ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 05 / 30 · ENTERPRISE EXPLOITATION

Enterprise Middleware Under Attack

ERP Directives, Cloud Director, and Workflow Engines

How staged toolkits target Epicor Kinetic, SQL Server Agent, VMware Cloud Director and low-code workflow engines

Prepared November 2026

Full Report · Public edition · Version 1.1

713
Unique file paths across 8 IP addresses (7 HTTP hosts and 1 configured relay), in 2,346 directory entries

2,176
Entries on the mirrored Epicor and SQL Server cluster: 544 paths on each of 4 servers

76
SQL Server Agent job, dropper and diagnostic entries that point to the relay (19 per server)

65
Files on the dual-use VMware VCDA and VCD workbench

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

How staged toolkits target Epicor Kinetic, SQL Server Agent, VMware Cloud Director and low-code workflow engines

01	Executive Summary	04
02	The Middleware Toolkits at a Glance	07
03	Epicor Kinetic Directive Injection and SQL Agent Persistence	10
04	VMware VCDA 4.7.4 and VCD 10.6.0 Workbench	14
05	Workflow Engines: n8n and Apache Activiti	17
06	Telling Abuse from Normal Admin Work	21
07	What Defenders Should Do	24
08	Detection Rules	26
09	Indicators	30
10	Scope and Limits	33
11	Sources	35
A	Indicators of compromise (20)	37
B	Detection rules (4 files)	38

IN SHORT

Odin by Brandefense found 713 unique attacker file paths on eight IP addresses, most seen on 13 November 2026. The toolkits target middleware that workstation endpoint tools rarely watch: Epicor Kinetic business-rule tables, SQL Server Agent jobs, VMware Cloud Director Availability appliances, and the n8n and Apache Activiti workflow engines. A mirrored four-server cluster holds most of the files. Audit `Ice.BpDirective` writes, restrict SQL Agent `CmdExec` steps, keep VCDA management ports off the internet, and treat workflow imports as code deployments.

01

CHAPTER 01

Executive Summary

IN SHORT

Odin by Brandefense found 713 unique attacker file paths on eight IP addresses, most seen on 13 November 2026. The toolkits target middleware that workstation endpoint tools rarely watch: Epicor Kinetic business-rule tables, SQL Server Agent jobs, VMware Cloud Director Availability appliances, and the n8n and Apache Activiti workflow engines. A mirrored four-server cluster holds most of the files. Audit `Ice.BpDirective` writes, restrict SQL Agent `CmdExec` steps, keep VCDA management ports off the internet, and treat workflow imports as code deployments.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 05

01 Executive Summary

Enterprise monitoring often centres on laptops and servers, not on ERP database tables, replication appliances or workflow engines. These platforms compile custom code and hold service credentials by design. That makes them attractive to attackers, and their abuse can look like routine admin work. This report shows what the staged toolkits contain, how they are meant to work, and how defenders can tell abuse from normal operations.

1. **A mirrored four-server cluster stages Epicor directive injection and SQL Agent persistence.**
   Four adjacent servers in 43.230.171[.]144/29 carry the same 544-path directory tree. The files inject custom code into the Epicor Kinetic directive table and set up SQL Server Agent jobs that call a relay at 43.230.171[.]146. The same tree holds Kerberos, AS-REP roasting and Windows privilege-escalation tools.

2. **A dual-use workbench co-stages VMware VCDA probes, Jetty smuggling and keystore decryption.**
   A 65-file workbench targets VMware Cloud Director Availability (VCDA) 4.7.4 and VMware Cloud Director (VCD) 10.6.0 lab appliances. It holds path-traversal and root-reset probes, Eclipse Jetty request-smuggling tests (CVE-2026-2332) and an offline keystore decryptor. The tracks target separate lab systems. Nothing in the files links them into one end-to-end chain.

3. **Workflow engine toolkits aim n8n and Activiti scripting at operating-system commands.**
   On a third-party n8n server, an inferred attacker working directory holds 24 files for session forgery, Code Node injection, a PostgreSQL webshell and PwnKit escalation. On a dedicated Apache Activiti workbench, 81 files build BPMN 2.0 models written to evade web application firewalls (WAFs) and to call Groovy. Operating-system command execution on any target is not shown.

Three Decisions for Leaders

1. **Audit ERP directives and lock down SQL Server Agent.** Alert on Epicor directives that combine database-layer origin or Business Object update hooks with operating-system code. Limit SQL Agent `CmdExec` and PowerShell steps to least-privilege proxies, require signed scripts, and disable ad-hoc linked servers.
2. **Isolate VCDA management interfaces and rotate keystore secrets.** Keep VCDA ports 8443, 8046 and 3030 on management networks only. Apply Broadcom and Jetty updates, and rotate Cloud Director encryption keys whenever pre-authentication exposure is suspected.
3. **Treat workflow and BPMN imports as privileged code deployments.** Run n8n Code Nodes in sandboxed task runners without access to environment variables. Disable unrestricted Groovy in Apache Activiti. Keep adversary block lists separate from research and incident-response watchlists.

The Study in Numbers

Number	What it is
713	Unique file paths across 8 IP addresses (7 HTTP hosts and 1 configured relay), in 2,346 directory entries
2,176	Entries on the mirrored Epicor and SQL Server cluster: 544 paths on each of 4 servers
76	SQL Server Agent job, dropper and diagnostic entries that point to the relay (19 per server)
65	Files on the dual-use VMware VCDA and VCD workbench
81	Files on the Apache Activiti workbench at 106.75.249[.]126:8081
24	Files in an inferred attacker working directory on a third-party n8n server
20	Published indicators: 6 adversary IP addresses and 14 SHA-256 file hashes

How to use this report.

Security leaders and architects can read Chapter 1, Chapter 6 and Chapter 7. ERP, database and virtualization engineers should read Chapters 3 and 4 and work through the checklist in Chapter 7. Detection engineers and threat hunters should start with Chapters 5, 8 and 9.

02

CHAPTER 02

The Middleware Toolkits at a Glance

IN SHORT

Eight IP addresses hold 2,346 directory entries and 713 unique file paths. Before using any indicator, defenders should know which hosts are adversary staging, which is a dual-use research workbench, and which is a likely victim.

INSIDE THIS CHAPTER

Four environments, four roles 08

What the files are for 08

02 The Middleware Toolkits at a Glance

The eight addresses are seven HTTP servers and one TCP relay. The relay, 43.230.171[.]146, appears in attacker scripts on ports 9001 and 8080. It was not listed as a file server itself.

The four-server ERP mirror repeats the same 544 paths on each node, giving 2,176 entries. Counting each environment once gives 714 paths (544 + 81 + 65 + 24). One short `.txt` file name appears both on the ERP mirror and in the n8n working directory, so the global count of distinct paths is 713.

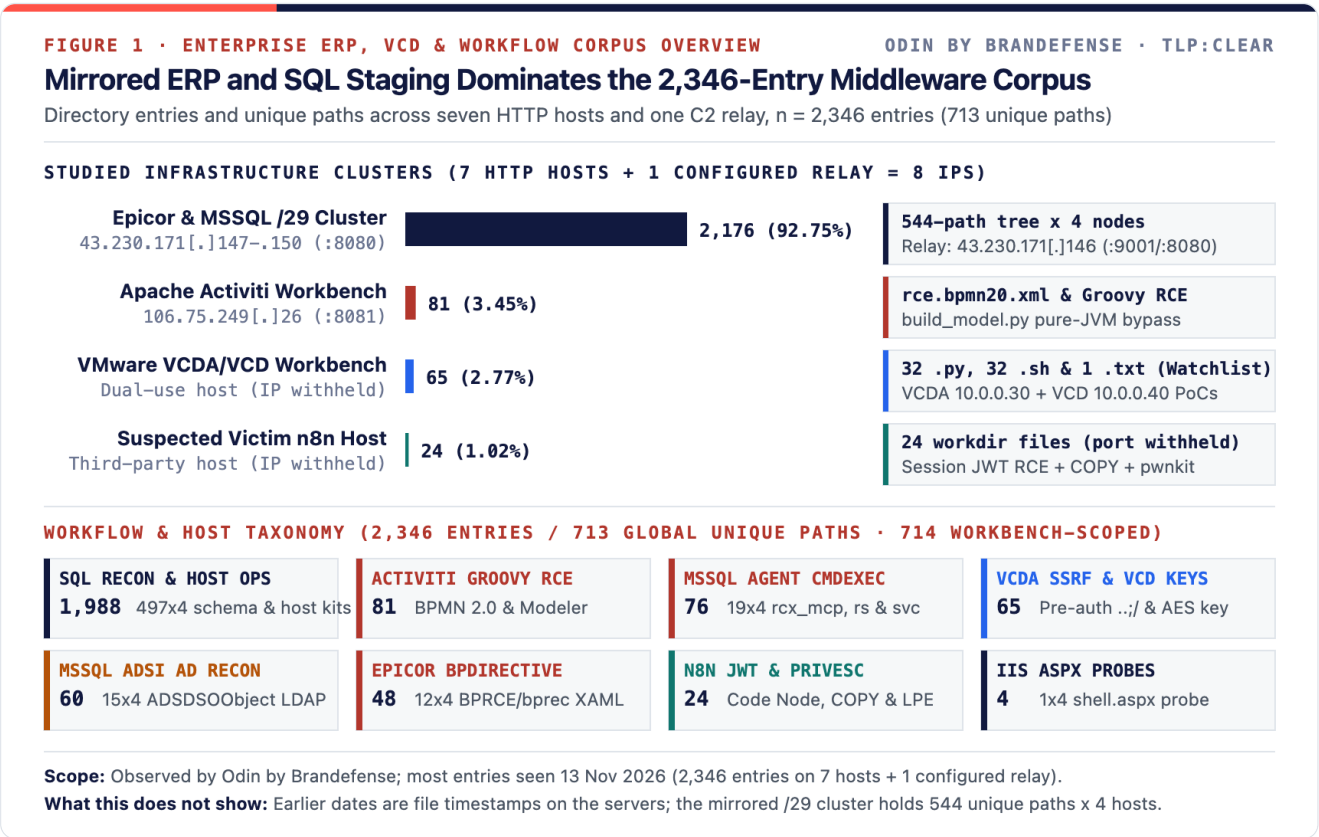


Figure 1: Mirrored ERP and SQL staging dominates the 2,346-entry middleware corpus

