ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 18 / 30 · TLS INTERCEPTION

Exposed Root CA Keys
Rogue Proxies, Yakit Suites and PowerSploit Staging

How a leaked mitmproxy signing key, proxy auto-config scripts, GUI attack workbenches and a PowerSploit tree fit one defensive picture

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

9 files
Root CA keys and certificates, a proxy auto-config script and two 5.94 MB logs on Server A (11.89 MB in total)

314
Files in 33 directories on Server B, including 159 PowerSploit files

2,992
Proxy, workbench and phishing endpoints on 2,170 unique IPv4 addresses, across five tool families

208
Confirmed or suspected mitmproxy interception listeners, out of 2,066 mitmproxy endpoints

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

How a leaked mitmproxy signing key, proxy auto-config scripts, GUI attack workbenches and a PowerSploit tree fit one defensive picture

01	Executive Summary	04
02	One Model, Four Roles	07
03	The Exposed Root CA on Server A	10
04	GUI Workbenches: Yakit, ScopeSentry, Nemo and GoPhish	13
05	Inside Server B: PowerSploit and Lure Staging	16
06	Fleet Comparison and How to Treat Each Finding	20
07	What Defenders Should Do	23
08	Detection Rules	25
09	Indicators	28
10	Scope and Limits	31
11	Sources	33
A	Indicators of compromise (9)	35
B	Detection rules (4 files)	36

IN SHORT

Odin by Brandefense found an open server exposing a mitmproxy root CA private key, a separate 314-file PowerSploit staging server, and 2,992 proxy, workbench and phishing endpoints. Nothing links these to one operator, so we group them in an analytical model, not a campaign. A leaked root key lets anyone sign certificates that every endpoint trusting that root will accept. Audit your root certificate stores for mitmproxy roots, and lock proxy auto-config settings to approved corporate hosts.

01

CHAPTER 01

Executive Summary

IN SHORT

Odin by Brandefense found an open server exposing a mitmproxy root CA private key, a separate 314-file PowerSploit staging server, and 2,992 proxy, workbench and phishing endpoints. Nothing links these to one operator, so we group them in an analytical model, not a campaign. A leaked root key lets anyone sign certificates that every endpoint trusting that root will accept. Audit your root certificate stores for mitmproxy roots, and lock proxy auto-config settings to approved corporate hosts.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 05

01 Executive Summary

TLS interception tools such as mitmproxy are standard kit for developers and red teams. They become dangerous when their root certificate sits in an endpoint's trust store and the private key is no longer private. This report looks at two exposed servers and five tool fleets that cover reconnaissance, phishing, interception and post-exploitation staging.

1. **An open server exposed a mitmproxy root CA private key.**
   Server A, on Scaleway in France, served 9 files (11.89 MB). They include the mitmproxy root CA in PEM and PKCS#12 form with the private key, a 300-byte proxy auto-config script and two 5.94 MB logs. Its address and port are withheld so nobody can fetch the key.

2. **Five tool fleets cover recon, lures and TLS interception.**
   We counted 2,992 endpoints on 2,170 IPv4 addresses running mitmproxy, GoPhish, ScopeSentry, YakitSecurityTool and Nemo. Of 2,066 mitmproxy endpoints, 208 are confirmed or suspected interception listeners. Yakit has the densest high-signal share: 26 of 34 endpoints.

3. **A 314-file staging server pairs lures with PowerSploit loaders.**
   Server B, on DigitalOcean in the United States, holds 314 files in 33 directories. They include OS-specific redirect pages, phishing templates and a 159-file PowerSploit tree with reflective loaders and raw NTFS parsers. Folder names suggest an authorized assessment, and no use against a target is shown.

Three Decisions for Leaders

1. **Audit root certificate stores and block unapproved imports.** Compare endpoint root stores with your approved PKI thumbprints. Remove any mitmproxy root that nobody approved, and alert on command-line certificate imports.
2. **Lock proxy auto-config to approved gateways.** Set the browser `AutoConfigURL` value through Group Policy or MDM. Watch browser egress to ports 8080, 18080, 31280 and 32300 when a PAC or certificate signal backs it up.
3. **Treat a leaked key differently from an assessment workspace.** Use Server A's key material for root-store cleanup and historical hunting. Keep Server B and uncorroborated recon consoles on review watchlists, not automatic block lists.

The Study in Numbers

Number	What it is
9 files	Root CA keys and certificates, a proxy auto-config script and two 5.94 MB logs on Server A (11.89 MB in total)
314	Files in 33 directories on Server B, including 159 PowerSploit files
2,992	Proxy, workbench and phishing endpoints on 2,170 unique IPv4 addresses, across five tool families
208	Confirmed or suspected mitmproxy interception listeners, out of 2,066 mitmproxy endpoints
152	Actionable GoPhish lure servers, out of 766 GoPhish endpoints
26 of 34	Yakit (Yaklang) backends that are confirmed or suspected (76.5%)
9	Published indicators: Server B file SHA-256 hashes (6 host entries and 9 file hashes withheld)

How to use this report.

Security leaders can read Chapter 1 and Chapter 7. PKI and endpoint teams should read Chapters 3 and 5, then the checklist in Chapter 7. Detection engineers should start with Chapters 8 and 9.

02

CHAPTER 02

One Model, Four Roles

IN SHORT

The two servers and five fleets are not linked to one actor. We use a four-role analytical model to show how these capabilities could chain, and where a defender can break the chain.

INSIDE THIS CHAPTER

What the model is and is not 07

What we looked at 08

The four roles 08

02 One Model, Four Roles

What the model is and is not

No shared credential, TLS certificate or network flow links Server A, Server B and the five fleets. The four roles below are an analytical model. They describe capabilities, not an observed campaign. Lure delivery, trust-store installation and session interception are all unverified.

