ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 32 / 30 · INTRUSION WORKFLOWS

ADCS Relays, Token-Gated Webshells and Macro Phishing

Three Exposed Workbenches

What three open directories show about certificate relay, modular PHP backdoors and GoPhish macro templates, and why scanner banners alone are not indicators

Prepared November 2026
Full Report · Public edition · Version 1.1

1,166
Dual-use scanner endpoints on 1,063 hosts (Nessus, Acunetix, OpenVAS, OWASP ZAP)

0
Shared addresses, certificates or file hashes between the scanners and the three workbenches

3
Open-directory workbenches, with 888 listed entries across six ports

3.99 MB
Largest ADCS relay log (path withheld), among 276 files on the Moldovan server

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

What three open directories show about certificate relay, modular PHP backdoors and GoPhish macro templates, and why scanner banners alone are not indicators

01	Executive Summary	04
02	Dual-Use Scanners: Context, Not Indicators	07
03	Workbench A: ADCS ESC8 Relay Logs	10
04	Workbench B: A Modular Token-Gated PHP Kit	13
05	Workbench C: GoPhish Macro Templates Beside Adminer	16
06	Other Explanations	19
07	What Defenders Should Do	21
08	Detection Rules	23
09	Indicators	26
10	Scope and Limits	28
11	Sources	30
A	Indicators of compromise (0)	32
B	Detection rules (4 files)	33

IN SHORT

On 20 November 2026, Odin by Brandefense recorded three open directories containing files associated with intrusion work. One holds Active Directory Certificate Services (ADCS) relay logs of up to 3.99 MB. One holds a small PHP webshell kit that appears to hide behind bearer tokens. One holds GoPhish macro templates next to an Adminer database console. We also mapped 1,166 dual-use scanner endpoints, which are context, not indicators. Enforce Extended Protection on ADCS web enrollment, watch PHP workers for shells, and block internet macros.

01

CHAPTER 01

Executive Summary

IN SHORT

On 20 November 2026, Odin by Brandefense recorded three open directories containing files associated with intrusion work. One holds Active Directory Certificate Services (ADCS) relay logs of up to 3.99 MB. One holds a small PHP webshell kit that appears to hide behind bearer tokens. One holds GoPhish macro templates next to an Adminer database console. We also mapped 1,166 dual-use scanner endpoints, which are context, not indicators. Enforce Extended Protection on ADCS web enrollment, watch PHP workers for shells, and block internet macros.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 04

01 Executive Summary

Enterprise intrusions rarely depend on custom malware at every step. Operators mix commercial scanners, public relay tools, small PHP kits and open-source phishing frameworks. Defenders, managed security providers and red teams use the same tools every day. This report separates what defenders can use with a cyber threat intelligence platform from what is only noise.

1. **Scanner banners are context, not indicators.**
   We mapped 1,166 Nessus, Acunetix, OpenVAS and OWASP ZAP endpoints on 1,063 hosts. Most sit on major cloud providers, and nine sit on dynamic telecom networks. No shared address, certificate or file hash links them to the three workbenches.

2. **Open directories hold ADCS ESC8 relay logs and a token-gated PHP kit.**
   A server in Moldova lists 276 files, including four ADCS relay logs ranging from 619 bytes to 3.99 MB. A server in China repeats an eight-file PHP login and token core across three ports. Going by the file names, that core guards a 509-byte command entry point and hides 117-byte PHP stubs in asset folders.

3. **A GoPhish source tree sits beside a large archive and Adminer.**
   A server in India mirrors 290 files across two web listeners, an exposure defenders can track with external attack surface management. They include a GoPhish source tree with Base64 macro-enabled Word templates, a sync-named Python service, a single-file Adminer console and a 616.6 MB archive.

Three Decisions for Leaders

1. **Close the ADCS HTTP relay path.** Remove HTTP Web Enrollment where it is not needed. Elsewhere, require HTTPS with Extended Protection for Authentication, as Microsoft KB5005413 describes.
2. **Detect webshells by behaviour, not by one file.** Alert when PHP workers start shells or database clients, turn off PHP in static asset folders, and remove unapproved Adminer consoles.
3. **Keep scanner banners out of block lists.** Use them as enrichment. Put the effort into mail and endpoint controls: block macros in files from the internet and review inbound `.docm` attachments.

The Study in Numbers

Number	What it is
1,166	Dual-use scanner endpoints on 1,063 hosts (Nessus, Acunetix, OpenVAS, OWASP ZAP)
0	Shared addresses, certificates or file hashes between the scanners and the three workbenches
3	Open-directory workbenches, with 888 listed entries across six ports
3.99 MB	Largest ADCS relay log (path withheld), among 276 files on the Moldovan server
32	Files in the token-gated PHP kit across three ports on the Chinese server
290	Files per listener in the GoPhish and Adminer tree on the Indian server (580 in total)
9	Detection rules: 3 Suricata, 3 Sigma, 2 YARA and 1 Sentinel KQL query

How to use this report.

Security leaders can read Chapters 1, 6 and 7. Active Directory, Linux web and messaging teams should read Chapters 3 to 5 and Chapter 7. SOC and detection engineers should start with Chapters 8 and 9. CTI analysts will find the scanner and workbench detail in Chapters 2 to 5.

02

CHAPTER 02

Dual-Use Scanners: Context, Not Indicators

IN SHORT

The 1,166 scanner endpoints run the same software that security teams run every day. None of them links to the three workbenches, so treat them as enrichment and never as a block list.

INSIDE THIS CHAPTER

Four scanner families 07

Where the scanners live 08

TLS fingerprints and certificates 08

02 Dual-Use Scanners: Context, Not Indicators

A vulnerability scanner banner tells you which product listens on a port. It does not tell you who owns the server or why. Many of these consoles belong to internal security teams, managed security providers and auditors. Blocking them by address breaks legitimate work and protects little.

We organise the scanners and the three workbenches as an **analytical four-stage model**: perimeter discovery, internal certificate relay, web persistence and macro phishing. This is a model, not one actor. No shared address, TLS certificate or file hash links the scanners to the workbenches, or the workbenches to each other.

