ApprovedSecurityHosts))
    | where (InitiatingProcessFileName in~ (WebWorkerBinaries) and FileName in~ (SuspiciousChildBinaries))
        or ProcessCommandLine has_any ("deploy_gcs_webshell", "gcs_service_account.json", "pax_cmd=",
"uploadType=media")
    | project ProcTime = Timestamp, DeviceId, DeviceName, InitiatingProcessFileName, FileName,
ProcessCommandLine;
DeviceNetworkEvents
| where Timestamp >= ago(Lookback)
| where not(DeviceName in~ (ApprovedCicdHosts)) and not(DeviceName in~ (ApprovedSecurityHosts))
| where (RemoteUrl has_any (CloudAuthAndStorageDomains) and InitiatingProcessFileName in~ ("python", "python3",
"curl", "php", "php-fpm"))
    or RemoteUrl has_any ("pax_cmd=", "/serve/shell.php", "/serve/shell2.php")
| summarize
    NetFirstSeen = min(Timestamp),
    NetLastSeen = max(Timestamp),
    DistinctRemoteUrls = dcount(RemoteUrl),
    SampleRemoteUrls = make_set(RemoteUrl, 12),
    ConnectionCount = count()
    by DeviceId, DeviceName, InitiatingProcessFileName
| join kind=fullouter (SuspiciousHostProcs) on DeviceId, DeviceName
| extend ResolvedDevice = coalesce(DeviceName, DeviceName1)
| where isnotempty(ProcessCommandLine) or SampleRemoteUrls has_any ("pax_cmd=", "uploadType=media",
"oauth2.googleapis.com", "storage.googleapis.com")
| extend AlertDisposition = case(
    ProcessCommandLine has_any ("deploy_gcs_webshell", "gcs_service_account.json") and SampleRemoteUrls has_any
("oauth2.googleapis.com", "storage.googleapis.com"), "ISOLATE_CLOUD_SA_WEBSHELL_DEPLOYER",
    InitiatingProcessFileName1 in~ (WebWorkerBinaries) or ProcessCommandLine has "pax_cmd=",
"ISOLATE_CMS_PAX_CMD_WEBSHELL_EXECUTION",
    "HUNT_UNAPPROVED_GCS_API_OR_THEME_PROBE"
)
| project NetFirstSeen, NetLastSeen, ProcTime, ResolvedDevice, AlertDisposition, InitiatingProcessFileName,
InitiatingProcessFileName1, FileName, ProcessCommandLine, SampleRemoteUrls, ConnectionCount
| sort by coalesce(ProcTime, NetLastSeen) desc
```

sigma_gcs_sa_key_abuse_and_cms_webshell_deploy.yml

SIGMA

```
title: Google Cloud Storage PHP Webshell Upload to WordPress Theme or Web-Root Bucket Path
id: 9a41d8c2-5f19-4e30-b724-202611203001
status: experimental
description: Detects Google Cloud Storage object creation or overwrite calls writing PHP files or WordPress theme
templates (404.php, functions.php) via service accounts outside approved CI/CD pipelines.
author: Odin by Brandefense
```

```
  date: 2026/11/20
  tags:
    - attack.defense-evasion
    - attack.t1078.004
    - attack.persistence
    - attack.t1505.003
  logsource:
    product: gcp
    service: gcp.audit
  detection:
    selection_method:
      gcp.audit.method_name:
        - 'storage.objects.create'
        - 'storage.objects.update'
    selection_cms_php_object:
      gcp.audit.resource_name|re: '(wp-content/(themes|plugins)/.*\.php|/404\.php$|/functions\.php$|/
  pax_cmd\.php$|/serve/shell2?\.php$)'
    filter_cicd:
      gcp.audit.authentication_info.principal_email:
        - 'wp-cicd-deployer@prod-cms-sync.iam.gserviceaccount.com'
        - 'terraform-media-sync@prod-cms-sync.iam.gserviceaccount.com'
    condition: selection_method and selection_cms_php_object and not filter_cicd
  falsepositives:
    - Authorized CMS release pipelines that sync pre-built WordPress theme code into object storage buckets
  level: critical
  ---
  title: PHP-FPM or Web Server Worker Spawning System Reconnaissance or Shell Subprocess
  id: 3e82b7f1-9c04-4d62-a815-202611203002
  status: experimental
  description: Identifies PHP-FPM, Apache, or CGI workers spawning OS reconnaissance or shell binaries consistent
  with pax_cmd fallback execution (system, passthru, exec, shell_exec) or Behinder eval payloads.
  author: Odin by Brandefense
  date: 2026/11/20
  tags:
    - attack.persistence
    - attack.t1505.003
    - attack.execution
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_parent:
      ParentImage|endswith:
        - '/php-fpm'
        - '/php-fpm8.2'
        - '/php-fpm8.3'
        - '/php-cgi'
        - '/apache2'
        - '/httpd'
        - '/lsphp'
    selection_child:
      Image|endswith:
        - '/id'
        - '/whoami'
        - '/uname'
        - '/sh'
        - '/bash'
        - '/dash'
        - '/curl'
        - '/wget'
    filter_lab:
      ComputerName:
        - 'SEC-VALIDATION-WEB-01'
    condition: selection_parent and selection_child and not filter_lab
  falsepositives:
    - Legacy PHP administration plugins invoking system utilities on non-hardened web hosts
  level: critical
  ---
  title: GCS Service-Account Webshell Deployer or curl pax_cmd Verification Probe Execution
  id: 7c15f0a8-2b61-4a93-8e57-202611203003
  status: experimental
  description: Detects command-line execution of the deploy_gcs_webshell Python playbook, local
```

```
  gcs_service_account.json reads, or post-upload curl -sk verification probes carrying the pax_cmd parameter.
  author: Odin by Brandefense
  date: 2026/11/20
  tags:
    - attack.credential-access
    - attack.t1528
    - attack.persistence
    - attack.t1505.003
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_runtime:
      Image|endswith:
        - '/python'
        - '/python3'
        - '/curl'
    selection_cli:
      CommandLine|contains:
        - 'deploy_gcs_webshell'
        - 'gcs_service_account.json'
        - 'pax_cmd=id'
        - '/CVE-2026-109618/tools/calibrate.py'
    filter_security_host:
      ComputerName:
        - 'SEC-VALIDATION-LNX-01'
    condition: selection_runtime and selection_cli and not filter_security_host
  falsepositives:
    - Registered security validation hosts replaying authorized cloud-to-CMS purple-team scenarios
  level: high
```

ODIN

About Odin Research

Odin Research is the public research series of Odin by Brandefense. It documents adversary infrastructure seen on the internet and turns each finding into indicators and detection rules that defenders can use. Victims and third parties stay anonymous, addresses appear only for confirmed adversary infrastructure, and no working attack code is published.

Odin is the agentic analyst inside the Brandefense threat intelligence platform, which unifies Cyber Threat Intelligence, Digital Risk Protection and External Attack Surface Management.

Online edition, indicators and rules · brandefense.io/odin-research/reports/stolen-cloud-keys-backdoored-wordpress/
Short Read · brandefense.io/odin-research/blog/how-a-stolen-cloud-key-becomes-a-wordpress-backdoor/
Defense Kit · brandefense.io/odin-research/defense-kit/