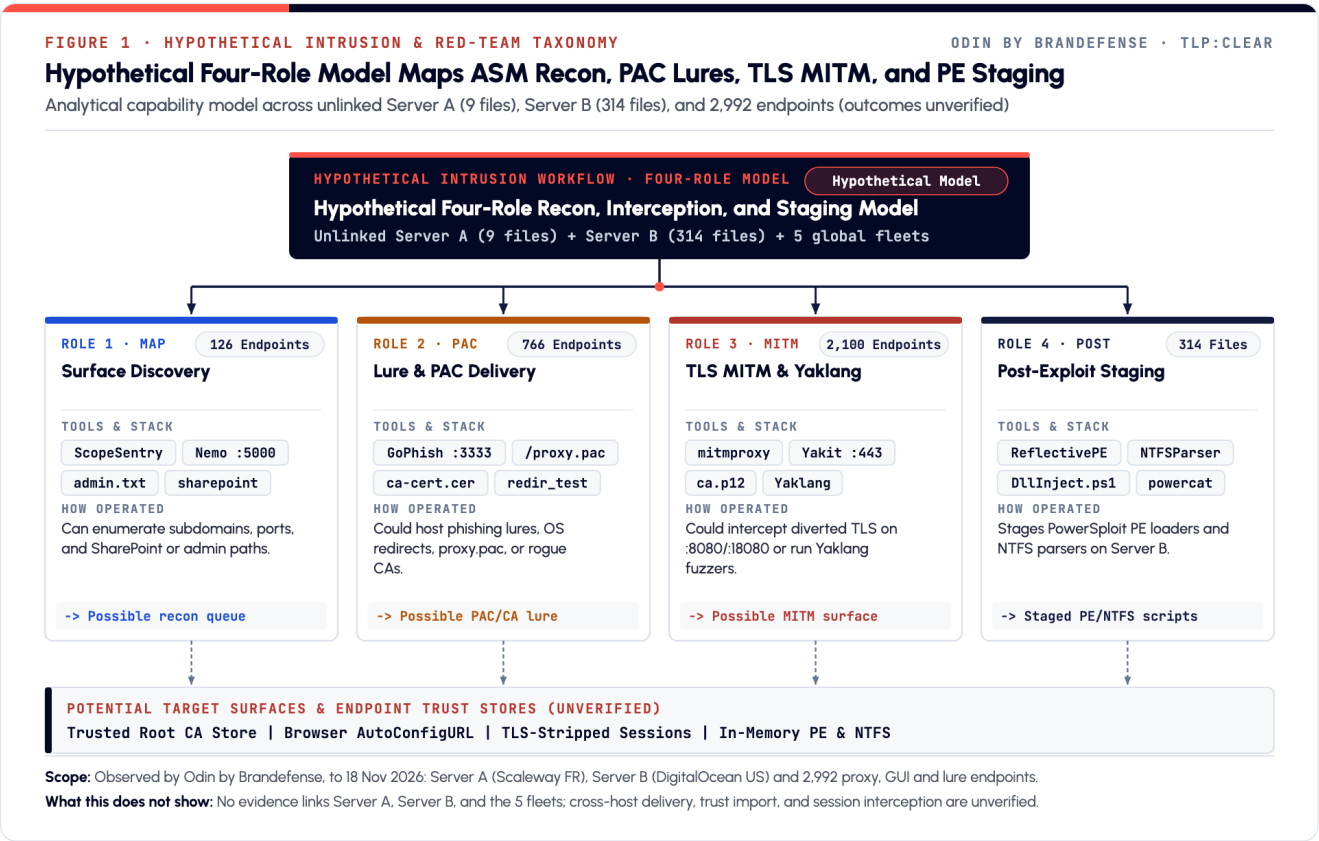


Figure 1: Hypothetical four-role model maps ASM recon, PAC lures, TLS MITM and PE staging

What we looked at

Cluster	Hosting	Size	How to treat it	Role in the model
Server A: exposed root CA	Scaleway, France	9 files	Block list for root-store cleanup, after a fresh ownership check	mitmproxy root keys, `proxy.pac` and two logs
Server B: staged engagement tree	DigitalOcean, United States	314 files	Watchlist and hunt	159 PowerSploit files, `powercat.ps1`, OS redirect pages and lures
mitmproxy fleet	1,345 IPs	2,066 endpoints	Hunt and watch	Programmable TLS interception proxies, led by 32300 (505), 8080 (192), 8081 (125) and 18080 (25)
GoPhish fleet	672 IPs	766 endpoints	Hunt and watch	Phishing panels, led by 3333 (417) and 443 (127)
ScopeSentry fleet	118 IPs	120 endpoints	Watchlist	Attack surface consoles, led by 8082 (101)
Yakit (Yaklang) fleet	31 IPs	34 endpoints	Hunt and watch	MITM, fuzzing, PoC scanning and reverse-listener backends on 443 and 8087
Nemo fleet	6 IPs	6 endpoints, all confirmed	Watchlist	Automated asset discovery consoles on 5000

The four roles

Role	Capability	What we saw	What is not shown
1. Surface discovery	Enumerate subdomains, ports and admin paths	ScopeSentry (120 endpoints, 101 on 8082), Nemo (6 on 5000), and `admin.txt` and `sharepoint.txt` wordlists on Server B	Any link from these consoles to Server B
2. Lure and PAC delivery	Host lures that ask a user to import a root CA or set an external `proxy.pac` URL	GoPhish panels on 3333; OS-specific redirect pages on Server B; `proxy.pac` and the public CA certificate on Server A	Any delivery from these servers. No `.pac` or `.cer` page was seen on GoPhish endpoints
3. TLS interception	Divert browser traffic and sign leaf certificates on the fly	mitmproxy listeners on 8080 and 18080; Yakit on 443 and 8087; the exposed root key on Server A	Any intercepted session
4. Post-exploitation staging	Hold in-memory loaders and relays for download	Reflective PE and DLL loaders, NTFS parsers and `powercat.ps1` on Server B	Any execution on a target

CHAPTER 03

The Exposed Root CA on Server A

IN SHORT

Server A exposed a full mitmproxy root CA, with the private key, next to a proxy auto-config script and two large logs. Anyone holding that key can forge certificates for any domain on endpoints that trust the root.

INSIDE THIS CHAPTER

What the server held 10
Why a leaked root key matters 11
How a PAC file steers only the traffic that matters 11

03 The Exposed Root CA on Server A

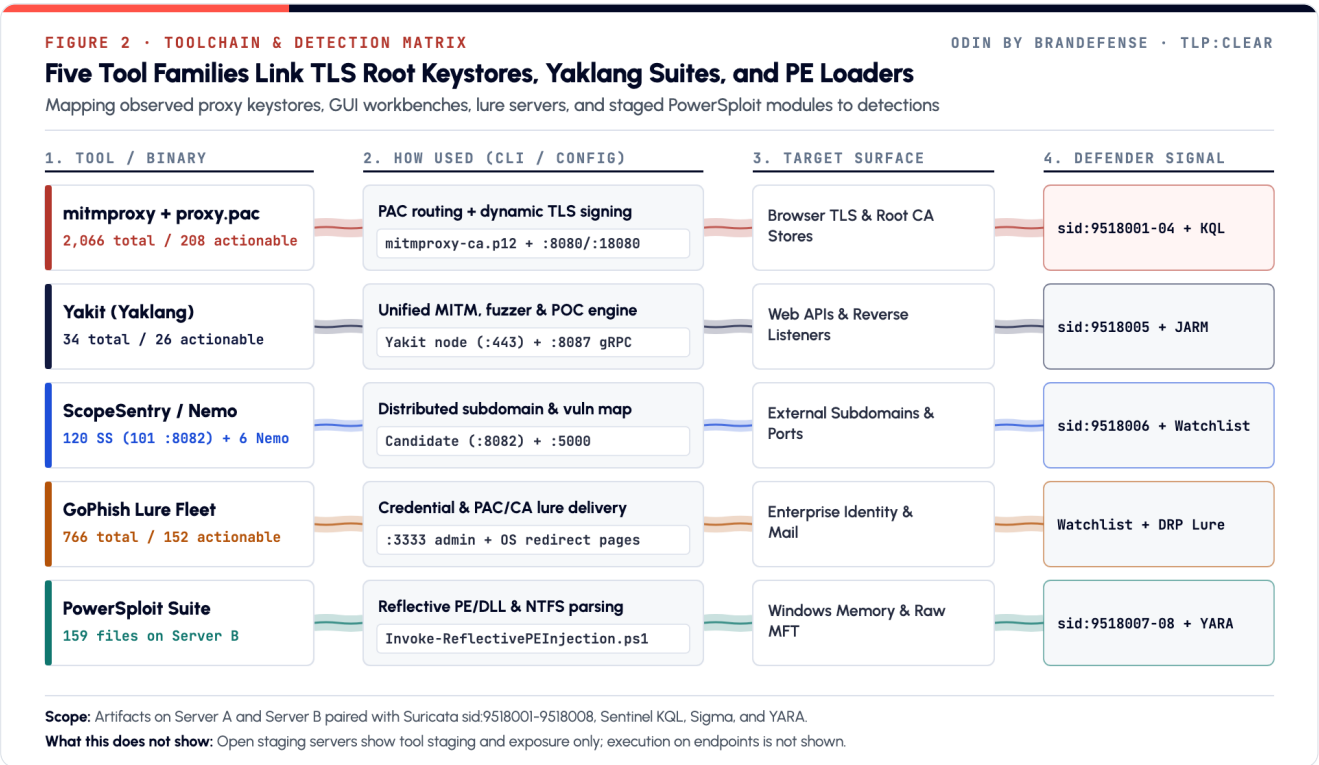


Figure 2: Five tool families link TLS root keystores, Yaklang suites and PE loaders

What the server held

Server A sits on Scaleway SAS in France. Its open directory listed 9 files totaling 11,891,086 bytes. The server was first seen before this window. We could not confirm that it was still online in November, so check liveness and ownership before any network block.