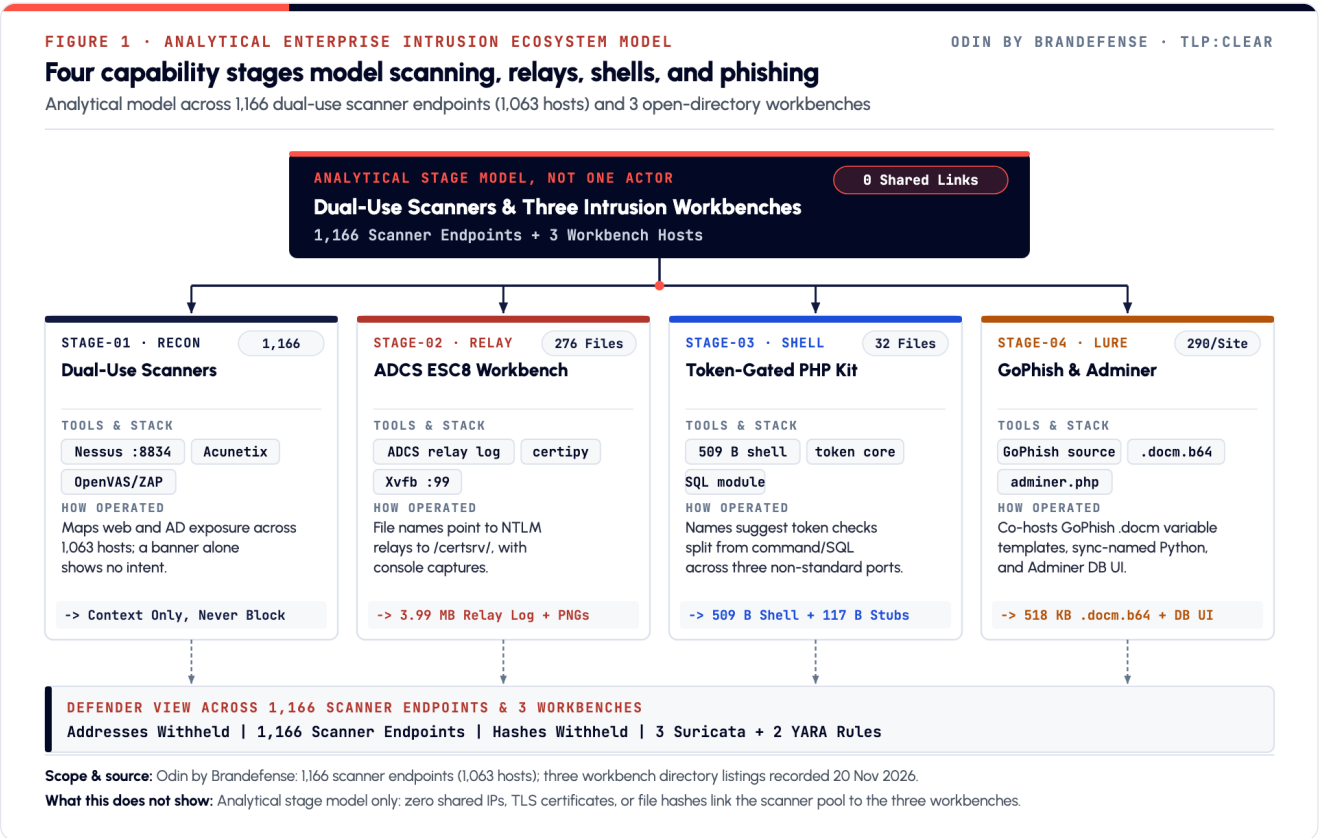


Figure 1: Four capability stages model scanning, relays, shells and phishing

Four scanner families

The 1,166 endpoints sit on 1,063 distinct hosts. The per-family host counts add up to 1,070 because seven hosts expose two scanner families at once. Four of those pair Nessus with OpenVAS. The other three pair Nessus with Acunetix, Acunetix with OpenVAS, and Acunetix with OWASP ZAP.

Scanner family	Endpoints	Share	Distinct hosts	Main ports (endpoints)
Tenable Nessus (`nessusva`)	489	41.9%	461	8834 (370, 75.7%)
Acunetix (`acunetix`)	474	40.7%	428	3443 (235, 49.6%), 7880 (69, 14.6%), 13443 (33, 7.0%)
Greenbone OpenVAS (`openvas`)	180	15.4%	163	443 (99), 80 (34), 9392 (9)
OWASP ZAP API (`owaspzapapi`)	23	2.0%	18	443 (8), 80 (6), 8080 to 8091 (6)
Total	**1,166**	**100%**	**1,063**	

Nessus and Acunetix mostly keep their default management ports. OpenVAS and ZAP sit mostly behind ordinary web ports, which suggests a reverse proxy in front.

Where the scanners live

Most endpoints sit on large cloud and VPS providers. The top six are Amazon Web Services (86), Contabo (75), OVH (58), Hetzner (47), Microsoft Azure (39) and DigitalOcean (37). By country, the largest groups are the United States (294), France (112), Germany (108), China (99) and India (63).

Nine endpoints (0.8%) sit on dynamic telecom networks. Telecom providers rotate subscriber addresses, so a static block can hit an unrelated customer hours later. Three endpoints sit on hosts that also expose an open directory, which deserves a closer look by an analyst.

TLS fingerprints and certificates

Of the 1,166 endpoints, 723 (62.0%) present a TLS JARM fingerprint and 443 (38.0%) do not. JARM describes how a server's TLS stack answers a set of handshakes. It identifies the software stack, not the operator. The three most common JARM values cover 186 of the 723 TLS endpoints (25.7%).

Scanner family	JARM	Endpoints
Acunetix	`2ad2ad0002ad2ad0002ad2ad2ad2ad02098c5f1b1aef82f7daaf9fed36c4e8`	79
Nessus	`2ad2ad0002ad2ad00042d42d000000020120996177a65431cde640fa58d2e8`	59
OpenVAS-linked stacks	`21d19d00021d21d00042d43d0000005ad20eceaf7f71ae0887d2ff117bf97f`	48

Certificate subjects tell the same story of ordinary deployments.

Certificate subject	Endpoints
No certificate (plain HTTP)	386
A domain name (withheld)	376
Generic VM or appliance hostname	277
`localhost` default	57
Self-signed, IP address as subject	36
`buildkitsandbox` container default	26
`CloudFlare Origin Certificate`	8

Use these JARM values only to label traffic. Legitimate installations of the same products present the same values.

CHAPTER 03

Workbench A: ADCS ESC8 Relay Logs

IN SHORT

A server in Moldova lists 276 files on a non-standard port, including four ADCS relay logs ranging from 619 bytes to 3.99 MB. The file names suggest ESC8 relay activity against certificate web enrollment, next to a text file named for directory replication. Enforce EPA on `/certsrv/`, or remove HTTP enrollment.

INSIDE THIS CHAPTER

What ESC8 is 10

Four relay logs 11

A headless browser and paired screenshots 11

03 Workbench A: ADCS ESC8 Relay Logs

What ESC8 is

Active Directory Certificate Services (ADCS) can offer a web enrollment page at `/certsrv/`. When that page accepts NTLM over plain HTTP without Extended Protection for Authentication (EPA), it is open to relay.

An attacker on the internal network coerces or intercepts NTLM authentication from a domain machine. The attacker then forwards it to `/certsrv/certfnsh.asp`. This attack is known as ESC8 (T1557.001, T1649).

If the Certification Authority accepts the request, it issues a client-authentication certificate for the relayed account. The holder can use that certificate to request Kerberos tickets (PKINIT). If the relayed account is a domain controller, that can lead to directory replication rights such as DCSync (T1003.006). Microsoft's KB5005413 describes the fix.