Four environments, four roles

Environment	Role	Address (port)	Entries	Target stack	Main capability
Epicor and SQL Server cluster	Adversary staging plus relay	43.230.171[.]147 to .150 (8080), relay .146 (9001)	2,176 (544 x 4)	Epicor Kinetic ERP, Microsoft SQL Server	`Ice.BpDirective` injection, SQL Agent `CmdExec` tunnel, ADSI LDAP queries, AD and privilege-escalation tools
Apache Activiti workbench	Adversary staging	106.75.249[.]126 (8081)	81	Apache Activiti BPMN 2.0	WAF-evading Modeler JSON (`build_model.py`) and Groovy `ProcessBuilder` models
VMware VCDA and VCD workbench	Dual-use research, watchlist only	Withheld (10000)	65 (32 `.py` , 32 `.sh` , 1 `.txt`)	VCDA 4.7.4, VCD 10.6.0	Semicolon path traversal, `Config-Secret` root reset, Jetty smuggling, VCD key unwrap
n8n and PostgreSQL server	Inferred victim working directory	Withheld	24	n8n, PostgreSQL	JWT forgery, Code Node injection, PostgreSQL `COPY` webshell, PwnKit

Only the first two environments are treated as adversary infrastructure. The workbench and the n8n server are withheld and go to watchlists, for reasons given in Chapters 4 and 5.

What the files are for

Grouping all 2,346 entries by purpose gives eight stages. Schema reconnaissance and host tooling on the mirror dominate the count.

Stage	Unique paths	Entries	Share	Main files
SQL schema reconnaissance and host post-exploitation	497	1,988	84.7%	`BP*.sql` , `CFG*.sql` , Kerberos `.ccache` , `npusers.log` , Potato privilege-escalation tools (497 x 4)
Activiti BPMN 2.0 Groovy models	81	81	3.5%	35 `.json` and 18 `.xml` models, `rce*.bpmn20.xml` , `build_model.py` , `agent.x64.exe`
SQL Server Agent `CmdExec` persistence	19	76	3.2%	`A_job.sql` , `AA_mcp.sql` , `AB_diag.sql` , `AC_diag2.sql` , `AD_svc.sql` (19 x 4)
VMware VCDA probes and VCD key decryption	65	65	2.8%	`set-site.sh` , `poc_iter1.py` , `rp.sh` , `decrypt_vcd_secrets.py` , `exploit_payloads.py`
SQL Server ADSI linked-server AD reconnaissance	15	60	2.6%	`adsi.sql` , `adsi2.sql` , domain user dumps `users*.txt` (15 x 4)
Epicor `Ice.BpDirective` injection	12	48	2.0%	`BPRCE.sql` to `BPRCE5.sql` , `bprec.sql` , `bpm_off.sql` , `apply_BPRCE.json` (12 x 4)
n8n JWT abuse and privilege escalation	24	24	1.0%	Python, C, SQL and PHP files for session forgery, PostgreSQL `COPY` and PwnKit
IIS ASPX check file	1	4	0.2%	`/shell.aspx` , which returns the marker `RCX-211-OK` (1 x 4)

CHAPTER 03

Epicor Kinetic Directive Injection and SQL Agent Persistence

IN SHORT

The four mirrored servers hold an ERP and SQL Server toolkit in four stages: schema discovery, directive injection, SQL Agent jobs that call a relay, and Active Directory queries from inside the database. Defenders should audit directive changes and SQL Agent job creation together.

INSIDE THIS CHAPTER

Stage 1: learning the schema from error messages 10

Stage 2: a directive written from the database layer 11

Stage 3: SQL Server Agent jobs instead of a shell 12

Stage 4: Active Directory queries from inside SQL Server 12

03 Epicor Kinetic Directive Injection and SQL Agent Persistence

Epicor Kinetic stores Business Process Management (BPM) hooks in the `Ice.BpDirective` SQL table. When a business method runs, Kinetic compiles and runs the C# custom actions attached to it. Whoever can write to that table can therefore place code inside the ERP application server.

Our analysis covers 40 SQL, JSON and ASPX files from 43.230.171[.]147:8080, seen on 13 November 2026. The same tree sits on .148, .149 and .150. These are staged files. No target-side record shows that a directive was compiled.



Figure 2: Operators chain Epicor schema errors into SQL Agent tunnels and ADSI queries

Stage 1: learning the schema from error messages

Schema enumeration scripts (`BP.sql` to `BP8.sql`, `CFG.sql` to `CFG7.sql`, `C1.sql`, `C2.sql`) read `Ice.BpDirective`, `Ice.BpMethod` and `ApplicationServerSetting`. They return the results through `RAISERROR` messages, so the data comes back inside SQL error text.

The operator then sent `BPRCE.sql` through an exposed target database migration endpoint. The target answered HTTP 409 with `applied: false` and `Invalid column name 'DisplayOrder'`. The response is saved in `apply_BPRCE.json`.

The operator staged revised scripts that drop the mismatched column and test `UPDATE` statements against the directive table. The server holds no later HTTP 200 response. So the files show adaptation to the error, but not a working directive on the target.

Stage 2: a directive written from the database layer

Directives built in the Workflow Designer normally carry `Source = 'BO'` or `'UI'`. `BPRCE.sql` instead deletes any earlier `rcx_directive` row and inserts a `DirectiveDefinition2` XAML payload with these values:

Field	Value
`BpMethodCode`	`Ice.AlternateKey.Update`
`Source`	`'DB'`
`Name`	`rcx_directive`
`CGCCode`	`rcx00001`
Action	A `CustomCodeAction` that calls `System.IO.File.WriteAllText` to write `rcx_bpm.txt` into the IIS webroot (database folder name withheld)

`/shell.aspx` returns `RCX-211-OK` to check that the webroot is writable. `clean.sql` removes rows with `Name LIKE 'rcx%'` after testing.

Scripts `BPRCE.sql` to `BPRCE3.sql` set `Company = 'EPIC06'`, a standard Epicor training company code. That fits testing against a demo tenant or reuse of a template value. The wider 544-path tree also holds Windows and Active Directory tools. Together, the pattern fits an operator who pairs domain access tooling with ERP directive development.

File group	Example paths	Size or count	Role
Migration error log	`/apply_BPRCE.json`	11,416 B (1 x 4)	Holds the HTTP 409 reply (`RC_UNKNOWN` , `Invalid column name 'DisplayOrder'`)
Directive injectors	`/BPRCE.sql` to `/BPRCE5.sql` , `/bprec.sql` , `/bpm_off.sql`	433 to 2,893 B (8 x 4)	Drop `DisplayOrder` and insert a `Source='DB'` XAML `CustomCodeAction` on `Ice.*.Update`
Schema enumerators	`/BP.sql` to `/BP8.sql` , `/CFG.sql` to `/CFG7.sql` , `/C1.sql` , `/C2.sql`	223 to 774 B (17 x 4)	Read directive, method and server settings through `RAISERROR`
Check and rollback	`/shell.aspx` , `/clean.sql`	37 B and 238 B (2 x 4)	Return the `RCX-211-OK` marker; delete `rcx%` rows after testing
AD and Windows privilege escalation	Kerberos `.ccache` (user name withheld), `npusers.log` , `GodPotato-NET4.exe` , `PrintSpoofer64.exe` , `efspotato.cs`	Within the 544-path tree	Kerberos ticket cache, AS-REP roasting logs, domain user dumps (`users*.txt`) and local privilege escalation

Stage 3: SQL Server Agent jobs instead of a shell

`A_job.sql` enables `xp_cmdshell` under job `rcx_cfg`. That path would make `sqlservr.exe` start `cmd.exe`, which many tools watch. The other scripts move persistence into SQL Server Agent instead. They call `msdb.dbo.sp_add_jobstep` with `@subsystem = N'CmdExec'`.