The address and port are withheld because the directory may still be reachable. Private key material and the client addresses inside the two logs are also withheld. The file names below are mitmproxy defaults, so hunters can search for them on their own systems.

File name	Size (bytes)	What it exposes
`http_server.log`	5,940,980	HTTP access log recording external fetches of `proxy.pac` and certificates
`nohup.out`	5,940,468	Output of a background HTTP or proxy process
`mitmproxy-ca.pem`	2,851	Unencrypted PEM bundle with the root CA **private key** and certificate
`mitmproxy-ca.p12`	2,338	PKCS#12 keystore with the root CA **private signing key**
`mitmproxy-ca-cert.cer`	1,172	Public root CA certificate, in the format Windows uses for trust import
`mitmproxy-ca-cert.pem`	1,172	The same public certificate in PEM form (same file hash as the `.cer`)
`mitmproxy-ca-cert.p12`	1,035	PKCS#12 wrapper of the public certificate, without the private key
`mitmproxy-dhparam.pem`	770	Diffie-Hellman parameters generated by mitmproxy
`proxy.pac`	300	JavaScript `FindProxyForURL` policy that steers browser traffic to an external proxy

Why a leaked root key matters

mitmproxy creates its own root CA. To intercept HTTPS, an endpoint must trust that root. This happens in red-team tests, developer setups and rogue proxy schemes alike.

Once the root sits in the Trusted Root Certification Authorities store, its key can sign a certificate for any domain. That includes `*.microsoftonline[.]com`, internal single sign-on portals and banking sites. The endpoint will accept those certificates without a warning.

The usual safeguard is that only the person who ran mitmproxy holds the key. On Server A, the key sat in an open directory. Any third party who fetched it gained the same signing power over every endpoint that trusts this root.

A note for hunters: Windows root-store queries use the SHA-1 thumbprint of the decoded certificate (`X509Certificate2.Thumbprint`), not a file hash. Search stores by the `CN=mitmproxy` subject, and search disks by the default mitmproxy file names.

How a PAC file steers only the traffic that matters

A proxy auto-config (PAC) file is a small script that tells a browser which proxy to use for each URL. Setting Windows `AutoConfigURL` to an external `proxy.pac` makes the browser call `FindProxyForURL(url, host)` for every request.

This is quieter than a VPN. The PAC script can send chosen domains to a high-port mitmproxy listener and let all other traffic go `DIRECT`. Across the mitmproxy fleet, the common listener ports are shown below.

Port	mitmproxy endpoints
32300	505
8080	192
8081	125
8443	64
8888	55
18080	25
3128 and 31280	9

Of the 2,066 mitmproxy endpoints, 15 also serve an open HTTP directory on the same host. One of these is confirmed and 14 are candidates.

04

CHAPTER 04

GUI Workbenches: Yakit, ScopeSentry, Nemo and GoPhish

IN SHORT

Graphical attack suites put reconnaissance, phishing and TLS interception behind one console. We counted 34 Yakit backends, 120 ScopeSentry consoles, 6 Nemo servers and 766 GoPhish endpoints.

INSIDE THIS CHAPTER

Yakit and the Yaklang engine 13

ScopeSentry and Nemo 14

GoPhish lure infrastructure 14

04 GUI Workbenches: Yakit, ScopeSentry, Nemo and GoPhish

FIGURE 3 · PROXY.PAC & ROOT CA INTERCEPTION SEQUENCE ODIN BY BRANDEFENSE · TLP:CLEAR

Hypothetical PAC Routing and Root CA Imports Could Enable Silent TLS Interception

Hypothetical six-step capability sequence across endpoint stores, staging servers, and MITM listeners



Scope: Built from Server A (proxy.pac and mitmproxy CA files), Server B PowerSploit modules, and the Yakit and mitmproxy fleets.
What this does not show: Hypothetical analytical sequence; Server A, Server B, and fleets are unlinked, and target-side delivery or execution is unverified.

Figure 3: Hypothetical PAC routing and root CA imports could enable silent TLS interception

Figure 3 shows a hypothetical six-step sequence. A PAC fetch and a root CA import could let a mitmproxy or Yakit listener read diverted HTTPS, followed by a PowerSploit download. Every step is a capability. None was seen on a target.

Yakit and the Yaklang engine

Yakit pairs a desktop Electron GUI with a remote `yak` engine that speaks gRPC over TLS. One Yaklang daemon runs interactive MITM, parameter fuzzing, Nuclei-compatible PoC templates and reverse-shell listeners.

Detail	Value
Endpoints	34 on 31 IPs
Confirmed / suspected / candidate	11 / 15 / 8
Confirmed or suspected share	26 of 34 (76.5%)
Ports	443 (15), 8087 (6), 80 (2); 2800 also seen
Top countries	China (17), Hong Kong (9), United States (3)
Main hosting	Tencent, Alibaba, AWS

One suspected Yakit backend on Alibaba Cloud in China, port 443, was seen on 18 November 2026. Its JARM value is `29d29d20d29d29d22c42d42d000000937221baefa0b90420c8e8e41903f1d5`. Its address is withheld because it is suspected, not confirmed.

ScopeSentry and Nemo

Both platforms automate subdomain enumeration, port scanning, web spidering and vulnerability correlation across distributed worker nodes.

ScopeSentry clusters on its default port 8082, with 101 of 120 endpoints (84.2%). Only 10 rank as actionable, 109 are candidates and one sits on a dynamic-ISP watchlist. One uncorroborated candidate on Hetzner in Germany, port 8082, was seen on 18 November 2026. Its address is withheld.

All 6 Nemo instances are confirmed, and all listen on 5000.

GoPhish lure infrastructure

Detail	Value
Endpoints	766 on 672 IPs
Actionable lure servers	152 (19.8%)
Watchlist or candidate	614
Ports	3333 admin (417, 54.4%), 443 listener (127, 16.6%), 80 (37)
Top countries	United States (151), Germany (109), France (70)
Main hosting	AWS, OVH, DigitalOcean

The GoPhish fleet fits credential lures. No `.pac` routing page or `.cer` onboarding page was seen on these endpoints. Using GoPhish to deliver a root CA or a PAC URL remains a hypothesis.

05

CHAPTER 05

Inside Server B: PowerSploit and Lure Staging

IN SHORT

Server B is an open HTTP server holding 314 files across 33 directories, including a full PowerSploit tree and phishing material. Folder names point to an assessment workspace, so we treat it as a watchlist entry and hunt its file hashes.

INSIDE THIS CHAPTER

What the server is 16

In-memory PE and DLL loading 17

Raw NTFS access and relays 17

SharePoint and admin portal wordlists 18

05 Inside Server B: PowerSploit and Lure Staging

FIGURE 4 · STAGING SERVER ARCHITECTURAL BLUEPRINT — ODIN BY BRANDEFENSE · TLP:CLEAR

Two Open Servers Exposed a TLS Root Keystore and a 314-File Engagement Tree

Architectural cut-away of unlinked Server A (Scaleway FR) and Server B (DigitalOcean US)



Scope: Observed by Odin by Brandefense, two-week window ending 18 Nov 2026; both servers first seen earlier; 2 Server B subdirs seen 18 Nov.
What this does not show: Host IPs, ports, and project and target folder names are withheld.