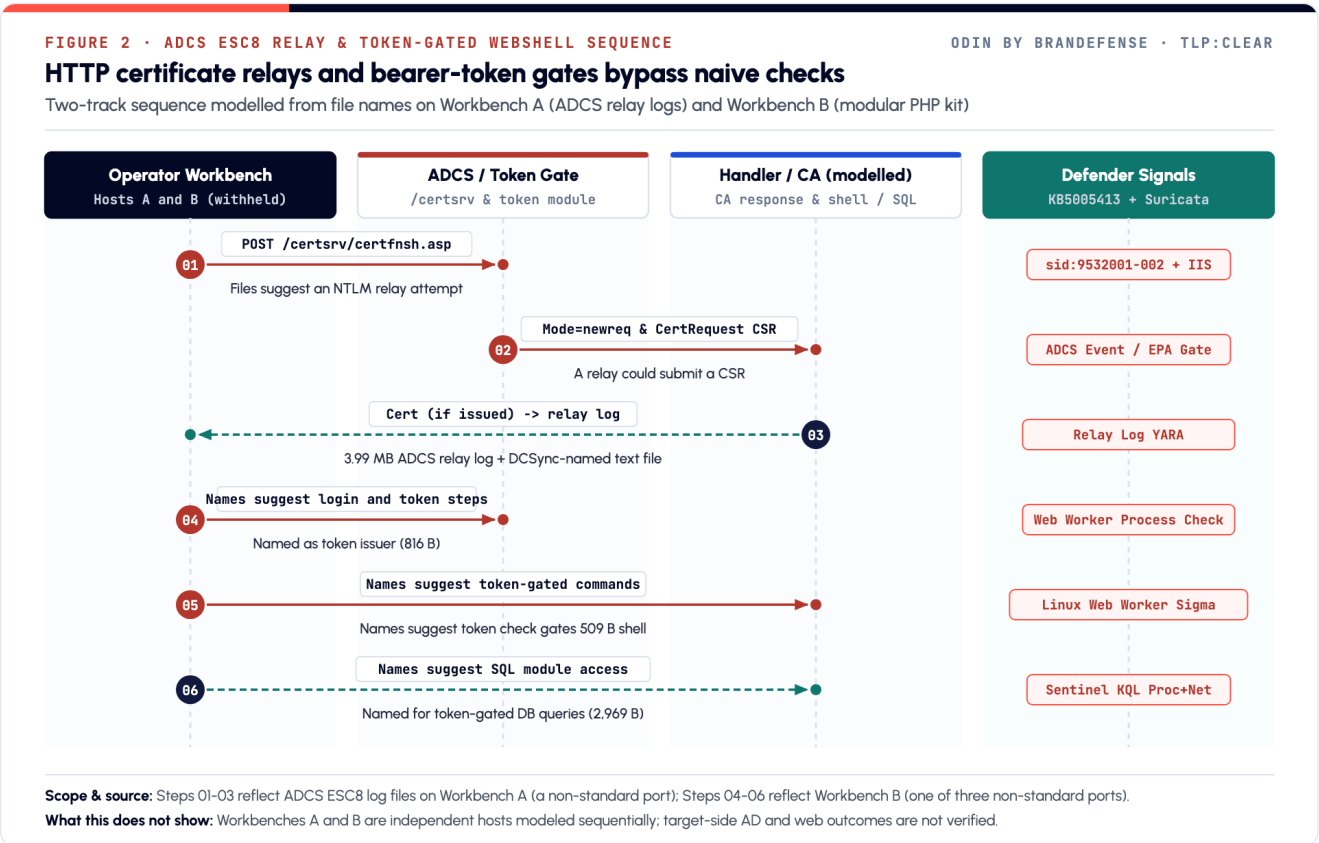


Figure 2: HTTP certificate relays and bearer-token gates bypass naive checks

Four relay logs

The directory's own HTTP server log, named after the listening port (withheld), is 8,720,527 bytes. Of 180 entries described in detail, 100 are PNG images, 28 are `.log` files, 18 are `.txt` notes, 13 are HTML pages and 13 have no extension. Four `.log` file names suggest ADCS relay activity; their contents and run history were not verified.

File	Size (bytes)	What it suggests
Certipy-named relay log (path withheld)	619	File named for the public Certipy relay tool
ADCS relay log 1 (path withheld)	15,586	File named for an ADCS relay
Root-named ADCS relay log (path withheld)	21,212	File named for privileged-account relay targeting
ADCS relay log 2 (path withheld)	3,995,499	Large file named for a relay listener; duration unverified

File size alone does not show how long a relay listener ran. Relay tools of this kind usually log the targeted enrollment URL, the account and template requested, and whether a certificate came back. Impacket can also print the certificate as Base64 PKCS#12. We did not publish or rely on the log content, and we cannot confirm any certificate was issued.

Beside the logs sit a 594-byte text file named for DCSync replication and a 19,353-byte compressed archive (paths withheld).

A headless browser and paired screenshots

The PNG images make up 55.6% (100 of 180) of the entries described in detail. The host also exposes a virtual X display: an 11-byte Xvfb display lock file, an X11 socket and two 54-byte `xvfb-run` authority files (paths withheld). Five HTML loop pages of about 26.7 KB each fit a headless browser harness.

Two kinds of image sit side by side:

Image group	Sizes	What names and sizes suggest
Automated login frames	9,539 bytes each	Repeated names and uniform size suggest login-screen captures (contents unverified)
Error stubs	212 bytes each	Tiny uniform files that suggest empty or failed captures
Console captures	342 KB to 1.03 MB	Larger PNG files named like desktop or terminal screenshots (contents unverified)
Paired before-and-after captures	56 KB to 85 KB	Pairs named for domain controller, command, Group Policy and identity checks

Target-named screenshot names are withheld, and so is the port number, because the host may itself belong to a victim. The paired naming fits someone documenting each step, which matters for the red-team reading in Chapter 6.

04

CHAPTER 04

Workbench B: A Modular Token-Gated PHP Kit

IN SHORT

A server in China repeats an eight-file PHP login and token core across three ports. The file names suggest the core guards a 509-byte command entry point and a SQL module, while 117-byte PHP stubs hide in font and JavaScript folders. Turn off PHP in static asset folders.

INSIDE THIS CHAPTER

The shared core 13

Sibling ports: stubs in asset folders 14

04 Workbench B: A Modular Token-Gated PHP Kit

Classic webshell scanners look for one large PHP file that holds a login form, a file manager and encoded `eval` or `passthru` calls. Going by the file names, this kit splits those jobs into small files. Login, token issue, token check, SQL access and command dispatch each live in a separate include. Each file stays small.

The server exposes 32 files on three non-standard ports (9, 10 and 13 files). They are 20 PHP files, 6 JavaScript files, 3 web fonts, 1 Base64 file, 1 text file and 1 file with no extension. We publish no PHP code bodies.