Script	Size	Job	What the job is configured to do
`A_job.sql`	636 B	`rcx_cfg`	Enable `xp_cmdshell`
`AA_mcp.sql`	734 B	`rcx_mcp`	Start `C:\ProgramData\rs.exe` as a TCP tunnel to 43.230.171[.]146:9001 with `--mode tcp --reconnect` and an authentication value (withheld)
`AB_diag.sql`	985 B	`rcx_diag`	Write `tasklist`, a `netstat -ano` check for the relay, and Windows Defender status (`Get-MpThreat`) to `C:\ProgramData\diag.txt`
`AC_diag2.sql`	1,044 B	`rcx_diag2`	A second diagnostic job with the same purpose
`AD_svc.sql`	832 B	`rcx_svc`	Download `svc.exe` and `start_svc.ps1` from `http://43.230.171[.]146:8080/` with `curl.exe`
`AE_read.sql`	Not recorded	None	Read job output back through `OPENROWSET(BULK ...)`

The diagnostic jobs check whether the tunnel is up and whether Defender has flagged anything. For defenders, the key signal is `sqlagent.exe` starting a binary from `C:\ProgramData\`.

Stage 4: Active Directory queries from inside SQL Server

`adsi.sql` (468 B) registers the OLE DB provider `ADSDSOObject` as a linked server named `ADSI` inside the `master` database. It sets `@useself = 'TRUE'`, so SQL Server queries the directory with its own identity.

On Windows, this provider loads `activeds.dll` and `adsldp.dll`. The script then queries domain `sAMAccountName` and `distinguishedName` values over LDAP through `OPENQUERY`. The results return inside the SQL response, so no separate tool touches the domain controller.

CHAPTER 04

VMware VCDA 4.7.4 and VCD 10.6.0 Workbench

IN SHORT

A 65-file workbench holds three separate tracks against VMware lab appliances: VCDA path-traversal probes, a VCDA root-password reset, and Jetty smuggling plus an offline keystore decryptor for VCD. Defenders should isolate appliance management ports and protect keystores.

INSIDE THIS CHAPTER

Three separate tracks 15

Which advisories apply 15

04 VMware VCDA 4.7.4 and VCD 10.6.0 Workbench

Cloud service providers use VMware Cloud Director Availability (VCDA) and VMware Cloud Director (VCD) to run multi-tenant disaster recovery across vCenter and ESXi. A flaw in these appliances can expose many tenants at once.

The workbench listens on port 10000. It was first seen before this window. It shows clear research markers: a `revalidation/` folder, paced probes such as `oracle_probe.sh`, and license-patching scripts aimed at private lab addresses (`10.0.0.10` to `.40` and `192.168.100.10` to `.20`).

Because it may be a legitimate research setup, we withhold its address and place it on a dual-use watchlist. Its file hashes are withheld as well, so it is not in the Indicators chapter.

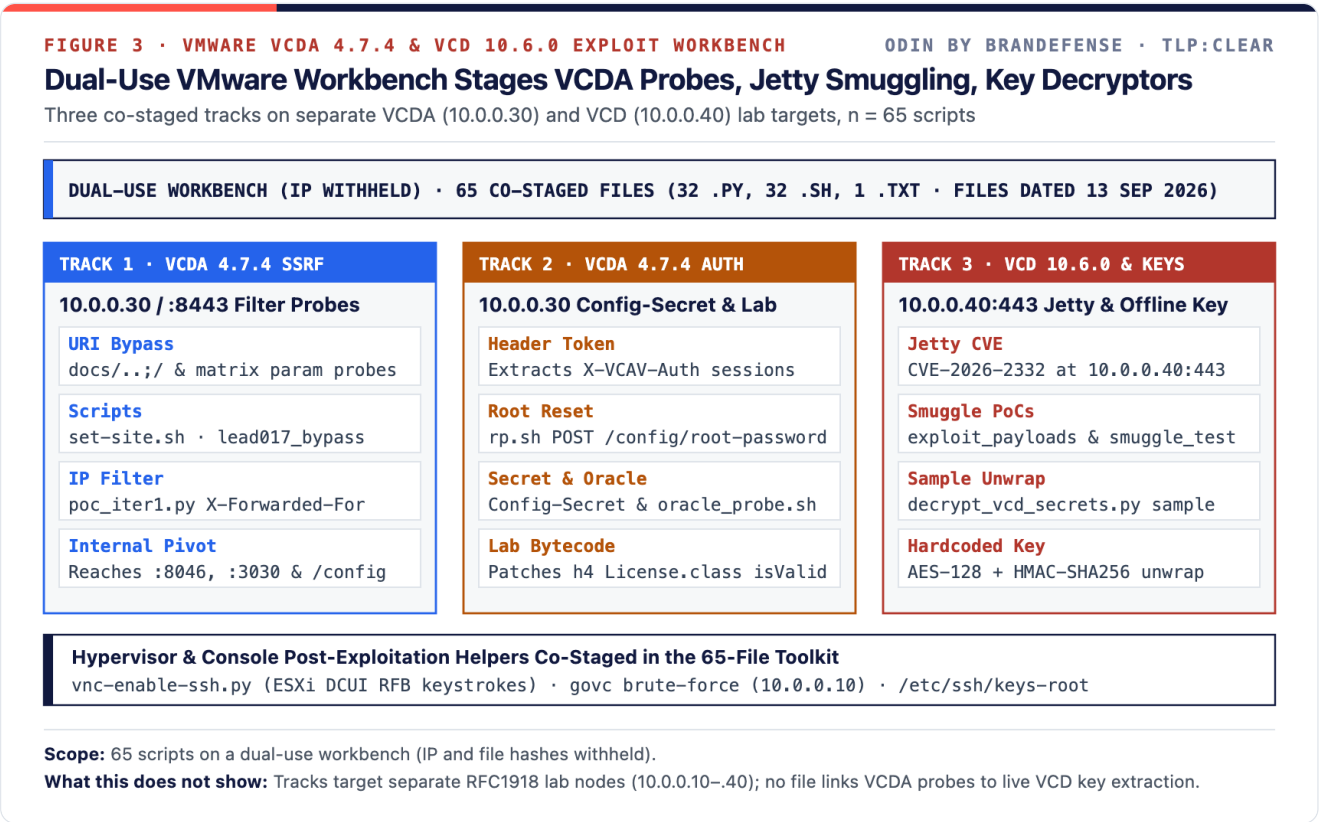


Figure 3: Dual-use VMware workbench co-stages VCDA probes, Jetty smuggling and key decryptors

Three separate tracks

The 65 files are not one verified chain from VCDA into VCD keystores. The VCDA scripts target lab node `10.0.0.30` and local ports 8443, 8046 and 3030. The Jetty scripts target VCD at `10.0.0.40:443`. The decryptor works on a hardcoded master key and sample values. No script links the VCDA probes to live key extraction.

Track	Target	Key files (size)	What it tests
1. VCDA 4.7.4 path traversal and IP filter	`10.0.0.30`, `localhost:8443`	`set-site.sh` (1,370 B), `open_firewall_and_test.sh` (893 B), `revalidation/lead017_bypass_test.sh` (4,748 B), `vcda_poc/poc_iter1.py` (21,220 B), `vcda_poc/poc_iter4.py` (16,737 B), `chunk_ext_probe.py` (2,604 B)	Semicolon matrix-parameter paths (`docs/..;/`) on 8443 with `X-VCAV-Auth` headers to reach internal :8046 and :3030 (`/config`, `/is-configured`); `X-Forwarded-For` tricks and raw TLS against the `SourceIpAuthorizer.class` allowlist; a Tomcat 9.0.83 probe
2. VCDA 4.7.4 root password reset	`10.0.0.30`	`rp.sh` (789 B), `weak_root_endpoint/attack.sh` (16,936 B), `b1/oracle_probe.sh` (2,501 B)	`POST /config/root-password` with a lab-configured `Config-Secret` header; single-character password-oracle timing probes; local helpers that change `iptables` and patch `License.class` (`isValid`)
3. VCD 10.6.0 Jetty smuggling and key unwrap	`10.0.0.40:443` (`/cloudapi`)	`exploit_payloads.py` (9,236 B), `smuggle_test.py` (13,055 B), `ssrf_e2e_smuggle.py` (10,165 B), `decrypt_vcd_secrets.py` (6,476 B)	Jetty 10.0.20 chunked-extension request smuggling (CVE-2026-2332); cross-appliance forwarding; offline AES-128 and HMAC-SHA256 unwrap of sample values

The toolkit also holds a VCD 10.6 Jackson deserialization proof of concept (`poc.py`, 12,548 B), PostgreSQL token scripts, and an ESXi console helper (`vnc-enable-ssh.py`, 3,327 B). Figure 3 also shows `govc` brute-force against `10.0.0.10` and the path `/etc/ssh/keys-root`.

Which advisories apply

The Jetty probes target CVE-2026-2332, a published request-smuggling flaw in chunked-extension quoted-string parsing. NVD lists Jetty up to 9.4.59, 10.0.0 to 10.0.27, 11.0.0 to 11.0.28, 12.0.0 to 12.0.32 and 12.1.0 to 12.1.6 as affected. Fixed releases are 9.4.60, 10.0.28, 11.0.29, 12.0.33 and 12.1.7. The VCD target's Jetty 10.0.20 falls in the affected range.