Figure 4: Two open servers exposed a TLS root keystore and a 314-file engagement tree

What the server is

Server B sits on DigitalOcean in the United States. Its address, port and folder names are withheld. Folder names suggest a contracted red-team or awareness assessment, and some folders are named after a target organization. No working script bodies are reproduced here.

The root listing of 314 files was first seen before this window. In our two-week observation window, two PowerSploit subfolders were seen again on 18 November 2026: `CodeExecution` (57 files) and `AntivirusBypass` (4 files). Whether the rest of the tree is still online is not confirmed.

Folder	Files	What it holds
Root	314	PowerSploit, phishing kits, executables and files flagged as secrets
Assessment parent folder (name withheld)	183	Engagement tree with cloud credential files, lures and `powercat`
`PowerSploit/` under a target-named folder	159	Full PowerSploit tree (mirrored under the assessment parent)
`PowerSploit/Exfiltration/`	58	Credential helpers and `NTFSParser/` (26 files; 13 in `NTFSParserDLL/`)
`PowerSploit/CodeExecution/`	57	`Invoke-ReflectivePEInjection.ps1`, `Invoke-DllInjection.ps1`; seen again 18 November 2026
Phishing email templates (folder name withheld)	23	21 email templates, 18 of them in an archive subfolder
`PowerSploit/Recon/`	11	Web path wordlists `generic.txt`, `admin.txt`, `sharepoint.txt`
OS redirect test folder	5	Landing pages `windows.html`, `mac.html`, `linux.html`, `android.html`
`PowerSploit/AntivirusBypass/`	4	`Find-AVSignature.ps1` (6,583 bytes) and module manifests; seen again 18 November 2026

A sample of 95 file records across 14 extension groups shows the mix: 18 `.h`, 15 `.cpp`, 10 `.txt`, 7 `.asm`, 6 `.ps1`, 6 `.html`, 6 `.vcxproj` and 27 project or image files.

In-memory PE and DLL loading

The `CodeExecution` folder holds the classic PowerSploit loaders. They are designed to map a PE file straight into `powershell.exe` or a remote process, without loading a DLL from disk.

File	Size (bytes)
`Invoke-ReflectivePEInjection.ps1`	135,782
`Invoke-Shellcode.ps1`	23,817
`Invoke-WmiCommand.ps1`	14,479
`Invoke-DllInjection.ps1`	12,721

The folder also holds Visual Studio C++ test harnesses (`DemoDLL_RemoteProcess`, `ExeToInjectInTo`). It has x86 and x64 assembly stubs as well: `LoadLibraryA.asm`, `GetProcAddress.asm` and `CallDllMain.asm`. These scripts are usually pulled into memory with a PowerShell download cradle.

Raw NTFS access and relays

The `NTFSParserDLL` folder (13 files) holds C++ code that reads raw NTFS volumes directly. Key files are `NTFS_Attribute.h` (38,460 bytes), `NTFS_FileRecord.h` (24,011 bytes) and `NTFSParserDLL.cpp` (3,795 bytes). Reading the raw volume can bypass operating system file locks.

Next to it, the assessment folder holds `powercat.ps1` (37,641 bytes), a PowerShell TCP and UDP relay. `Find-AVSignature.ps1` (6,583 bytes) tests where antivirus signatures start and end in a file.

SharePoint and admin portal wordlists

Three wordlists target enterprise portals: `generic.txt` (33,792 bytes), `admin.txt` (3,072 bytes) and `sharepoint.txt` (2,662 bytes). The SharePoint list includes paths such as `_layouts/mngsiteadmin.aspx`, `_layouts/aclinv.aspx` and `_catalogs/masterpage/Forms/AllItems.aspx`.

The server pairs these lists with the OS-specific redirect pages and awareness lure templates. Together they look like an engagement kit: map the portal, send a lure, then stage tooling.

CHAPTER 06

Fleet Comparison and How to Treat Each Finding

IN SHORT

mitmproxy and GoPhish produce large candidate volumes, while Yakit and Nemo are small but mostly confirmed. Block only on several signals that agree.

INSIDE THIS CHAPTER

The five fleets side by side 20

mitmproxy is mostly noise, with a hard core 20

Three alternative explanations, and what to do with each 21

06 Fleet Comparison and How to Treat Each Finding

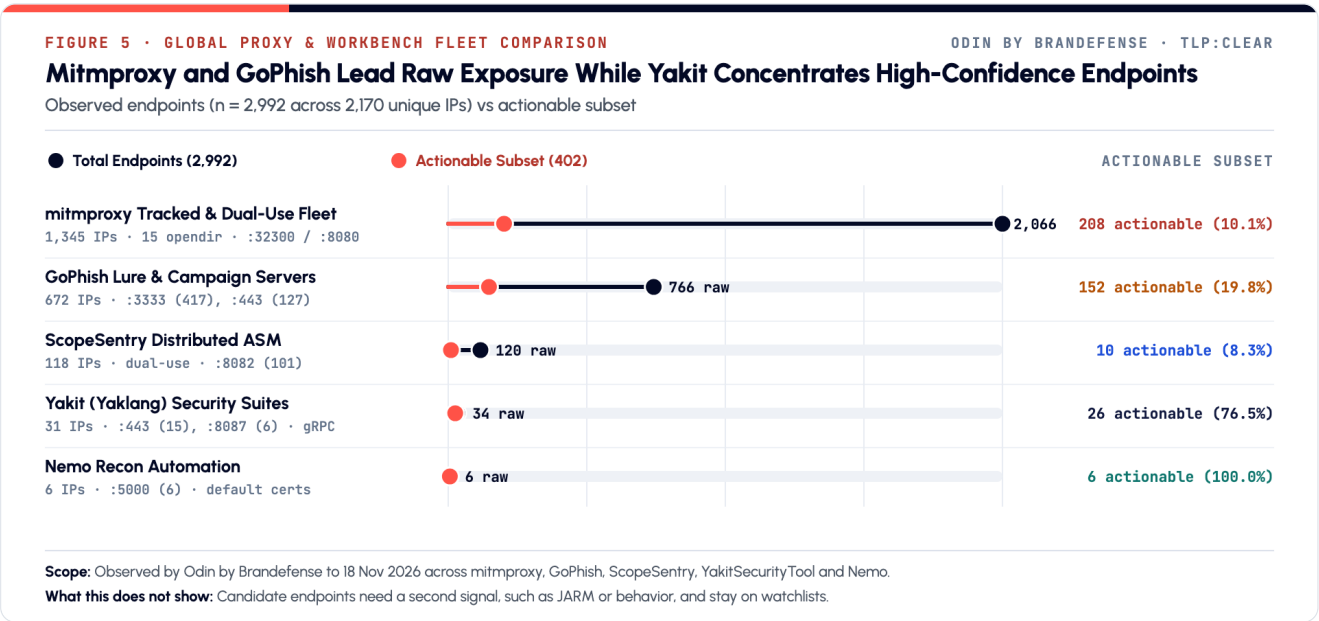


Figure 5: mitmproxy and GoPhish lead raw exposure while Yakit concentrates high-confidence endpoints

The five fleets side by side

Fleet	Endpoints (IPs)	Actionable (share)	Main ports	Top countries	Main hosting
mitmproxy	2,066 (1,345)	208 (10.1%)	32300 (505), 8080 (192), 8081 (125)	NL (592), CN (584), US (314)	ALFA TELECOM, Alibaba
GoPhish	766 (672)	152 (19.8%)	3333 (417), 443 (127), 80 (37)	US (151), DE (109), FR (70)	AWS, OVH, DigitalOcean
ScopeSentry	120 (118)	10 (8.3%)	8082 (101), 18082 (3), 8443 (3)	CN (37), HK (22), US (17)	Tencent, Alibaba
Yakit (Yaklang)	34 (31)	26 (76.5%)	443 (15), 8087 (6), 80 (2)	CN (17), HK (9), US (3)	Tencent, Alibaba, AWS
Nemo	6 (6)	6 (100.0%)	5000 (6)	CN (5), US (1)	Tencent, Alibaba, HostPapa

mitmproxy is mostly noise, with a hard core

Of the 2,066 mitmproxy endpoints, 208 (10.1%) are confirmed (161) or suspected (47). The other 1,858 (89.9%) are candidates.