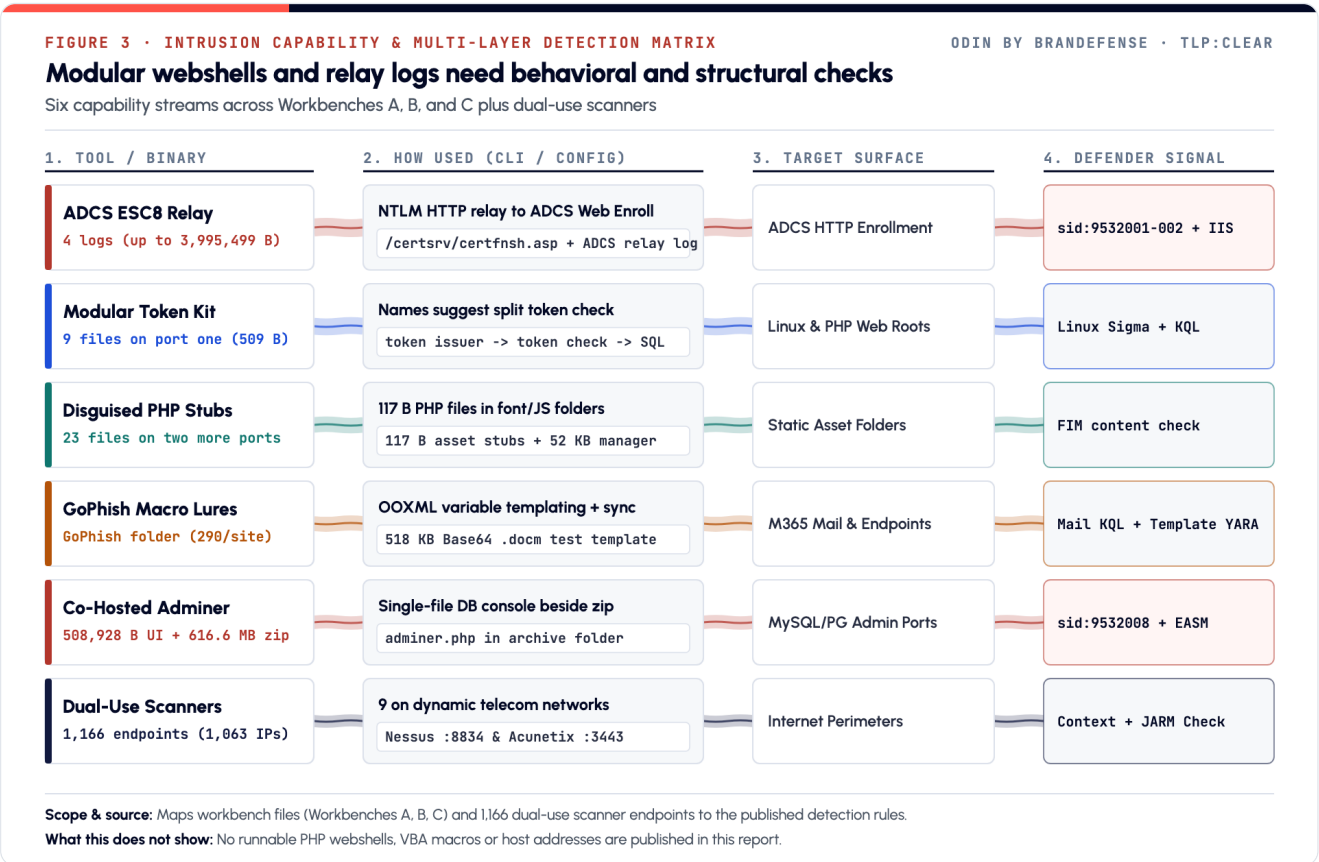


Figure 3: Modular webshells and relay logs need behavioral and structural checks

The shared core

All three ports carry the same eight-file shared library core (paths withheld), which accounts for 24 of the 32 files. The front end loads standard jQuery 3.5.1 (287,744 bytes) and a web font (119,221 bytes). A custom 372-byte JavaScript loader named like the jQuery library sits beside it; we did not see what it does.

We saw file names and sizes, not code. Going by the names, the server side runs through five small PHP files (T1505.003):

1. **Login and token issue.** A 2,662-byte PHP file is named as the operator login. An 816-byte PHP file is named as a token issuer, perhaps for a bearer token in the style of a JSON Web Token (JWT).
2. **Token check.** A 659-byte PHP file and a 267-byte PHP file are named as a token verifier and an include guard. The names suggest a token check before the command and SQL handlers.

3. **Command and SQL entry points.** On one port, the main command entry point is only 509 bytes, which fits login and interface code living in the shared PHP library folder. On all three ports, a 2,969-byte PHP file is named as a SQL database module (all paths withheld).

If the kit works as its names suggest, a scanner without a token would see nothing unusual.

Sibling ports: stubs in asset folders

The other two ports add fallback footholds next to the same core.

File (path withheld)	Port	Size (bytes)	Role
Modular PHP command entry point	Port 1	509	Command entry point, by name
PHP SQL database module	all three	2,969	SQL module, by name
PHP login handler	all three	2,662	Operator login, by name
PHP token issuer module	all three	816	Token issuer, by name
PHP token verifier module	all three	659	Token check, by name
PHP include guard module	all three	267	Shared include guard, by name
Custom JavaScript loader	all three	372	Custom script loader named like jQuery
Font-folder PHP stub	Port 3	117	PHP stub named like a Bootstrap font asset
Script-folder PHP stub	Port 3	117	PHP stub named like a jQuery asset
Full PHP web manager	Port 3	52,224	PHP web manager, going by name and size
Base64 staged archive	Port 3	23,552	Base64 staged archive in a subfolder
Minimal PHP stager	Port 2	28	Minimal PHP stager
Text marker	Port 2	20	Small marker file

Ports 1, 2 and 3 are the three non-standard ports, holding 9, 10 and 13 files. We withhold the port numbers because the host may itself be a victim.

A PHP file in a font or JavaScript asset folder should never exist. It works only when Nginx or Apache passes every `*.php` request to PHP-FPM, instead of a short list of application entry points. On such a server, a 117-byte PHP file blends into the static theme tree.

CHAPTER 05

Workbench C: GoPhish Macro Templates Beside Adminer

IN SHORT

A server in India mirrors 290 files across two web listeners. A GoPhish source tree with macro-enabled Word templates and a sync-named Python service sits beside an Adminer console and a 616.6 MB archive. Block internet macros and remove exposed Adminer copies.

INSIDE THIS CHAPTER

A GoPhish tree and a sync-named Python service 16

Macro-enabled Word templates 16

The archive and the Adminer console 17

05 Workbench C: GoPhish Macro Templates Beside Adminer

Phishing infrastructure often mixes open-source automation with custom tools for importing recipients. On this server, two web listeners serve the same 290-file web root, 580 entries in total. Of 360 entries described in detail, Go source files (158), HTML templates (36), CSS (28), Python scripts (22), Markdown (18) and Base64 fixtures (16) dominate.



Figure 4: Open listeners expose relay logs, three-port PHP kits and GoPhish

A GoPhish tree and a sync-named Python service

The tree sits in a folder whose name is withheld. The public GoPhish releases page lists v0.12.1 as the latest release, so the folder name does not match a public release. It may be a renamed or modified copy. Beside the Go source sits a custom Python orchestrator in its own Docker container: a Dockerfile (526 bytes) and three Python modules of 4,288 to 7,260 bytes.