`LEAD-017` and `CONF-002` are the author's own project labels, not Broadcom advisory IDs. Broadcom `VMSA-2023-0026` (CVE-2023-34060) covers VCD Appliance 10.5 upgrades. It is cited only as hardening context, not as the flaw these scripts test.

CHAPTER 05

Workflow Engines: n8n and Apache Activiti

IN SHORT

Workflow engines link stored credentials to scripting runtimes. Files on a third-party n8n server and on a dedicated Activiti workbench show workflow JSON and BPMN 2.0 models built to call operating-system commands once imported into an unsandboxed engine.

INSIDE THIS CHAPTER

n8n: an inferred attacker working directory on a victim server 17

Apache Activiti: WAF-evading BPMN models 18

05 Workflow Engines: n8n and Apache Activiti

Workflow platforms run custom script nodes by design. A staged workflow definition can therefore ask for operating-system commands under the engine's service account. To get there, an attacker needs a valid session or an exposed model-import endpoint on an engine that allows scripts.

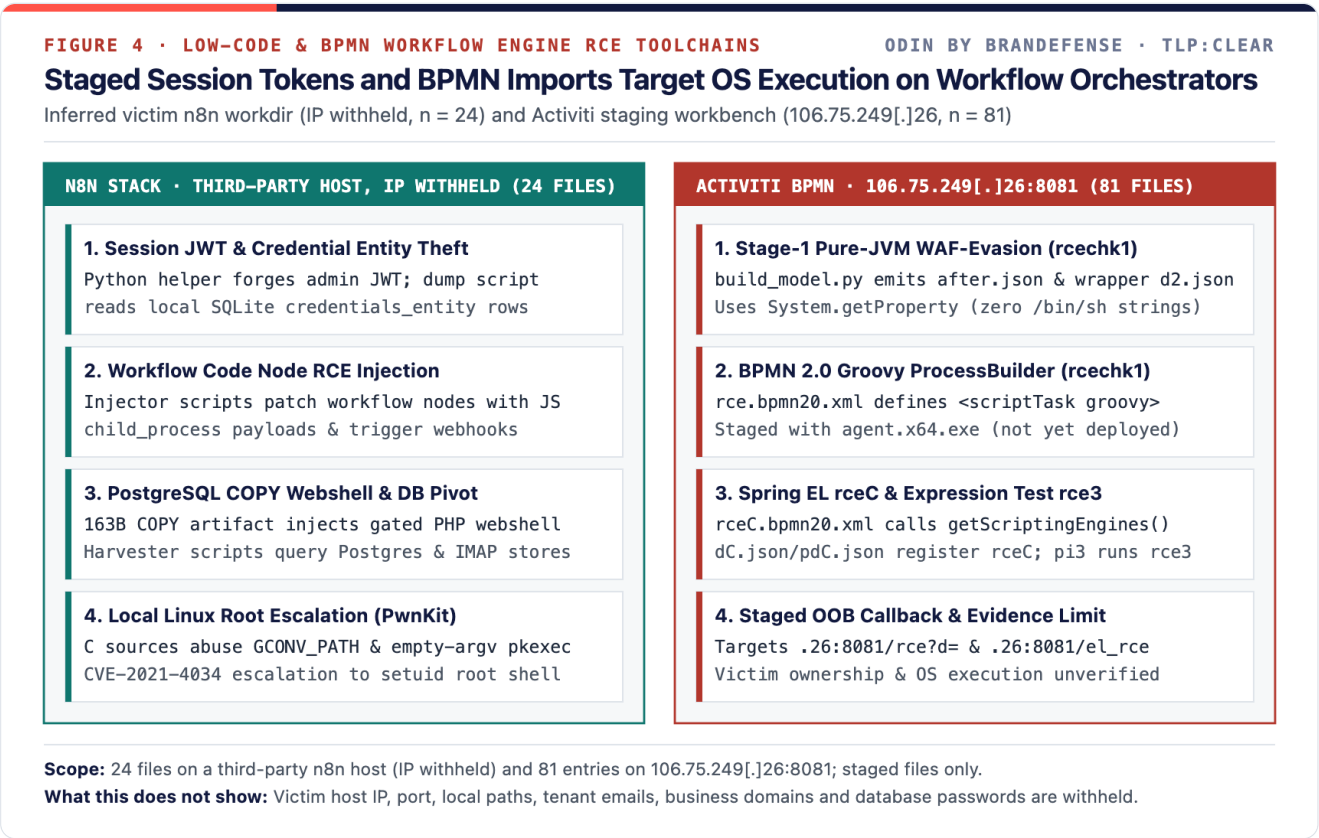


Figure 4: Staged session tokens and BPMN imports target OS execution on workflow orchestrators

n8n: an inferred attacker working directory on a victim server

The 24 files sit on a server that is not adversary infrastructure. The files were first seen before this window. Two Python scripts (6,144 B and 6,466 B) query local PostgreSQL databases. A 1,178 B script reads a local n8n SQLite store. Other scripts probe local container-bridge and loopback services.

These local references suggest an attacker working directory on a third-party n8n and PostgreSQL server. We did not confirm this with the owner. We withhold the address, port and host paths, and place the host on a watchlist. Its two file hashes are withheld as well.

Capability	Files	What they are written to do
JWT session forgery and Code Node injection	Three Python scripts (898 B, 3,701 B, 5,280 B)	Forge or replay n8n session JWTs to query `/rest/workflows`, read `credentials_entity` rows from SQLite, and patch workflows with `n8n-nodes-base.code` JavaScript nodes calling `child_process`
PostgreSQL webshell	One 163 B text file	Record a PostgreSQL `COPY (SELECT ...)` that writes a PHP webshell gated by a `POST` parameter (key withheld)
Credential harvesting	One 6,144 B Python script	Query local database and IMAP stores (all third-party identifiers withheld)
PwnKit escalation	Two C sources (781 B, 1,152 B)	Target Polkit `pkexec` (CVE-2021-4034) to start a setuid root shell

Apache Activiti: WAF-evading BPMN models

The dedicated workbench at 106.75.249[.]26:8081 holds 81 entries, seen on 13 November 2026. They include 35 `.json` files and 18 `.xml` files, 13 of them `rce*.bpmn20.xml` definitions. The operator built three families of workflow definitions and kept REST replies from an Activiti endpoint.

> **Stage 1: a model with no shell strings (`rcechk1`).**
>
> `build_model.py` (4,021 B) carries a Chinese comment that translates as "stage1: pure JVM code without shell signature strings to evade WAF". It builds an Activiti Modeler graph with a start event, one script node and an end event.

The script node reads `System.getProperty("user.name")` and `InetAddress.getLocalHost()`. It then calls `openStream()` to `http://106.75.249[.]26:8081/rce?d=...`. It contains no `/bin/sh` string. The output is saved as `after.json` (2,970 B) and wrapped in Base64 in `d2.json` (4,530 B).

> **Stage 2: Groovy and Spring EL models.**
>
> `rce.bpmn20.xml` (1,150 B) defines process `rcechk1` with `<scriptTask scriptFormat="groovy">` calling `new ProcessBuilder(...)`. It is staged next to `agent.x64.exe`. Query files `pd2.json` and `pi2.json` show that `rcechk1` was not yet registered when they were saved.

`rceC.bpmn20.xml` defines process `rceC`. Its `serviceTask` expression calls `springProcessEngineConfiguration.getScriptingEngines().evaluate(..., 'groovy')` to send an HTTP callback to `106.75.249[.]26:8081/el_rce`.

> **What the saved REST replies show.**
>
> The replies come from an Activiti endpoint whose address is withheld.

Saved reply	Size	What it records
`d3.json` to `dB.json`	1,286 B	Deployment replies for test processes
`pi3.json`	372 B	Process `rce3:1:695771` completed (`completed: true`); it evaluated the harmless expression `${1+1}`
`dC.json`	1,494 B	Deployment 695800
`pdC.json`	575 B	Process definition `rceC:1:695802` registered

These replies show that test processes were accepted. They do not show who owns the endpoint, and they do not show any operating-system command running.

06

CHAPTER 06

Telling Abuse from Normal Admin Work

IN SHORT

ERP systems and SQL servers run many legitimate directives and scheduled jobs every day. Blocking on a single signal causes outages and false attribution, so each check below pairs two or more signals.

06 Telling Abuse from Normal Admin Work

Blocking every `Ice.BpDirective` change, or treating every exposed server as attacker-owned, would break business processes and blame the wrong people. Four checks keep alerts accurate.