Many candidates sit on port 32300 (505 endpoints) inside commercial proxy networks. ALFA TELECOM s.r.o. hosts 377 of them and Serverel Inc. hosts 128. Ports 32300 and 8080 carry a lot of commercial and dual-use proxy traffic. Do not block by port alone. Require a PAC, JARM or certificate-store signal first.

Three alternative explanations, and what to do with each

Question	What the evidence shows	What to do
Is Server B an authorized test or hostile staging?	Folder names, a proofs folder and an awareness page point to a contracted assessment. No use against a target is shown.	Watchlist pending review. Hunt its PowerSploit file hashes on endpoints.
Is Server A hostile?	It exposed a working root CA signing key and a `proxy.pac` router, whoever runs it.	Block list for root-store removal and historical egress hunting. Check liveness and ownership before a network block.
Are ScopeSentry and GoPhish candidates attacker C2?	Internal security teams and bug-bounty researchers use both tools widely. Only 10 of 120 ScopeSentry and 152 of 766 GoPhish endpoints are actionable.	Keep candidates, such as the Hetzner ScopeSentry console, on a dual-use scanner watchlist.
Is the suspected Yakit node an active workbench?	One strong signal, no endpoint evidence.	Watch and rate-limit. Contain a host only if endpoint signals agree: a root CA import, an `AutoConfigURL` change or a PowerShell loader.

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

Within 24 hours

- [ ] Inventory Windows, macOS and browser root stores. Compare them with your approved SHA-1 thumbprints (`X509Certificate2.Thumbprint`). Remove or distrust any `CN=mitmproxy` root that nobody approved.
- [ ] Search disks and certificate stores for the default mitmproxy file names and `CN=mitmproxy` roots.
- [ ] Load the Suricata rules in alert mode and fill the approved PAC server list first.

Within 72 hours

- [ ] Lock `AutoConfigURL` through Group Policy or MDM, and turn off unauthenticated WPAD fallback.
- [ ] Alert on, or block, browser fetches of `proxy.pac` from external hosts.
- [ ] Watch browser egress to 8080, 8081, 8087, 8888, 18080, 31280 and 32300 outside documented gateways. Block when a PAC or root CA signal agrees.

Within 7 to 30 days

- [ ] Turn on PowerShell Script Block Logging and AMSI. Alert on `Invoke-ReflectivePEInjection`, `Invoke-DllInjection` and `powercat.ps1`.
- [ ] Audit raw volume reads (`\\.\C:`) by processes other than backup tools, to catch NTFS parser use.
- [ ] Rate-limit bursts of SharePoint `_layouts/` enumeration.
- [ ] Keep Server B, the suspected Yakit node and the ScopeSentry candidate on review watchlists, not block lists.

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 25
How the other rules work 25
How to sort a host 26

08 Detection Rules

The full rules are in Appendix B. Every rule needs local tuning before automated action.

Rule file	What it catches
`odin_mitmproxy_yakit_scopesentry_powersploit.rules`	Suricata 7: eight signatures for mitmproxy CA downloads, external PAC files, Yakit and ScopeSentry markers, and PowerSploit fetches
`sentinel_rogue_root_ca_and_proxy_pac_tamper.kql`	Microsoft Sentinel and Defender XDR: root CA imports, external `AutoConfigURL` edits, PowerSploit strings, and proxy-port egress
`sigma_windows_root_ca_install_and_proxy_pac_hijack.yml`	Sigma (2 rules): certificate imports of mitmproxy roots, and external PAC registry edits or PowerSploit loader strings
`odin_mitmproxy_keystore_pac_and_powersploit.yar`	YARA (4 rules): PEM root CA bundles, external PAC routing scripts, PowerSploit reflective loaders, and SharePoint recon wordlists

How the Suricata signatures work

SID	Looks for	Why
9518001	HTTP `GET` for `mitmproxy-ca.p12`, `.pem`, `.cer` or the `-cert` variants	Catches a root CA or keystore download
9518002	Outbound request for a URI ending in `/proxy.pac` to a host not on the approved PAC list	Catches a browser using an external PAC file
9518003	Inbound response with `FindProxyForURL` and `PROXY` from a host not on the approved PAC list	Catches the PAC script itself arriving
9518004	HTTP host `mitm.it`	Catches the mitmproxy certificate onboarding page
9518005	`yakit`, `Yaklang` or `grpc-yakit` in HTTP headers to 443, 2800 or 8087	Catches Yakit engine traffic
9518006	`/api/` request with a `ScopeSentry` header to 8082 or 18082	Catches ScopeSentry console API use
9518007	`GET` for `Invoke-ReflectivePEInjection`, `Invoke-DllInjection`, `Invoke-Shellcode` or `Invoke-WmiCommand` `.ps1`	Catches a PowerSploit loader download
9518008	`GET` under `/PowerSploit/Exfiltration/NTFSParser/` or `/PowerSploit/Recon/Dictionaries/`	Catches NTFS parser or recon wordlist downloads

SIDs 9518002 and 9518003 need an `APPROVED_PAC_SERVERS` address group in `suricata.yaml`. Suricata will not load them without it. Leave it empty and normal cloud PAC delivery will fire both rules. Exclude mobile and QA lab VLANs from SID 9518004, since developers there may onboard `mitm.it` on purpose.

How the other rules work

The KQL query looks back 14 days and has three branches. The first reads process events. It flags `certutil.exe`, `powershell.exe` or `pwsh.exe` running `-addstore`, `Import-Certificate` or `Import-PfxCertificate` with a root or mitmproxy file. It also flags PowerSploit names on a command line.

The second branch reads registry events for `AutoConfigURL`. It parses the host from the URL with `parse_url` and compares it with an approved PAC host list. This stops a subdomain or query string from slipping past the list.

The third branch reads network events. It flags contact with withheld staging addresses you add from your own feed. It also flags browsers or PowerShell reaching public addresses on 8080, 8081, 8087, 8888, 18080, 31280 or 32300. Fill the approved PAC host list and the PKI admin host list before alerting.

The first Sigma rule fires on `certutil` or a PowerShell import cmdlet aimed at the root store, together with a mitmproxy file name. The second fires on an `AutoConfigURL` change to an external `.pac` that fails an anchored hostname check, or on PowerSploit loader strings.

YARA rule	Fires when
`ODIN_Mitmproxy_Exposed_Root_CA_Keystore_Bundle`	PEM private-key and certificate headers appear with one of the three Base64 forms of the string `mitmproxy`
`ODIN_Rogue_Proxy_PAC_External_Listener_Routing`	A file under 64 KB has `function FindProxyForURL`, a `PROXY` directive and one of the ports 8080, 18080, 31280 or 32300
`ODIN_PowerSploit_ReflectivePE_And_Injection_Staging`	A file under 500 KB has two of four `Invoke-ReflectivePEInjection` loader strings
`ODIN_PowerSploit_SharePoint_And_Admin_Recon_List`	A file under 256 KB has all three listed SharePoint paths, or all three listed admin paths

Each YARA rule needs at least two strings, and all carry file-size limits. They have no allowlist, so tune them before automated action.