A custom Python module is named for directory sync; its behaviour was not verified. If paired with GoPhish's REST API for campaigns and imports, such a service could automate recipient groups and campaign scheduling.

Macro-enabled Word templates

The GoPhish test-data folder holds eight Base64 OpenXML fixtures, 16 across both ports. GoPhish's 5,072-byte attachment handler decodes a Base64 `.docm` file and opens the archive in memory.

It then fills per-recipient variables such as `{{.FirstName}}`, `{{.RId}}` and `{{.URL}}` inside the XML parts. It keeps the macro project `word/vbaProject.bin` as it is. Each recipient gets a file with a different hash, which defeats simple hash blocking (T1566.001).

File (path withheld)	Size (bytes)	Role
Base64 `.docx` test fixture	599,737	Upstream GoPhish Base64 `.docx` test fixture
Base64 `.docm` variable template	517,989	Upstream GoPhish Base64 macro-enabled `.docm` template
Rendered Base64 `.docm` fixture	183,256	The same template after variable substitution
GoPhish attachment handler	5,072	GoPhish code that fills variables into Office files
GeoLite2 City database	38,708,418	MaxMind database bundled for click geolocation

A note of caution: the public GoPhish project ships the 517,989-byte `.docm` Base64 template as a unit-test file. Its presence shows that the tree can template `.docm` lures. It does not show custom VBA work or that any lure was sent.

The archive and the Adminer console

The tree shares a web root with a second folder. That folder appears to be an unpacked copy of an organisation's web archive; its name is withheld.

File	Size (bytes)	Role
Archive ZIP at the web root (name withheld)	616,562,688	A 616.6 MB web or organisation archive
`adminer.php` (folder path withheld)	508,928	Single-file console for MySQL, PostgreSQL, SQLite and MS SQL
PHP dependency lock file (path withheld)	265,318	Dependency list inside the unpacked archive
Backup HTML page (path withheld)	10,671	Backup page at the web root

Adminer is a legitimate tool. An exposed copy next to a large archive and a phishing tree is still a risk, whoever owns the server.

06

CHAPTER 06

Other Explanations

IN SHORT

Each surface has a plausible benign or authorised reading. Defensive actions stay the same whichever is true, so fix ADCS, PHP and macro settings without waiting for attribution.

06 Other Explanations

1. **Authorised scanning.** Organisations run Nessus on 8834, Acunetix on 3443, OpenVAS on 443 and OWASP ZAP on 8080 to 8091 across AWS, Azure, OVH and Hetzner. Security engineers also test from home and mobile telecom networks. A scanner banner alone says nothing about intent.
2. **A red-team server on Workbench A.** Before-and-after screenshot pairs next to four Certipy- and ADCS-named relay logs could be a penetration-testing team keeping report evidence. Even so, relay logs on open HTTP put credentials at risk, and the ESC8 fix still applies.
3. **A custom admin tool on Workbench B.** On their own, the login, token-issuer, token-verifier and SQL modules could be a small custom session library. Next to a 509-byte command entry point, a 52,224-byte web manager, a 28-byte stager, a 23,552-byte Base64 file and PHP stubs in font and script folders, webshell staging is the better reading. The server's owner is not established and may itself be a victim.
4. **Awareness training on Workbench C.** A GoPhish tree with a sync-named Python module could belong to a security awareness provider that also hosts Adminer and a backup ZIP. It could also be an operator staging a stolen archive beside GoPhish. Behaviour-based detection works in both cases.

For these reasons, all three host addresses, host-specific file paths and file hashes are withheld.

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

Active Directory and PKI

- [ ] List every ADCS server. Remove the HTTP Web Enrollment role (`/certsrv/`) where it is not needed.
- [ ] Where web enrollment stays, require HTTPS and turn on Extended Protection for Authentication, as KB5005413 describes. Enforce SMB and LDAP signing.
- [ ] Run Suricata SID 9532002 in alert mode first, to find clients that still use NTLM over HTTP before you turn HTTP off.

Linux and web platforms

- [ ] Send only approved front-controller paths to PHP-FPM or FastCGI. Turn off PHP in static font, JavaScript, image and upload folders.
- [ ] Alert when a PHP or web server worker starts a shell, `id`, `whoami` or a database client.
- [ ] Remove unapproved `adminer.php` and phpMyAdmin copies, backup `.zip` and `.bak` files, and open directory listings from public and staging servers.

Messaging and endpoints

- [ ] Keep Microsoft's default that blocks macros in files from the internet. Enforce it by Group Policy or Intune.
- [ ] Quarantine inbound `.docm` attachments from unapproved external domains. Alert when Word starts PowerShell, `cmd.exe` or a script host.

Detection and CTI

- [ ] Deploy the rules in Chapter 8. Fill the named allowlists with your PKI proxies, admin hosts and phishing-simulation senders first.
- [ ] Keep dual-use scanner feeds as enrichment with a short expiry. Never send them to a firewall block list.

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 23
How the other rules work 23
MITRE ATT&CK mapping 24

08 Detection Rules

The full rules are in Appendix B. The Suricata file passes a syntax lint. We could not load the rules in live Suricata, YARA, Sigma or Sentinel engines for this edition, and they were not replayed against live traffic. Test and tune them before production use.

Rule file	What it catches
`odin_adcs_esc8_webshell_and_macro_gophish.rules`	Suricata 7: three signatures for ADCS web enrollment POSTs, NTLM over HTTP to `/certsrv/`, and exposed Adminer logins
`sentinel_adcs_esc8_relay_jwt_webshell_and_gophish.kql`	Microsoft Sentinel and Defender XDR: web or Word processes starting shells and requests to `/certsrv/certfnsh.asp` or `adminer.php`, enriched with inbound `.docm` mail to the same user
`sigma_adcs_certsrv_relay_jwt_webshell_and_docm.yml`	Sigma (3 rules): ADCS enrollment POSTs in IIS logs, PHP workers starting shells on Linux, and Word `.docm` files starting script hosts on Windows
`odin_adcs_relay_logs_webshell_kit_and_gophish_docm.yar`	YARA (2 rules): ADCS ESC8 relay logs and Base64 macro-enabled `.docm` template files

How the Suricata signatures work

SID	Looks for	Why
9532001	Internal `POST /certsrv/certfnsh.asp` with `Mode=newreq` and `CertRequest=` in the body	Hunt signal: a web enrollment certificate request to review for ESC8 relay
9532002	`Authorization: NTLM` on any `/certsrv/` request	Hunt signal: NTLM over plain HTTP to ADCS, the precondition for ESC8
9532008	`POST` to `adminer.php` with an `auth[driver]=` field	A login to an exposed Adminer console

SIDs 9532001 and 9532002 need visibility of plain HTTP on the ADCS network segment.

How the other rules work

The KQL query looks back seven days and builds three views. The mail view finds inbound `.docm` attachments, excludes approved simulation domains and senders, and joins the results to a matching device user.

The process view finds web workers (`php-fpm`, `php-cgi`, `apache2`, `httpd`, `w3wp.exe`) or Word starting shells and script hosts. It also finds commands that name `certfnsh.asp` or Certipy.