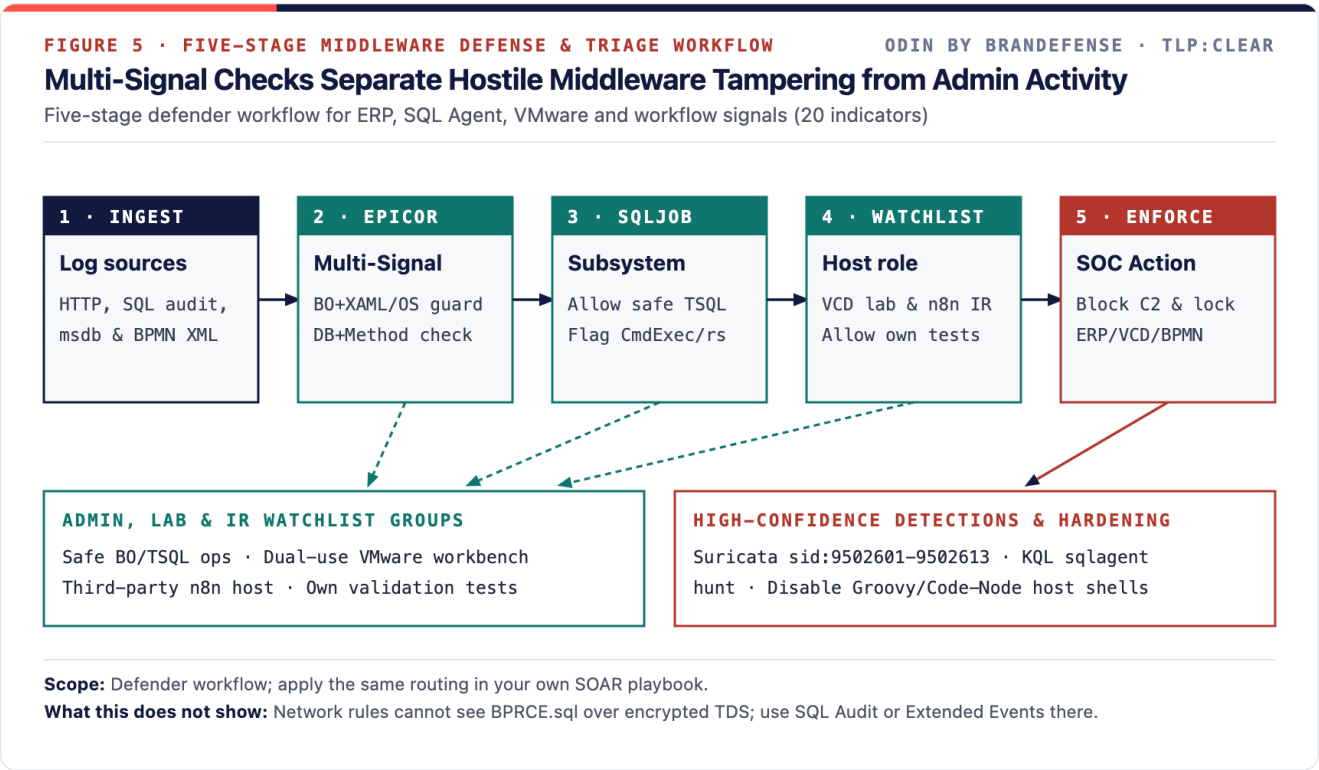


Figure 5: Multi-signal checks separate hostile middleware tampering from admin activity

1 **Epicor directives need more than one signal.**

An attacker who injects SQL controls the `Source` column. Legitimate data directives may also use `Source = 'DB'`. So `Source` alone settles nothing. Correlate three signals:

- `Source = 'DB'` on a Business Object update method (`*.Update` in `Ice.*` or `Erp.*`)
- `CustomCodeAction` XAML that calls operating-system or file namespaces (`System.IO`, `Process`, `WebClient`)
- A write by a SQL principal that is not the Epicor application-server login

Treat an unknown SQL principal as unknown, not as benign. Only a verified application-server login should clear a directive change.

2 **SQL Agent maintenance is not the same as `CmdExec` abuse.**

Database administrators schedule `msdb` jobs with the `TSQL` or `SSIS` subsystems for index rebuilds, `BULK INSERT` feeds and backups. Allow those. Flag the `CmdExec`, `PowerShell` and `ActiveScripting` subsystems, and `TSQL` steps that call `xp_cmdshell`, `sp_OACreate` or `OPENROWSET(BULK ...)`.

3 **Research workbenches and victims go to watchlists.**

The VMware workbench carries lab markers (`LEAD-017`, private targets `10.0.0.10` to `.40`). The n8n server looks like a victim hosting an attacker's working directory. Neither belongs on a permanent block list. Their addresses and file hashes are withheld, so hunt both through the behaviours in Chapters 4 and 5.

4 **Allowlist your own validation tests.**

Out-of-band callbacks from your own security testing or bug-bounty tooling can look like attacker callbacks. List those hosts explicitly so they are never flagged as hostile.

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

These steps cover the Epicor and SQL Server chain, the VMware workbench and the workflow engines in Chapters 3 to 5, ordered by urgency.

Within 24 hours

- [ ] Hunt or block 43.230.171[.]146 to .150 and 106.75.249[.]26 after checking their current owner and service banners. Cloud and VPS addresses get reassigned.
- [ ] Search SQL Server hosts for `sqlagent.exe` or `sqlservr.exe` starting `cmd.exe`, `powershell.exe`, `curl.exe`, `rs.exe` or `svc.exe` with `C:\ProgramData\` paths.
- [ ] Search IIS webroots for `rcx_bpm.txt` and unexpected `.aspx` files written by `w3wp.exe`.

Within 72 hours

- [ ] Audit `Ice.BpDirective` with SQL Audit or Extended Events. Flag non-application-server principals, `CustomCodeAction` code that calls `System.IO` or `Process`, and `Source='DB'` on `*.Update` methods.
- [ ] Revoke `SQLAgentOperatorRole` in `msdb` from ERP application logins.
- [ ] Disable `xp_cmdshell` and OLE Automation. Restrict SQL Agent `CmdExec` and PowerShell steps to least-privilege proxies and require signed scripts or executables.
- [ ] Remove unauthorized `ADSI` linked servers (`sp_dropserver`) and set `DisallowAdhocAccess=1` on `ADSDSOObject`.

Within 7 to 30 days

- [ ] Remove VCDA ports 8443, 8046 and 3030 from public internet access. Apply Broadcom and Eclipse Jetty patches. Rotate VCD keystore secrets and `Config-Secret` values.
- [ ] In n8n, use sandboxed task runners, set `N8N_BLOCK_ENV_ACCESS_IN_NODE=true` and rotate `N8N_ENCRYPTION_KEY`.
- [ ] Patch Polkit on Linux hosts and add `auditd` `EXECVE` monitoring for `pkexec` started with no arguments.
- [ ] Disable unrestricted Groovy and JSR-223 `<scriptTask>` execution in Apache Activiti. Require role-based review of every BPMN 2.0 XML and Modeler JSON import.
- [ ] Deploy the Suricata rules behind TLS offload, and send matches on the workbench and n8n behaviours to analyst review.

08

CHAPTER 08

Detection Rules

IN SHORT

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

INSIDE THIS CHAPTER

What each rule needs 26

How the Suricata signatures work 27

How the KQL hunt works 27

How the Sigma and YARA rules work 28

How to route an event 28

08 Detection Rules

The full rules are in Appendix B. Network rules catch delivery and staging attempts; pair them with host logs before treating a match as execution.

Rule file	What it catches
`suricata_erp_vcda_and_workflow_rce.rules`	Suricata 7: thirteen signatures for directive injection, SQL Agent dropper staging, the IIS check marker, VCDA path traversal and root reset, and Activiti and n8n script uploads
`sentinel_mssql_cmdexec_and_erp_hunt.kql`	Microsoft Sentinel and Defender XDR: SQL Agent child processes, ADSI module loads with LDAP traffic, IIS webroot writes, audited directive writes, and traffic to the listed IPs
`sigma_erp_sqlagent_and_bpmn_rce.yml`	Sigma (3 rules): SQL Agent droppers, IIS writes of BPM marker files, and workflow runtimes starting shells or PwnKit
`yara_erp_vcda_and_workflow_toolkits.yar`	YARA (4 rules): Epicor directive injectors, SQL Agent and ADSI scripts, VCDA and VCD tooling, and Activiti and n8n payloads

What each rule needs

Rule set	Data it needs
Suricata	Suricata 7 inspecting HTTP behind the reverse proxy or load balancer that ends TLS. Put ERP servers, SQL Server segments, VCDA appliances (443, 8443, 8046, 3030) and workflow engines in `$HOME_NET`.
KQL	Microsoft Defender for Endpoint process, image-load, file and network events on SQL Server and Epicor IIS hosts. SQL Server Audit logging `SCHEMA_OBJECT_ACCESS_GROUP` on `Ice.BpDirective` (Event ID 33205) to the Security or Application log.
Sigma	Windows process creation with command lines (Event ID 4688 or Sysmon Event ID 1) and file creation (Sysmon Event ID 11 or EDR). On Linux, process creation that records parent command line, working directory and empty command lines, plus `auditd` `EXECVE` rules on n8n and Activiti servers.
YARA	YARA 4 on endpoint file scans, web-upload inspection or incident-response collectors. Keep the file-size caps as written.

Exclude authorized DBA script folders that hold `ADSDSOObject` linked-server scripts from the SQL Agent and ADSI YARA rule. Exclude approved security-assessment jump hosts from the VMware rule.