How to sort a host

The triage logic never contains a host on uncorroborated evidence.

What the host shows	Action
A corroborated root-store implant or endpoint execution	Block and contain the host
An external PAC hijack	Isolate the browser and reset the PAC setting
A single certificate hash, a `CN=mitmproxy` subject, PowerSploit staging or a suspected Yakit node	Watchlist and rate-limit
An uncorroborated recon console	Dual-use scanner watchlist
An authorized assessment	Allowlist
An approved corporate proxy	Ignore

09

CHAPTER 09

Indicators

IN SHORT

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

INSIDE THIS CHAPTER

File hashes from Server A 28
File hashes from Server B 28
MITRE ATT&CK mapping 29

09 Indicators

Appendix A holds the 9 published file hashes, all public-tool or wordlist files from Server B. The other 15 records in the selected set are withheld.

Type	Count
Root CA keystore host for root-store cleanup (address withheld)	1
Review or watchlist hosts (addresses withheld)	5
File hashes (SHA-256) for hunting, from Server B	9
Withheld file hashes (8 distinct hashes for Server A's 9 files, 1 Server B landing page)	9

The five watchlist entries are Server B, the suspected Yakit node, the ScopeSentry candidate, and one representative node each for the mitmproxy and GoPhish fleets. Supporting fingerprints are the subject `CN=mitmproxy`, the onboarding host `mitm.it`, and the Yakit JARM value in Chapter 4.

File hashes from Server A

All Server A file hashes are withheld, because the server's owner is not established. Hunt by the default mitmproxy file names and the certificate subject `CN=mitmproxy` instead.

File hashes from Server B

File	Size (bytes)	SHA-256	Where to hunt
`Invoke-ReflectivePEInjection.ps1`	135,782	`c81d0073c92d0b4d0cc66a4c1d64d4ca2afbe56d78eb9d9611be8055919fd8ba`	Endpoint, EDR
`Invoke-Shellcode.ps1`	23,817	`457cd41fbb528812aa51bc4b31fce042cdf736281b162181d91c47733d0e9e4b`	Endpoint, EDR
`Invoke-WmiCommand.ps1`	14,479	`2cdba7fab6f6fcec2e4f10fc1a45c13a6a9d26cca2ea8b8b4e15989a198618ef`	Endpoint, EDR
`Invoke-DllInjection.ps1`	12,721	`788121850fe9095492f79af631d43e342cdcf7a09d1becf97356dc97e4ae869d`	Endpoint, EDR
`powercat.ps1`	37,641	`cb978873ed0877c8f2c9b4d9b4cfd1c3fc4eff223abb3afc24ec687cc77f93b8`	Endpoint, EDR
`Find-AVSignature.ps1`	6,583	`366f5b37821f4266697350847048d2df55f6de96ad1e578baf910b9da8eed742`	Endpoint, EDR
`NTFSParserDLL.cpp`	3,795	`814a812279c600b707500322d4e1027795cd90ed09a69a2802078c075a0ac743`	Server and network
`admin.txt`	3,072	`f2e72552cd62d3e9ecc82b80a6918706d752fcf24799dd6b2c1c09caa96307b3`	Server and network
`sharepoint.txt`	2,662	`aafc643316b5486764cb99a655978860140dfdd2aa1a19f14e955ecf5683ef48`	Server and network

Withheld from Server A: all file hashes, the key material and client addresses. Withheld from Server B: the landing page hash. Also withheld: both server addresses and ports, Server B folder names that identify an organization, and every suspected or candidate fleet address.

MITRE ATT&CK mapping

The rules map to three ATT&CK tactics.

Tactic	How it applies here	Detection
Reconnaissance (TA0043)	ScopeSentry and Nemo consoles; SharePoint and admin wordlists	SID 9518006, 9518008; YARA wordlist rule
Credential Access (TA0006)	Root CA trust, external PAC routing and TLS interception that exposes session traffic	SID 9518001 to 9518005; KQL; Sigma rules
Stealth (TA0005)	Reflective in-memory loading with no DLL on disk	SID 9518007; YARA loader rule; Sigma rule 2

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.** Fleet counts and the two Server B subfolders come from a two-week observation window ending 18 November 2026. Both servers were first seen before this window; no earlier dates are given.
- **Server A may be gone.** We could not confirm Server A was still online in November. Its key material stays useful for root-store cleanup and historical hunting either way.
- **No links between clusters.** Nothing ties Server A, Server B and the five fleets to one operator. The four-role model is analytical.
- **Staged is not used.** Server B holds tooling and lures. We saw no execution, delivery or interception on any target.
- **Candidates are leads.** Most mitmproxy, GoPhish and ScopeSentry endpoints are candidates, and these tools have wide legitimate use.
- **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. MITRE ATT&CK. Reconnaissance tactic (TA0043). https://attack.mitre.org/tactics/TA0043/ (verified 19 November 2026)
2. MITRE ATT&CK. Credential Access tactic (TA0006). https://attack.mitre.org/tactics/TA0006/ (verified 19 November 2026)
3. MITRE ATT&CK. Stealth tactic (TA0005). https://attack.mitre.org/tactics/TA0005/ (verified 19 November 2026)
4. mitmproxy project. About Certificates. https://docs.mitmproxy.org/stable/concepts-certificates/ (verified 19 November 2026)
5. Yaklang project. Yakit and the Yaklang engine. https://www.yaklang.com/ (verified 19 November 2026)
6. Autumn-27. ScopeSentry repository. https://github.com/Autumn-27/ScopeSentry (verified 19 November 2026)
7. PowerShellMafia. PowerSploit repository. https://github.com/PowerShellMafia/PowerSploit (verified 19 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 (9) 35

B. Detection rules (4 files) 36

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
R18-IOC-01	SHA-256	c81d0073c92d0b4d0cc66a4c1d64d4ca2afbe56d78eb9d9611be8055919fd8ba		Endpoint Post-Exploit	HUNT
R18-IOC-02	SHA-256	457cd41fbb528812aa51bc4b31fce042cdf736281b162181d91c47733d0e9e4b		Endpoint Post-Exploit	HUNT
R18-IOC-03	SHA-256	2cdba7fab6f6fcec2e4f10fc1a45c13a6a9d26cca2ea8b8b4e15989a198618ef		Endpoint Post-Exploit	HUNT
R18-IOC-04	SHA-256	788121850fe9095492f79af631d43e342cdcf7a09d1becf97356dc97e4ae869d		Endpoint Post-Exploit	HUNT
R18-IOC-05	SHA-256	cb978873ed0877c8f2c9b4d9b4cfd1c3fc4eff223abb3afc24ec687cc77f93b8		Endpoint Relay Script	HUNT
R18-IOC-06	SHA-256	366f5b37821f4266697350847048d2df55f6de96ad1e578baf910b9da8eed742		AV Signature Probe	HUNT
R18-IOC-07	SHA-256	814a812279c600b707500322d4e1027795cd90ed09a69a2802078c075a0ac743		NTFS Volume Parser	HUNT
R18-IOC-08	SHA-256	f2e72552cd62d3e9ecc82b80a6918706d752fcf24799dd6b2c1c09caa96307b3		Admin Path Recon	HUNT
R18-IOC-09	SHA-256	aafc643316b5486764cb99a655978860140dfdd2aa1a19f14e955ecf5683ef48		SharePoint Recon	HUNT

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_mitmproxy_keystore_pac_and_powersploit.yar