The network view finds requests to `/certsrv/certfnsh.asp` or `adminer.php`. Defender often records only the host name in `RemoteUrl`, so path matches require full URLs; use proxy or IIS logs as well.

Each match gets one of four `INVESTIGATE_*` or `HUNT_*` labels: an ADCS relay source, a token webshell host, a `.docm` macro run, or a hunt lead. A label starts an investigation; it is not a verdict on its own. Approved PKI and admin hosts go in a named allowlist. There are no wildcards.

The first Sigma rule is a hunt signal that reads IIS logs on ADCS servers. It flags successful `POST` requests to `/certsrv/certfnsh.asp` for relay review, except from two exact PKI proxy addresses and one enrollment service account.

The second flags PHP or web workers on Linux that start shells, `id`, `whoami`, `uname` or database clients. The third flags Word opening a `.docm` file and starting PowerShell, `cmd.exe`, `mshta.exe`, `rundll32.exe` or a script host. Each filter names exact hosts or accounts.

YARA rule	Fires when
`ODIN_ADCS_ESC8_NTLM_Relay_Capture_Log`	A file under 8 MB holds Certipy relay output (`/certsrv/certfnsh.asp`, `(ESC8)` and a certificate request or write line), or Impacket output (`Generating CSR` and `GOT CERTIFICATE! ID` or a PKCS#12 line)
`ODIN_GoPhish_Base64_Docm_Variable_Template_Artifact`	A file between 100 KB and 2 MB starts with a Base64 ZIP header and holds the Base64 form of the macro project name `word/vbaProject.bin`, as in a `.docm.b64` template. Recipient variables sit in compressed XML, so the rule does not look for them

MITRE ATT&CK mapping

Technique	How it shows up	Detection
T1557.001 Adversary-in-the-Middle: Name Resolution Poisoning and SMB Relay	NTLM relay to ADCS web enrollment	SIDs 9532001 and 9532002; Sigma rule 1
T1649 Steal or Forge Authentication Certificates	Certificates issued for the relayed account	SID 9532001; KQL; YARA relay log rule
T1003.006 OS Credential Dumping: DCSync	A staged text file named for replication beside the relay logs	Directory replication auditing (no rule here)
T1505.003 Server Software Component: Web Shell	The modular PHP kit, the 52 KB web manager and asset-folder PHP stubs	KQL; Sigma rule 2
T1566.001 Phishing: Spearphishing Attachment	GoPhish `.docm` templates with recipient variables	KQL; Sigma rule 3; YARA template rule

09

CHAPTER 09

Indicators

IN SHORT

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

09 Indicators

Appendix A holds no indicators for this report: 0 addresses and 0 file hashes. No host in this study is confirmed adversary infrastructure. The scanners may be legitimate, and each workbench has a plausible benign or victim reading (Chapter 6).

The patterns below appear in this report only. Use them for hunting, not for blocking by themselves.

Pattern	Type	Use
`/certsrv/certfnsh.asp`	Standard ADCS enrollment path	Hunt in IIS logs and internal network traffic for potential relay requests
Certipy and Impacket ADCS relay log strings	Relay console output patterns	Hunt on file servers with the YARA relay log rule (paths withheld)
PHP files in font, JavaScript or static asset folders	Asset-folder PHP stub pattern	Hunt in web roots and disable PHP execution in static asset folders (paths withheld)
Base64-encoded `.docm` archives with `word/vbaProject.bin`	Macro template archive pattern	Hunt on staging servers with the YARA template rule (paths withheld)
`adminer.php`	Single-file database console	Hunt at the WAF and in web roots for unapproved database consoles
Three JARM values in Chapter 2	TLS fingerprints of scanner products	Enrichment only, never block

Withheld: all three workbench host addresses, all scanner host addresses and the 14 file hashes recorded for the workbenches. The workbench port numbers and all host-specific file and folder names are also withheld.

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

- **Capture date.** These findings reflect directory listings recorded on 20 November 2026.
- **Files, not outcomes.** We looked at directory listings, file names and sizes. We ran no files and contacted no system named in them. Nothing we saw confirms that a relay issued a certificate, that the webshells reached a victim, or that any lure was sent.
- **Three hosts, no link.** The three workbenches share no address, certificate or file. The four-stage picture is an analytical model, not one actor.
- **Owners are not established.** Each workbench may belong to an attacker, an authorised tester, a training provider or a victim. For that reason no address, host-specific path or hash is published.
- **Scanners are dual-use.** The 1,166 endpoints run widely sold or open-source products. Their presence says nothing about intent.
- **GoPhish fixtures are public.** The `.docm` template is a GoPhish unit-test file. It shows capability, not a campaign.
- **No CVE.** ESC8 is a configuration weakness, not a single software flaw, so this report cites no CVE.
- **Rule testing.** The Suricata file passes a syntax lint. The rules were not loaded in live engines or replayed against live traffic for this edition.

11

CHAPTER 11

Sources

IN SHORT

This chapter lists the public references used in the report.

11 Sources

1. Microsoft Support. KB5005413: Mitigating NTLM Relay Attacks on Active Directory Certificate Services (AD CS). https://support.microsoft.com/en-us/topic/kb5005413-mitigating-ntlm-relay-attacks-on-active-directory-certificate-services-ad-cs-3612b773-4043-4aa9-b23d-b87910cd3429 (verified 20 November 2026)
2. MITRE ATT&CK. T1649 Steal or Forge Authentication Certificates. https://attack.mitre.org/techniques/T1649/ (verified 20 November 2026)
3. MITRE ATT&CK. T1557.001 Adversary-in-the-Middle: Name Resolution Poisoning and SMB Relay. https://attack.mitre.org/techniques/T1557/001/ (verified 20 November 2026)
4. MITRE ATT&CK. T1003.006 OS Credential Dumping: DCSync. https://attack.mitre.org/techniques/T1003/006/ (verified 20 November 2026)
5. IETF. RFC 7519, JSON Web Token (JWT). https://datatracker.ietf.org/doc/html/rfc7519 (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. OWASP Cheat Sheet Series. File Upload Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html (verified 20 November 2026)
8. MITRE ATT&CK. T1566.001 Phishing: Spearphishing Attachment. https://attack.mitre.org/techniques/T1566/001/ (verified 20 November 2026)
9. GoPhish documentation. Templates. https://docs.getgophish.com/user-guide/documentation/templates (verified 20 November 2026)
10. GoPhish documentation. Template Reference. https://docs.getgophish.com/user-guide/template-reference (verified 20 November 2026)
11. GoPhish on GitHub. Releases. https://github.com/gophish/gophish/releases (verified 20 November 2026)
12. Microsoft Learn. Macros from the internet are blocked by default in Office. https://learn.microsoft.com/en-us/microsoft-365-apps/security/internet-macros-blocked (verified 20 November 2026)

APPENDICES

Indicators and Detection Rules

FOR DEFENDERS

No individual indicators are published for this study; Appendix A explains why. Appendix B carries every published detection rule in full.