How the Suricata signatures work

SID	Looks for	Why
9502601	`Ice.BpDirective` in a request body with `CustomCodeAction`, `BpMethodCode`, `'DB'` or `rcx_directive`	Directive injection over HTTP (T1505.001)
9502602	`sp_add_jobstep` and `CmdExec` with `ProgramData`, `rs.exe`, `svc.exe`, `curl.exe` or `--reconnect`	SQL Agent dropper scripts in transit (T1505.001)
9502603	`RCX-211-OK` or `RCX-BPM-RCE` in a response from a protected server	The webroot check marker (T1505.003)
9502604	Literal `..;/` in the raw URI on ports 443, 8443, 8046, 3030	VCDA semicolon path traversal (T1190)
9502605	`POST /config/root-password` with a `Config-Secret` header	VCDA root password reset (T1190)
9502606	`ProcessBuilder` and `groovy` in a request body	Plaintext Groovy BPMN upload (T1059)
9502607	`/workflows` URI with `n8n-nodes-base.code` and `child_process`	n8n Code Node injection (T1059.007)
9502608	URL-encoded `%3b/` after `..` or `%2e%2e`	Encoded VCDA path traversal (T1190)
9502609	`json_xml` with `StartNoneEvent` and JVM reconnaissance calls, in plaintext or Base64	Activiti Modeler stage-1 upload (T1059)
9502610	`getScriptingEngines().evaluate` in a request body	Plaintext Spring EL BPMN deployment (T1059)
9502611 to 9502613	The same call in each of three Base64 alignments	Base64-wrapped Spring EL deployments, with no file name needed

Tune SID 9502601 by allowlisting the hosts that deploy Epicor solution packages. Tune SID 9502607 by scoping it to approved CI/CD runners if your developers use `child_process` in Code Nodes.

How the KQL hunt works

The query looks back 14 days and joins five branches.

Branch	Flags
SQL child processes	`sqlagent.exe` or `sqlservr.exe` starting `cmd.exe`, PowerShell, `curl.exe`, `certutil.exe`, `rs.exe` or `svc.exe` with `C:\ProgramData\`, `--reconnect`, `--port 9001`, `start_svc.ps1`, `diag.txt` or `Get-MpThreat`
ADSI module loads	`sqlservr.exe` loading `activeds.dll`, `adsldp.dll` or `adsldpc.dll`
IIS webroot writes	`w3wp.exe` writing `.aspx` files or `rcx_bpm.txt` under `inetpub\wwwroot`
SQL Audit	Event ID 33205 inserts, updates or deletes on `Ice.BpDirective` by a principal outside your Epicor application logins
Network	Connections to the relay or staging IPs, public ports 9001, 8080 or 8081 from SQL or IIS processes, and SQL Server LDAP traffic on 389, 636 or 3268

SQL Server talks to domain controllers during normal Windows authentication. So LDAP traffic is labelled a linked-server query only when an ADSI DLL loaded in the same process within the previous 15 minutes. Otherwise it is a low-confidence hunt lead. Fill the authorized principal list with your real Epicor application-pool logins before you create incidents from this branch.

How the Sigma and YARA rules work

Rule	Fires when
Sigma 1: SQL Agent dropper	`sqlagent.exe` or `sqlservr.exe` starts a shell, `curl.exe`, `certutil.exe`, `rs.exe` or `svc.exe`, and the command line holds a `ProgramData` path, tunnel flag, relay IP or diagnostic string
Sigma 2: IIS BPM write	`w3wp.exe` writes `rcx_bpm.txt` or an `.aspx` file under `inetpub\wwwroot`; check `.aspx` writes against upgrade windows, but always alert on `rcx_bpm.txt`
Sigma 3: workflow shell or PwnKit	A Java or Node workflow runtime (Activiti, Flowable, n8n) starts a shell or `curl`, excluding npm build scripts; or `pkexec` starts with an empty command line from a writable directory or workflow parent; or `GCONV_PATH=.` or `/pwned_bash` appears
YARA: Epicor injector	A directive table name plus XAML or update-method strings plus `'DB'` or `rcx` markers; or the `DisplayOrder` error with `RC_UNKNOWN`; or `RCX-211-OK`
YARA: SQL Agent and ADSI	`sp_add_jobstep` with `CmdExec` or `xp_cmdshell` plus a `ProgramData`, `start_svc.ps1` or `rcx_` string; or `sp_addlinkedserver` with `ADSDSOObject` and ADSI query strings
YARA: VCDA and VCD	Two or more VCDA strings (`X-VCAV-Auth`, `/config/root-password`, `Config-Secret`, `docs/..;/` and others); or ECB mode with a key-unwrap or PBKDF2 string
YARA: Activiti and n8n	Groovy with `ProcessBuilder` or JVM reconnaissance calls; Base64 Modeler markers; `getScriptingEngines().evaluate`; n8n credential or REST strings with `child_process`; `COPY (SELECT` with PHP; or `GCONV_PATH=.` with `/usr/bin/pkexec`

The YARA rules cap file size at 150 KB to 250 KB to avoid scanning database backups.

How to route an event

What the event shows	Action
A host on your own validation-test allowlist	Allowlist
A `BO` or `UI` directive by a verified application-server login, with no operating-system code	No action
A `TSQL`, `SSIS` or Analysis Services job step with no dangerous calls	No action
A directive change by an unknown SQL principal	Analyst review
A `Source='DB'` directive on an update method, a non-application-server writer, or a `CmdExec` or `xp_cmdshell` step	Block or isolate
Directive injection, SQL Agent tunnel and ADSI staging together	Block or isolate, highest priority
Any match on the dual-use VMware workbench	Dual-use watchlist
Any match on the third-party n8n server	Victim watchlist

09

CHAPTER 09

Indicators

IN SHORT

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

INSIDE THIS CHAPTER

Hunt surfaces 30

Adversary IP addresses 30

File hashes 30

MITRE ATT&CK mapping 31

09 Indicators

The full list is in Appendix A. It holds 20 indicators: 6 addresses and 14 file hashes.

Type	Count
Adversary IP addresses to block or hunt	6
SHA-256 file hashes for hunting	14, all from the two adversary hosts (the dual-use workbench hashes and a captured target response are withheld)

Withheld: the dual-use workbench address and the n8n server's address, port, paths and two file hashes. Also withheld: the internal Activiti endpoint, the ERP database name, user names, the tunnel authentication value and the webshell key.

Hunt surfaces

Some files land on target machines: `rcx_bpm.txt`, `/shell.aspx`, `C:\ProgramData\rs.exe`, `svc.exe`, `start_svc.ps1`, PostgreSQL `COPY` webshells and setuid `pkexec` binaries. Hunt for these on disk and in EDR logs. The other files live on attacker servers. Hunt for them on the wire with the Suricata rules and in SQL, IIS and workflow logs.

Adversary IP addresses

Address	Port	Role
43.230.171[.]146	9001	Relay and payload host, configured in `AA_mcp.sql` (9001) and `AD_svc.sql` (8080)
43.230.171[.]147	8080	Epicor and SQL Agent staging node 1 (544 files, 40 analysed)
43.230.171[.]148	8080	Staging node 2, same tree, including `apply_BPRCE.json` and `/UF.sql`
43.230.171[.]149	8080	Staging node 3, same tree
43.230.171[.]150	8080	Staging node 4, same tree, including `BPRCE2.sql` to `BPRCE5.sql`
106.75.249[.]26	8081	Apache Activiti BPMN workbench (81 files)

File hashes

File	Size	Host	SHA-256
`/BPRCE.sql`	2,484 B	43.230.171[.]147	`c82a37d15dd6ace1980345c5a6546ef9661d36cdf68df071c9e8e04a4a490195`
`/BPRCE2.sql`	2,470 B	43.230.171[.]147	`2fc82f149dfd3548501cc6ecbfc7c03122b5cc43812e101720689ec538513b61`
`/BPRCE3.sql`	2,469 B	43.230.171[.]147	`8f37377809fe32920d519a908862547fbbb5d6fe4da7d895be5976ab5d839f7d`
`/BPRCE4.sql`	816 B	43.230.171[.]147	`581cd858db05bc7f477f82b79f130e40737b7c4d73274518d18b332fb5610d8b`
`/A_job.sql`	636 B	43.230.171[.]147	`4d639f02eac420635177cd7d9f8bf2d8b2c440ed76dcaaeb67e36556c97ef61c`
`/AA_mcp.sql`	734 B	43.230.171[.]147	`ebd371b85fef6d4337fe115a9bea3e741a188ac0c6584a501717aa7f9f06820f`