YARA

```
/*
   Odin by Brandefense - Report #18 YARA Keystore, PAC & PowerSploit Artifact Suite
   Classification: TLP:CLEAR
   Scope: Four deterministic filesize-bounded rules matching exposed mitmproxy Root CA keystores
          (raw PEM Base64 ASN.1 DER or HTTP staging delivery logs), external proxy.pac routing scripts,
          PowerSploit reflective PE modules, and SharePoint/admin wordlists.
*/

rule ODIN_Mitmproxy_Exposed_Root_CA_Keystore_Bundle {
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        report_id = "Odin-Monograph-18"
        severity = "critical"
        description = "Detects exposed mitmproxy PEM Root CA keystores via raw PEM armor and Base64 ASN.1 DER
mitmproxy subject bytes, or staging logs serving mitmproxy-ca certificates"
    strings:
        $k1 = "-----BEGIN PRIVATE KEY-----" ascii
        $k2 = "-----BEGIN RSA PRIVATE KEY-----" ascii
        $c1 = "-----BEGIN CERTIFICATE-----" ascii
        $b64_mitm_0 = "bWl0bXByb3h5" ascii
        $b64_mitm_1 = "1pdG1wcm94e" ascii
        $b64_mitm_2 = "taXRtcHJveH" ascii
        $l1 = "GET /proxy.pac HTTP/1.1\" 200" ascii
        $l2 = "GET /mitmproxy-ca-cert.cer HTTP/1.1\" 200" ascii
    condition:
        filesize < 10MB and ((($k1 or $k2) and $c1 and any of ($b64_mitm*)) or ($l1 and $l2))
}

rule ODIN_Rogue_Proxy_PAC_External_Listener_Routing {
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        report_id = "Odin-Monograph-18"
        severity = "high"
        description = "Detects Proxy Auto-Configuration (proxy.pac) scripts routing traffic to non-standard proxy
listeners"
    strings:
        $f1 = "function FindProxyForURL" ascii nocase
        $p1 = "PROXY " ascii
        $port1 = ":8080" ascii
        $port2 = ":18080" ascii
        $port3 = ":31280" ascii
        $port4 = ":32300" ascii
    condition:
        filesize < 64KB and $f1 and $p1 and 1 of ($port*)
}

rule ODIN_PowerSploit_ReflectivePE_And_Injection_Staging {
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        report_id = "Odin-Monograph-18"
        severity = "critical"
        description = "Detects staged PowerSploit Invoke-ReflectivePEInjection or in-memory PE loader modules"
    strings:
        $s1 = "function Invoke-ReflectivePEInjection" ascii nocase
        $s2 = "PowerSploit Function: Invoke-ReflectivePEInjection" ascii nocase
        $s3 = "Reflectively loads a DLL or EXE in to memory" ascii nocase
        $s4 = "ExeToInjectInTo" ascii nocase
```

```
    condition:
        filesize < 500KB and 2 of ($s*)
}

rule ODIN_PowerSploit_SharePoint_And_Admin_Recon_List {
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        report_id = "Odin-Monograph-18"
        severity = "medium"
        hunt_scope = "staging_workbench_recon_dictionary_hunt"
        description = "Detects staged PowerSploit SharePoint (_layouts/mngsiteadmin.aspx) and admin portal
enumeration lists"
    strings:
        $sp1 = "_catalogs/masterpage/Forms/AllItems.aspx" ascii nocase
        $sp2 = "_layouts/AdminRecycleBin.aspx" ascii nocase
        $sp3 = "_layouts/mngsiteadmin.aspx" ascii nocase
        $ad1 = "admin/controlpanel.asp" ascii nocase
        $ad2 = "admin4_account/" ascii nocase
        $ad3 = "administr8.php" ascii nocase
    condition:
        filesize < 256KB and (all of ($sp*) or all of ($ad*))
}
```

odin_mitmproxy_yakit_scopesentry_powersploit.rules

SURICATA

```
# Odin by Brandefense - Report #18 Suricata 7.x TLS Interception, Yakit, ScopeSentry & PowerSploit Suite
# Classification: TLP:CLEAR
# Deployment: define APPROVED_PAC_SERVERS under address-groups in suricata.yaml (approved PAC hosts or CIDRs;
#   use a placeholder such as 192.0.2.1 if none) before loading sid:9518002 and sid:9518003.
# Scope: 8 network signatures (sid:9518001-9518008) covering mitmproxy Root CA keystore/cert downloads,
#        external proxy.pac AutoConfigURL fetches, FindProxyForURL PAC routing, mitm.it onboarding,
#        Yakit/Yaklang markers, ScopeSentry :8082 probes, and PowerSploit reflective PE/recon staging.

# 1. Mitmproxy Root CA Certificate or PKCS#12 Keystore HTTP Download
alert http any any -> any any (msg:"ODIN-MITMPROXY Exposed Root CA Keystore or Certificate HTTP Download";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"mitmproxy-ca"; nocase; fast_pattern;
pcre:"/\/mitmproxy-ca(?:-cert)?\.(?:p12|pem|cer)$/i"; metadata:created_at 2026_11_18, mitre_tactic TA0006;
classtype:trojan-activity; sid:9518001; rev:1;)

# 2. Outbound Browser Proxy Auto-Config (/proxy.pac) Request to External Public Host
# Add your approved PAC servers as a pass rule or an address-group variable before enabling SIDs 9518002 and
9518003.
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-MITMPROXY Outbound Browser Request for External /
proxy.pac"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/proxy.pac"; nocase;
endswith; fast_pattern; metadata:created_at 2026_11_18, mitre_tactic TA0006; classtype:policy-violation; sid:
9518002; rev:2;)

# 3. External Proxy Auto-Config Script Routing Traffic via FindProxyForURL
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-MITMPROXY External PAC Script Directing Traffic via
FindProxyForURL"; flow:established,to_client; http.response_body; content:"FindProxyForURL"; nocase;
fast_pattern; content:"PROXY "; nocase; metadata:created_at 2026_11_18, mitre_tactic TA0006; classtype:trojan-
activity; sid:9518003; rev:2;)

# 4. Mitmproxy Onboarding Domain (mitm.it) HTTP Host Lookup
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-MITMPROXY Certificate Onboarding Domain (mitm.it)
Request"; flow:established,to_server; http.host; content:"mitm.it"; fast_pattern; metadata:created_at 2026_11_18,
mitre_tactic TA0006; classtype:trojan-activity; sid:9518004; rev:1;)

# 5. YakitSecurityTool (Yaklang Engine / MITM / Web Fuzzer) HTTP Marker
alert http any any -> any [443,2800,8087] (msg:"ODIN-YAKIT YakitSecurityTool Yaklang gRPC or Web Fuzzer Marker";
flow:established,to_server; http.header; content:"yakit"; nocase; fast_pattern; pcre:"/(Yakit|Yaklang|grpc-
yakit)/i"; metadata:created_at 2026_11_18, mitre_tactic TA0006; classtype:trojan-activity; sid:9518005; rev:1;)

# 6. ScopeSentry Distributed ASM Console API Probe on Default Ports (:8082 / :18082)
alert http any any -> any [8082,18082] (msg:"ODIN-RECON ScopeSentry ASM Console API or Scanner Header Marker";
flow:established,to_server; http.uri; content:"/api/"; nocase; fast_pattern; http.header; content:"ScopeSentry";
nocase; metadata:created_at 2026_11_18, mitre_tactic TA0043; classtype:attempted-recon; sid:9518006; rev:1;)
```

```
# 7. PowerSploit Reflective PE or DLL Injection Script HTTP Staging
alert http any any -> any any (msg:"ODIN-POWERSPLOIT Reflective PE or DLL Injection Script HTTP Fetch";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/Invoke-"; nocase; fast_pattern;
pcre:"/\/Invoke-(?:ReflectivePEInjection|DllInjection|Shellcode|WmiCommand)\.ps1$/i"; metadata:created_at
2026_11_18, mitre_tactic TA0005; classtype:trojan-activity; sid:9518007; rev:1;)

# 8. PowerSploit NTFSParser Exfiltration or SharePoint/Admin Recon Dictionary HTTP Fetch
alert http any any -> any any (msg:"ODIN-POWERSPLOIT NTFSParser or SharePoint Recon Dictionary HTTP Fetch";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/PowerSploit/"; nocase; fast_pattern;
pcre:"/\/PowerSploit\/(?:Exfiltration\/NTFSParser|Recon\/Dictionaries)\//i"; metadata:created_at 2026_11_18,
mitre_tactic TA0043; classtype:trojan-activity; sid:9518008; rev:1;)
```

sentinel_rogue_root_ca_and_proxy_pac_tamper.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Report #18 Microsoft Sentinel / Defender XDR Hunt
// Classification: TLP:CLEAR
// Scope: Rogue Root CA imports, external AutoConfigURL proxy.pac hijacks, PowerSploit staging, and proxy egress.