INSIDE

A. Indicators of compromise (0) 32

B. Detection rules (4 files) 33

A Indicators of Compromise

No individual addresses or file hashes are published for this study. The hosts in this report are exposed or unattributed systems rather than confirmed adversary infrastructure, so they stay anonymous. Use the detection rules in Appendix B.

B Detection Rules

Each rule file loads in its own engine (Suricata 7, YARA-X, 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_adcs_esc8_webshell_and_macro_gophish.rules

SURICATA

```
# Odin by Brandefense - Monograph 32 / 35 Suricata 7.x Detection Suite
# Coverage: ADCS /certsrv/certfnsh.asp enrollment POSTs for relay review, unencrypted
#           HTTP NTLM authentication to /certsrv/, and exposed Adminer authentication POSTs
# Classification: TLP:CLEAR

alert http any any -> $HOME_NET any (msg:"ODIN [R32] ADCS Web Enrollment POST: Review for Relay";
flow:established,to_server; http.method; content:"POST"; http.uri; bsize:>=21; content:"/certsrv/certfnsh.asp";
nocase; fast_pattern; http.request_body; bsize:>24; content:"Mode=newreq"; nocase; content:"CertRequest=";
nocase; classtype:attempted-admin; sid:9532001; rev:2; metadata:created_at 2026_11_20, deployment
internal_adcs_sensor, tactic credential_access;)

alert http any any -> $HOME_NET any (msg:"ODIN [R32] HUNT Unencrypted HTTP NTLM Authentication to ADCS /
certsrv/"; flow:established,to_server; http.uri; bsize:>=9; content:"/certsrv/"; nocase; fast_pattern;
http.header; content:"Authorization: NTLM TlRMTVNTUA"; classtype:policy-violation; sid:9532002; rev:2;
metadata:created_at 2026_11_20, deployment internal_adcs_sensor, policy kb5005413_epa_audit;)

alert http any any -> $HOME_NET any (msg:"ODIN [R32] Exposed Single-File Adminer Database Console Authentication
POST"; flow:established,to_server; http.method; content:"POST"; http.uri; bsize:>=12; content:"adminer.php";
nocase; fast_pattern; http.request_body; bsize:>12; content:"auth"; pcre:"/auth(\[|%5[Bb])driver(\]|%5[Dd])=/i";
classtype:web-application-attack; sid:9532008; rev:1; metadata:created_at 2026_11_20, deployment
perimeter_and_waf, tactic discovery;)
```

odin_adcs_relay_logs_webshell_kit_and_gophish_docm.yar

YARA

```
/*
   Odin by Brandefense - Monograph 32 / 35 YARA Artifact Suite
   Scope: Bounded structural detection rules for ADCS ESC8 NTLM relay console logs
          (Certipy and Impacket ntlmrelayx output) and Base64-encoded OpenXML .docm
          macro templates.
   Classification: TLP:CLEAR
*/

rule ODIN_ADCS_ESC8_NTLM_Relay_Capture_Log
{
    meta:
        author = "Odin by Brandefense"
        monograph = "32 / 35"
        classification = "TLP:CLEAR"
        disposition = "HUNT_ADCS_ESC8_RELAY_LOG"
        description = "Matches Certipy relay (ESC8) or Impacket ntlmrelayx ADCS attack console logs, using
strings these tools print"
        note = "Neither tool logs the POST body (Mode=newreq / CertRequest=); Impacket does not print the
certfnsh.asp URL"
    strings:
        // Certipy relay output
        $u1 = "/certsrv/certfnsh.asp" ascii nocase
        $c_t = "(ESC8)" ascii
        $c_r = "Requesting certificate for" ascii
        $c_w = "Wrote certificate and private key" ascii
        // Impacket ntlmrelayx adcsattack output
        $i_g = "Generating CSR" ascii
        $i_ok = "GOT CERTIFICATE! ID" ascii
        $i_p = "Base64-encoded PKCS#12 certificate" ascii
        $i_w = "Writing PKCS#12 certificate" ascii
    condition:
        filesize < 8MB and (($u1 and $c_t and 1 of ($c_r, $c_w)) or ($i_g and 1 of ($i_ok, $i_p, $i_w)))
```

```
}

rule ODIN_GoPhish_Base64_Docm_Variable_Template_Artifact
{
    meta:
        author = "Odin by Brandefense"
        monograph = "32 / 35"
        classification = "TLP:CLEAR"
        disposition = "HUNT_GOPHISH_DOCM_TEMPLATE"
        description = "Matches Base64-encoded macro-enabled OpenXML files (such as GoPhish .docm.b64 templates):
a Base64 ZIP header at offset 0 and the Base64 form of the uncompressed ZIP entry name word/vbaProject.bin at any
of its three alignments"
        note = "Recipient variables sit inside deflated XML parts, so they are not matched; single-line Base64
only"
    strings:
        $pk_b64 = "UEsDB" ascii
        // "word/vbaProject.bin" Base64-encoded at byte offsets 0, 1 and 2
        $vba0 = "d29yZC92YmFQcm9qZWN0LmJpb" ascii
        $vba1 = "dvcmQvdmJhUHJvamVjdC5iaW" ascii
        $vba2 = "3b3JkL3ZiYVByb2plY3QuYmlu" ascii
    condition:
        filesize > 100KB and filesize < 2MB and $pk_b64 at 0 and 1 of ($vba*)
}
```

sentinel_adcs_esc8_relay_jwt_webshell_and_gophish.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Monograph 32 / 35 Microsoft Sentinel & Defender XDR Hunt
// Correlates inbound macro-enabled .docm phishing deliveries (EmailEvents + EmailAttachmentInfo),
// non-PKI ADCS /certsrv/certfnsh.asp HTTP enrollment requests, exposed adminer.php
// web requests (DeviceNetworkEvents), and Office or PHP worker child shell spawns (DeviceProcessEvents)
// Classification: TLP:CLEAR

let Lookback = 7d;
let ApprovedPhishSimDomains = dynamic(["phishsim-training.internal.corp"]);
let ApprovedPhishSimSenders = dynamic(["awareness-sim@phishsim-training.internal.corp"]);
let ApprovedPkiAndAdminHosts = dynamic(["PKI-ENROLL-PROXY-01", "DB-ADMIN-BASTION-01", "SEC-VALIDATION-01"]);
let WebAndOfficeParents = dynamic(["php-fpm", "php-cgi", "apache2", "httpd", "w3wp.exe", "WINWORD.EXE"]);
let SuspiciousChildShells = dynamic(["sh", "bash", "dash", "id", "whoami", "cmd.exe", "powershell.exe",
"pwsh.exe", "mshta.exe", "wscript.exe", "cscript.exe"]);
let InboundDocmEmails = EmailEvents
    | where Timestamp >= ago(Lookback)
    | where EmailDirection == "Inbound"
    | where not(SenderFromDomain in~ (ApprovedPhishSimDomains)) and not(SenderFromAddress in~
(ApprovedPhishSimSenders))
    | join kind=inner (
        EmailAttachmentInfo
        | where Timestamp >= ago(Lookback)
        | where FileName endswith ".docm" or FileType has "docm"
    ) on NetworkMessageId
    | summarize
        MailFirstSeen = min(Timestamp),
        MailLastSeen = max(Timestamp),
        DocmFiles = make_set(FileName, 8),
        Senders = make_set(SenderFromAddress, 8),
        SenderDomains = make_set(SenderFromDomain, 8),
        MailCount = count()
        by RecipientEmailAddress;
let HostChildSpawns = DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedPkiAndAdminHosts))
    | where (InitiatingProcessFileName in~ (WebAndOfficeParents) and FileName in~ (SuspiciousChildShells))
        or ProcessCommandLine has_any ("certfnsh.asp", "certipy")
    | summarize
        ProcFirstSeen = min(Timestamp),
        ProcLastSeen = max(Timestamp),
        SampleCommands = make_set(ProcessCommandLine, 8),
        SpawnCount = count()
        by DeviceId, DeviceName, AccountUpn, InitiatingProcessFileName;
```