File	Size	Host	SHA-256
`/AB_diag.sql`	985 B	43.230.171[.]147	`fc2b41026eae4f1f6c18790b787f7adf5199f83b11f3a4236620fdee773224df`
`/AD_svc.sql`	832 B	43.230.171[.]147	`1f783938b95a80c0974389401f8dc5df9f151244a176610964e812957d1757eb`
`/adsi.sql`	468 B	43.230.171[.]147	`eddefd7868e0687429d6529afe0fc30c909bae1f7cd07f10cc533fd0587e1e0e`
`/shell.aspx`	37 B	43.230.171[.]147	`ec01db6bdbbf6ca8f6a3e9f1569a9c5e7974046f04d4b1c26f178c53f12a791a`
`/rce.bpmn20.xml`	1,150 B	106.75.249[.]26	`66cb5d6befdd13394da5f979e2021680ebe37a2e9059e4f8f08f35e750382ba7`
`/build_model.py`	4,021 B	106.75.249[.]26	`266e2b914dacfe7b24a6cb0dd41d76d0d19535accefd3daf6ad0e2a98e0a4427`
`/after.json`	2,970 B	106.75.249[.]26	`d32ff4622418a9194bc19ae2b062df3a29d6b2fa2465ba21db1f0f29052ea7bc`
`/d2.json` (Base64 wrapper)	4,530 B	106.75.249[.]26	`1b74813d4e4f39472402cdfb2baefb02988a27b280f56c6a81d2547ef1915126`

MITRE ATT&CK mapping

Tactic	Technique	How the toolkits use it	Detection
Persistence	T1505.001 Server Software Component: SQL Stored Procedures	`Ice.BpDirective` injection and SQL Agent `CmdExec` jobs	SID 9502601, 9502602; Sigma rule 1; KQL
Persistence	T1505.003 Server Software Component: Web Shell	`/shell.aspx` check file, `rcx_bpm.txt`, PostgreSQL `COPY` PHP webshell	SID 9502603; Sigma rule 2; YARA
Initial Access	T1190 Exploit Public-Facing Application	VCDA semicolon path traversal and `Config-Secret` root reset	SID 9502604, 9502605, 9502608
Execution	T1059 Command and Scripting Interpreter	Groovy and Spring EL BPMN models, WAF-evading Modeler JSON	SID 9502606, 9502609 to 9502613
Execution	T1059.007 JavaScript	n8n Code Nodes calling `child_process`	SID 9502607; YARA
Execution	T1059.004 Unix Shell	Workflow runtimes starting POSIX shells	Sigma rule 3
Privilege Escalation	T1068 Exploitation for Privilege Escalation	PwnKit (CVE-2021-4034) sources on the n8n server	Sigma rule 3; YARA

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.** Most entries were seen on 13 November 2026, inside a two-week observation window ending 18 November 2026, with a research cutoff of 15 November. The VMware workbench and the n8n server were first seen earlier; no dates are given for them.
- **Staged files, not outcomes.** The files show intent and tooling. No target-side record shows a compiled directive, a running SQL Agent job, a decrypted live keystore or an operating-system command run through a workflow engine.
- **Error replies are the attacker's own records.** The HTTP 409 reply and the Activiti REST replies were saved by the operator. They show what an endpoint answered, not who owns it.
- **Separate tracks.** The VMware files target separate lab systems. Nothing links the VCDA probes to VCD key extraction.
- **Roles are judgements.** The dual-use and victim roles rest on markers in the files, not on contact with the owners. That is why both hosts are withheld and watchlisted.
- **Addresses change.** Cloud and VPS addresses can be reassigned. Check current ownership before a permanent block.
- **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. T1505.001 Server Software Component: SQL Stored Procedures. https://attack.mitre.org/techniques/T1505/001/ and T1505.003 Web Shell. https://attack.mitre.org/techniques/T1505/003/ (verified 18 November 2026)
2. Microsoft. Create a CmdExec job step (SQL Server Agent). https://learn.microsoft.com/en-us/sql/ssms/agent/create-a-cmdexec-job-step (verified 18 November 2026)
3. Broadcom. VMSA-2023-0026, VMware Cloud Director Appliance authentication bypass (CVE-2023-34060). https://support.broadcom.com/web/ecx/support-content-notification/-/external/content/SecurityAdvisories/0/23679 (verified 18 November 2026)
4. Epicor. Epicor Kinetic product overview. https://www.epicor.com/en-us/products/enterprise-resource-planning-erp/kinetic/ (verified 18 November 2026)
5. Activiti. BPMN 2.0 user guide, script task. https://www.activiti.org/userguide/#bpmnScriptTask (verified 18 November 2026)
6. n8n. Securing n8n, hosting documentation. https://docs.n8n.io/hosting/securing/overview/ (verified 18 November 2026)
7. NVD. CVE-2026-2332, Eclipse Jetty request smuggling via chunked-extension parsing. https://nvd.nist.gov/vuln/detail/CVE-2026-2332 and Eclipse Jetty security reports. https://jetty.org/security.html (verified 18 November 2026)
8. NVD. CVE-2021-4034, Polkit `pkexec` local privilege escalation (PwnKit). https://nvd.nist.gov/vuln/detail/CVE-2021-4034 (verified 18 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 (20) 37

B. Detection rules (4 files) 38

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
R05-IOC-01	IPv4	43.230.171[.]146	9001	Epicor/MSSQL C2 Relay & Payload Server	BLOCK
R05-IOC-02	IPv4	43.230.171[.]147	8080	Epicor Kinetic & SQL Agent Staging Node #1	BLOCK
R05-IOC-03	IPv4	43.230.171[.]148	8080	Epicor Kinetic & SQL Agent Staging Node #2	BLOCK
R05-IOC-04	IPv4	43.230.171[.]149	8080	Epicor Kinetic & SQL Agent Staging Node #3	BLOCK
R05-IOC-05	IPv4	43.230.171[.]150	8080	Epicor Kinetic & SQL Agent Staging Node #4	BLOCK
R05-IOC-06	IPv4	106.75.249[.]26	8081	Apache Activiti BPMN 2.0 Groovy RCE Workbench	BLOCK
R05-IOC-07	SHA-256	c82a37d15dd6ace1980345c5a6546ef9661d36cdf68df071c9e8e04a4a490195	8080	Epicor Ice.BpDirective XAML RCE Injector SHA-256	HUNT
R05-IOC-08	SHA-256	2fc82f149dfd3548501cc6ecbfc7c03122b5cc43812e101720689ec538513b61	8080	Epicor Ice.BpDirective Variant 2 SQL SHA-256	HUNT
R05-IOC-09	SHA-256	8f37377809fe32920d519a908862547fbbb5d6fe4da7d895be5976ab5d839f7d	8080	Epicor Ice.BpDirective Variant 3 SQL SHA-256	HUNT
R05-IOC-10	SHA-256	581cd858db05bc7f477f82b79f130e40737b7c4d73274518d18b332fb5610d8b	8080	Epicor Ice.BpDirective Compact Variant 4 SHA-256	HUNT
R05-IOC-11	SHA-256	4d639f02eac420635177cd7d9f8bf2d8b2c440ed76dcaaeb67e36556c97ef61c	8080	SQL Agent rcx_cfg xp_cmdshell Enabler SHA-256	HUNT
R05-IOC-12	SHA-256	ebd371b85fef6d4337fe115a9bea3e741a188ac0c6584a501717aa7f9f06820f	8080	SQL Agent rcx_mcp CmdExec rs.exe Launcher SHA-256	HUNT
R05-IOC-13	SHA-256	fc2b41026eae4f1f6c18790b787f7adf5199f83b11f3a4236620fdee773224df	8080	SQL Agent rcx_diag Host & Netstat Collector SHA-256	HUNT
R05-IOC-14	SHA-256	1f783938b95a80c0974389401f8dc5df9f151244a176610964e812957d1757eb	8080	SQL Agent rcx_svc Curl & PowerShell Dropper SHA-256	HUNT
R05-IOC-15	SHA-256	eddefd7868e0687429d6529afe0fc30c909bae1f7cd07f10cc533fd0587e1e0e	8080	MSSQL ADSDSOObject Linked-Server AD Recon SHA-256	HUNT
R05-IOC-16	SHA-256	ec01db6bdbbf6ca8f6a3e9f1569a9c5e7974046f04d4b1c26f178c53f12a791a	8080	IIS ASPX RCX-211-OK Verification Probe SHA-256	HUNT
R05-IOC-17	SHA-256	66cb5d6befdd13394da5f979e2021680ebe37a2e9059e4f8f08f35e750382ba7	8081	Apache Activiti BPMN 2.0 Groovy ProcessBuilder RCE SHA-256	HUNT
R05-IOC-18	SHA-256	266e2b914dacfe7b24a6cb0dd41d76d0d19535accefd3daf6ad0e2a98e0a4427	8081	Activiti Modeler Pure-JVM WAF-Evasion Generator SHA-256	HUNT
R05-IOC-19	SHA-256	d32ff4622418a9194bc19ae2b062df3a29d6b2fa2465ba21db1f0f29052ea7bc	8081	Activiti Modeler Exported Workflow Payload JSON SHA-256	HUNT
R05-IOC-20	SHA-256	1b74813d4e4f39472402cdfb2baefb02988a27b280f56c6a81d2547ef1915126	8081	Activiti Stage-1 Pure-JVM Modeler Deployment Wrapper JSON SHA-256	HUNT