let Lookback = 14d;
let AuthorizedAdminHosts = dynamic(["pki-admin-01.corp.example", "sec-jump-01.corp.example"]);
let ApprovedInternalPacHosts = dynamic(["pac.corp.example", "wpad.corp.example"]);
let VerifiedStagingIPs = dynamic(["198.51.100.10"]); // Populate from active TI blocklist feed (Server A address
withheld under TLP:CLEAR)
let SuspectedWorkbenchIPs = dynamic(["198.51.100.30", "203.0.113.15"]); // Populate from staging/Yakit review
watchlists (Server B and suspected Yakit node addresses withheld)
let RogueProxyPorts = dynamic([8080, 8081, 8087, 8888, 18080, 31280, 32300]);
union isfuzzy=true
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (AuthorizedAdminHosts))
    | where (FileName in~ ("certutil.exe", "powershell.exe", "pwsh.exe")
             and ProcessCommandLine has_any ("-addstore", "Import-Certificate", "Import-PfxCertificate")
             and ProcessCommandLine has_any ("Root", "mitmproxy-ca", ".p12", ".cer"))
         or ProcessCommandLine has_any ("Invoke-ReflectivePEInjection", "Invoke-DllInjection", "NTFSParserDLL",
"powercat.ps1")
    | extend HuntSignal = "Rogue_Root_CA_Import_Or_PowerSploit_Execution",
             ArtifactDetail = ProcessCommandLine
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
),
(
    DeviceRegistryEvents
    | where Timestamp >= ago(Lookback)
    | where RegistryValueName =~ "AutoConfigURL"
    | where RegistryValueData has_any ("http://", "https://")
    | extend PacHost = tolower(tostring(parse_url(RegistryValueData).Host))
    | where isempty(PacHost) or not(PacHost in~ (ApprovedInternalPacHosts))
    | extend HuntSignal = "External_AutoConfigURL_Proxy_PAC_Tamper",
             ArtifactDetail = RegistryValueData
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName = RegistryKey,
ArtifactDetail
),
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback) and ActionType == "ConnectionSuccess"
    | where not(DeviceName in~ (AuthorizedAdminHosts))
    | where RemoteIP in (VerifiedStagingIPs)
         or RemoteIP in (SuspectedWorkbenchIPs)
         or (RemoteIPType == "Public" and RemotePort in (RogueProxyPorts)
             and InitiatingProcessFileName in~ ("chrome.exe", "msedge.exe", "firefox.exe", "powershell.exe"))
    | extend HuntSignal = case(
             RemoteIP in (VerifiedStagingIPs), "Verified_RootCA_Or_PowerSploit_Staging_Egress",
             RemoteIP in (SuspectedWorkbenchIPs), "Suspected_Yakit_Workbench_Egress",
             "Browser_Or_PowerShell_NonStandard_Proxy_Egress"
         ),
         ArtifactDetail = strcat(RemoteIP, ":", tostring(RemotePort), " ", RemoteUrl)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName = InitiatingProcessFileName,
```

```
ArtifactDetail
)
| summarize
    FirstSeen = min(Timestamp),
    LastSeen = max(Timestamp),
    EventCount = count(),
    SampleDetails = make_set(ArtifactDetail, 5)
    by DeviceName, HuntSignal, InitiatingProcessFileName, FileName
| order by EventCount desc
```

sigma_windows_root_ca_install_and_proxy_pac_hijack.yml

SIGMA

```
title: Windows Rogue Root CA Certificate Import via Certutil or PowerShell
id: 8c52b4e1-71d8-4a30-9f11-18a095180001
status: experimental
description: Detects unauthorized Trusted Root Certification Authorities store imports referencing mitmproxy
keystores or user-writable certificate files.
author: Odin by Brandefense
date: 2026-11-18
tags:
  - attack.ta0006
  - attack.credential_access
  - attack.ta0005
  - attack.stealth
logsource:
  category: process_creation
  product: windows
detection:
  selection_certutil:
    Image|endswith: '\certutil.exe'
    CommandLine|contains|all:
      - '-addstore'
      - 'Root'
  selection_ps_cmdlet:
    CommandLine|contains:
      - 'Import-Certificate'
      - 'Import-PfxCertificate'
  selection_ps_root_store:
    CommandLine|contains|all:
      - 'Cert:\'
      - 'Root'
  selection_mitm_artifact:
    CommandLine|contains:
      - 'mitmproxy-ca'
      - '\AppData\Local\Temp\'
      - '\Downloads\'
  filter_pki_admin:
    ComputerName:
      - 'pki-admin-01.corp.example'
  condition: ((selection_certutil or (selection_ps_cmdlet and selection_ps_root_store)) and
selection_mitm_artifact) and not filter_pki_admin
falsepositives:
  - Authorized developer proxy debugging on isolated test workstations
level: critical
---
title: Windows External Proxy AutoConfigURL Tamper and Reflective PE Staging
id: 8c52b4e1-71d8-4a30-9f11-18a095180002
status: experimental
description: Detects command-line tampering of Internet Settings AutoConfigURL toward external HTTP/HTTPS
proxy.pac scripts or execution of PowerSploit reflective PE loaders.
author: Odin by Brandefense
date: 2026-11-18
tags:
  - attack.ta0006
  - attack.credential_access
  - attack.ta0005
  - attack.stealth
logsource:
```

```
    category: process_creation
    product: windows
  detection:
    selection_pac_key:
      CommandLine|contains|all:
        - 'AutoConfigURL'
        - '.pac'
    selection_pac_scheme:
      CommandLine|contains:
        - 'http://'
        - 'https://'
    selection_powersploit:
      CommandLine|contains:
        - 'Invoke-ReflectivePEInjection'
        - 'Invoke-DllInjection'
        - 'NTFSParserDLL'
        - 'powercat.ps1'
    filter_corporate_pac:
      CommandLine|re:
        - '(?i)^(?:[^:]|:[^/])*\bAutoConfigURL\b(?:[^:]|:[^/])*\s(?:/d|-Value)\s+["'']?https?://(?:pac|wpad)
\.corp\.example/[A-Za-z0-9._/-]+\.pac["'']{0,2}(?:\s+/f)?\s*$'
    condition: ((selection_pac_key and selection_pac_scheme) and not filter_corporate_pac) or selection_powersploit
  falsepositives:
    - Managed cloud Secure Web Gateway PAC enrollment scripts (allowlist verified corporate PAC FQDNs)
  level: high
```

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/exposed-root-ca-keys/
Short Read · brandefense.io/odin-research/blog/a-leaked-mitmproxy-root-key-and-the-tools-built-around/
Defense Kit · brandefense.io/odin-research/defense-kit/