```
let SuspiciousWireEvents = DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where not(DeviceName in~ (ApprovedPkiAndAdminHosts))
    | where RemoteUrl has_any ("/certsrv/certfnsh.asp", "adminer.php")
    | summarize
        NetFirstSeen = min(Timestamp),
        NetLastSeen = max(Timestamp),
        MatchedUrls = make_set(RemoteUrl, 10),
        NetHits = count()
        by DeviceId, DeviceName, InitiatingProcessFileName;
HostChildSpawns
| join kind=fullouter (SuspiciousWireEvents) on DeviceId, DeviceName
| extend ResolvedDevice = coalesce(DeviceName, DeviceName1), NormUpn = tolower(AccountUpn)
| join kind=leftouter (
    InboundDocmEmails
    | extend NormRecipient = tolower(RecipientEmailAddress)
) on $left.NormUpn == $right.NormRecipient
| extend AlertDisposition = case(
    MatchedUrls has "/certsrv/certfnsh.asp" or SampleCommands has_any ("certfnsh.asp", "certipy"),
"INVESTIGATE_ADCS_ESC8_HTTP_RELAY_SOURCE",
    InitiatingProcessFileName in~ ("php-fpm", "php-cgi", "apache2", "httpd"),
"INVESTIGATE_MODULAR_TOKEN_WEBSHELL_HOST",
    InitiatingProcessFileName =~ "WINWORD.EXE" and array_length(Senders) > 0,
"INVESTIGATE_GOPHISH_DOCM_MACRO_EXECUTION",
    "HUNT_EXPOSED_ADMINER_OR_UNCORRELATED_SPAWN"
)
// Labels are single-signal leads for analyst triage, not automatic isolation verdicts.
| project ProcFirstSeen, NetFirstSeen, MailFirstSeen, ResolvedDevice, AlertDisposition,
InitiatingProcessFileName, SampleCommands, MatchedUrls, DocmFiles, Senders, SenderDomains, SpawnCount, NetHits
| sort by coalesce(ProcFirstSeen, NetFirstSeen, MailFirstSeen) desc
```

sigma_adcs_certsrv_relay_jwt_webshell_and_docm.yml

SIGMA

```
title: ADCS Web Enrollment POST for Relay Review (/certsrv/certfnsh.asp)
id: 4d91a8c3-7b12-4e58-9c31-202611203201
status: experimental
description: Detects HTTP POST enrollment requests to Active Directory Certificate Services (/certsrv/
certfnsh.asp) originating from non-approved hosts as a hunt signal to review for potential ESC8 NTLM relay
activity (such as Certipy or ntlmrelayx).
author: Odin by Brandefense
date: 2026/11/20
tags:
  - attack.credential-access
  - attack.t1649
logsource:
  category: webserver
  product: windows
detection:
  selection_certsrv:
    cs-method: 'POST'
    cs-uri-stem|contains: '/certsrv/certfnsh.asp'
    sc-status: 200
  filter_pki_proxies:
    c-ip:
      - '10.20.10.15'
      - '10.20.10.16'
  filter_enroll_service_accounts:
    cs-username:
      - 'CORP\svc-pki-autoenroll'
  condition: selection_certsrv and not 1 of filter_*
falsepositives:
  - Legacy manual browser certificate enrollment on ADCS servers where HTTPS and Extended Protection for
Authentication (KB5005413) have not yet been enforced
level: critical
---
title: Modular Token-Gated PHP Webshell Worker Spawning Shell or Database Export Utility
id: 8b24e5d9-1a63-4f74-b082-202611203202
status: experimental
```

```
description: Identifies PHP-FPM or Linux web server workers spawning OS shells, reconnaissance commands, or
database client utilities consistent with webshell activity or database misuse.
author: Odin by Brandefense
date: 2026/11/20
tags:
  - attack.persistence
  - attack.t1505.003
logsource:
  category: process_creation
  product: linux
detection:
  selection_web_parent:
    ParentImage|endswith:
      - '/php-fpm'
      - '/php-fpm8.2'
      - '/php-fpm8.3'
      - '/php-cgi'
      - '/apache2'
      - '/httpd'
  selection_shell_or_db_child:
    Image|endswith:
      - '/sh'
      - '/bash'
      - '/dash'
      - '/id'
      - '/whoami'
      - '/uname'
      - '/mysql'
      - '/mysqldump'
      - '/psql'
  filter_validation_hosts:
    ComputerName:
      - 'SEC-VALIDATION-WEB-01'
  filter_backup_cron_users:
    User:
      - 'svc-db-backup'
  condition: selection_web_parent and selection_shell_or_db_child and not 1 of filter_*
falsepositives:
  - Legacy PHP backup plugins invoking mysqldump directly from web worker pools instead of isolated cron jobs
level: critical
---
title: Microsoft Word Macro-Enabled Document (.docm) Spawning Script or LOLBin Child Process
id: 2f67c9a1-5d40-4c89-a416-202611203203
status: experimental
description: Detects WINWORD.EXE opened with a macro-enabled OpenXML document (.docm) spawning PowerShell,
command shells, or Windows script hosts consistent with templated GoPhish macro attachment lures.
author: Odin by Brandefense
date: 2026/11/20
tags:
  - attack.initial-access
  - attack.t1566.001
logsource:
  category: process_creation
  product: windows
detection:
  selection_word_docm:
    ParentImage|endswith: '\WINWORD.EXE'
    ParentCommandLine|contains: '.docm'
  selection_child_interpreter:
    Image|endswith:
      - '\powershell.exe'
      - '\pwsh.exe'
      - '\cmd.exe'
      - '\wscript.exe'
      - '\cscript.exe'
      - '\mshta.exe'
      - '\rundll32.exe'
  filter_finance_macro_hosts:
    ComputerName:
```

```
        - 'FIN-LEGACY-MACRO-01'
    condition: selection_word_docm and selection_child_interpreter and not 1 of filter_*
falsepositives:
    - Approved internal finance workstations running legacy signed VBA templates that invoke local batch helpers
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/adcs-relays-token-gated-webshells-and-macro-phishing/
Short Read · brandefense.io/odin-research/blog/adcs-relay-logs-hidden-php-kits-and-gophish-macros/
Defense Kit · brandefense.io/odin-research/defense-kit/