B Detection Rules

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

sentinel_mssql_cmdexec_and_erp_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Report #05 Detection Engineering Suite
// Target: Microsoft Sentinel / Defender XDR
// Context: Hunts SQL Server Agent (sqlagent.exe / sqlservr.exe) spawning CmdExec shells or ProgramData
//          tunnel binaries (rs.exe / svc.exe / curl.exe), in-process Active Directory ADSI module loads
//          (activeds.dll / adsldp.dll, noting MDE DeviceImageLoadEvents sampling) correlated per-load with
//          SQL Server LDAP queries within 15 minutes (separating uncorrelated LDAP egress into a lower-
confidence
//          hunt), Epicor IIS (w3wp.exe) webroot writes, and SQL Audit (EventID 33205 across SecurityEvent and
//          Application Event logs) Ice.BpDirective mutations parsed by schema, object, action, and principal.

let Lookback = 14d;
let AuthorizedEpicorSqlPrincipals = dynamic(["svc_epicor_app", "epicor_apppool"]);
let ConfiguredRelayIPs = dynamic(["43.230.171.146"]);
let StagingHostIPs = dynamic(["43.230.171.147", "43.230.171.148", "43.230.171.149", "43.230.171.150"]);
let SuspiciousSqlChildren = dynamic(["cmd.exe", "powershell.exe", "pwsh.exe", "curl.exe", "rs.exe", "svc.exe",
"certutil.exe"]);
let HostileRelayPorts = dynamic([9001, 8080, 8081]);
let AdsiLoadedSqlEvents =
    DeviceImageLoadEvents
    | where Timestamp >= ago(Lookback)
    | where InitiatingProcessFileName =~ "sqlservr.exe" and FileName in~ ("activeds.dll", "adsldp.dll",
"adsldpc.dll")
    | project DeviceName, SqlPid = InitiatingProcessId, AdsiLoadTime = Timestamp;
let SqlAuditEvents =
    union isfuzzy=true
    (
        SecurityEvent
        | where TimeGenerated >= ago(Lookback) and EventID == 33205
        | project Timestamp = TimeGenerated, DeviceName = Computer, AuditSource = "SecurityEvent", AuditPayload =
tostring(EventData)
    ),
    (
        Event
        | where TimeGenerated >= ago(Lookback) and EventLog =~ "Application" and EventID == 33205
        | project Timestamp = TimeGenerated, DeviceName = Computer, AuditSource = "ApplicationEvent",
AuditPayload = coalesce(tostring(RenderedDescription), tostring(EventData))
    );
union isfuzzy=true
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where InitiatingProcessFileName in~ ("sqlagent.exe", "sqlservr.exe")
    | where FileName in~ (SuspiciousSqlChildren)
    | where ProcessCommandLine has_any (@"C:\ProgramData\", "--reconnect", "--port 9001", "start_svc.ps1",
"diag.txt", "Get-MpThreat")
    | extend HuntSignal = "MSSQL_CmdExec_ProgramData_Or_Relay_Spawn",
             ArtifactDetail = ProcessCommandLine
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
),
(
    DeviceImageLoadEvents
    | where Timestamp >= ago(Lookback)
    | where InitiatingProcessFileName =~ "sqlservr.exe" and FileName in~ ("activeds.dll", "adsldp.dll",
"adsldpc.dll")
    | extend HuntSignal = "MSSQL_InProcess_ADSDSOObject_Module_Load",
             ArtifactDetail = FolderPath
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
),
(
    DeviceFileEvents
```

```
    | where Timestamp >= ago(Lookback)
    | where InitiatingProcessFileName =~ "w3wp.exe"
    | where FolderPath has @"inetpub\wwwroot" and (FileName endswith ".aspx" or FileName =~ "rcx_bpm.txt")
    | extend HuntSignal = "Epicor_IIS_BpDirective_Or_ASPX_Webroot_Write",
             ArtifactDetail = FolderPath
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
),
(
    SqlAuditEvents
    | extend ActionId = toupper(trim(@" []""'", extract(@"(?im)^\s*action_id\s*:\s*([^\r\n<]+)", 1,
AuditPayload))),
             SchemaName = tolower(trim(@" []""'", extract(@"(?im)^\s*schema_name\s*:\s*([^\r\n<]+)", 1,
AuditPayload))),
             ObjectName = tolower(trim(@" []""'", extract(@"(?im)^\s*object_name\s*:\s*([^\r\n<]+)", 1,
AuditPayload))),
             ServerPrincipal = tolower(trim(@" []""'", extract(@"(?im)^\s*server_principal_name\s*:\s*([^\r\n<]
+)", 1, AuditPayload))),
             SessionPrincipal = tolower(trim(@" []""'", extract(@"(?im)^\s*session_server_principal_name\s*:
\s*([^\r\n<]+)", 1, AuditPayload))),
             DbPrincipal = tolower(trim(@" []""'", extract(@"(?im)^\s*database_principal_name\s*:\s*([^\r\n<]+)",
1, AuditPayload))),
             StatementText = coalesce(extract(@"(?ims)^\s*statement\s*:\s*(.+?)(?:^\s*[a-z0-9_]+\s*:|\z)", 1,
AuditPayload), AuditPayload)
    | extend ServerPrincipalShort = tostring(split(ServerPrincipal, "\\")[-1]),
             SessionPrincipalShort = tostring(split(SessionPrincipal, "\\")[-1]),
             DbPrincipalShort = tostring(split(DbPrincipal, "\\")[-1]),
             EffectivePrincipal = case(
                 isnotempty(ServerPrincipal), ServerPrincipal,
                 isnotempty(SessionPrincipal), SessionPrincipal,
                 isnotempty(DbPrincipal), DbPrincipal,
                 "unknown"
             )
    | extend IsBpDirectiveObject = (SchemaName =~ "ice" and ObjectName =~ "bpdirective")
                 or StatementText matches regex @"(?i)\b(?:insert(?:\s+into)?|update|delete(?:\s+from)?|merge(?:
\s+into)?)\s+(?:\[?[a-z0-9_]+\]?\s*\.\s*)?\[?ice\]?\s*\.\s*\[?bpdirective\]?\b",
             IsMutationAction = ActionId in ("IN", "UP", "DL", "AL", "INSERT", "UPDATE", "DELETE", "ALTER")
                 or (isempty(ActionId) and StatementText matches regex @"(?i)\b(insert|update|delete|merge)\b"),
             IsAuthorizedPrincipal = case(
                 isnotempty(ServerPrincipal),
                     (ServerPrincipal in~ (AuthorizedEpicorSqlPrincipals) or ServerPrincipalShort in~
(AuthorizedEpicorSqlPrincipals))
                     and (isempty(SessionPrincipal) or SessionPrincipal in~ (AuthorizedEpicorSqlPrincipals) or
SessionPrincipalShort in~ (AuthorizedEpicorSqlPrincipals)),
                 isnotempty(SessionPrincipal),
                     (SessionPrincipal in~ (AuthorizedEpicorSqlPrincipals) or SessionPrincipalShort in~
(AuthorizedEpicorSqlPrincipals)),
                 isnotempty(DbPrincipal),
                     (DbPrincipal in~ (AuthorizedEpicorSqlPrincipals) or DbPrincipalShort in~
(AuthorizedEpicorSqlPrincipals)),
                 false
             )
    | where IsBpDirectiveObject and IsMutationAction and not(IsAuthorizedPrincipal)
    | extend HuntSignal = "Epicor_SQLAudit_NonAppServer_BpDirective_Write",
             ArtifactDetail = strcat("src=", AuditSource, " principal=", EffectivePrincipal, " action=",
ActionId, " stmt=", StatementText),
             InitiatingProcessFileName = "sqlservr.exe",
             FileName = "Ice.BpDirective"
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
),
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback) and ActionType == "ConnectionSuccess"
    | where InitiatingProcessFileName in~ ("sqlservr.exe", "sqlagent.exe", "rs.exe", "w3wp.exe")
        or (InitiatingProcessFileName in~ ("curl.exe", "powershell.exe", "cmd.exe") and
InitiatingProcessParentFileName in~ ("sqlagent.exe", "sqlservr.exe"))
    | where RemoteIP in (ConfiguredRelayIPs) or RemoteIP in (StagingHostIPs)
        or (RemoteIPType == "Public" and RemotePort in (HostileRelayPorts))
        or (InitiatingProcessFileName =~ "sqlservr.exe" and RemotePort in (389, 636, 3268))
    | join kind=leftouter AdsiLoadedSqlEvents on DeviceName, $left.InitiatingProcessId == $right.SqlPid
    | extend IsTimelyAdsiLoad = isnotempty(AdsiLoadTime) and Timestamp between (AdsiLoadTime .. (AdsiLoadTime +
15m))
    | summarize HasPrecedingAdsiLoad = max(toint(IsTimelyAdsiLoad)) == 1
```

```
        by Timestamp, DeviceName, InitiatingProcessId, InitiatingProcessFileName, RemoteIP, RemoteIPType,
RemotePort
    | extend HuntSignal = case(
             RemoteIP in (ConfiguredRelayIPs), "Configured_Relay_IP_Egress_Hunt",
             RemoteIP in (StagingHostIPs), "Staging_Host_IP_Egress_Hunt",
             InitiatingProcessFileName =~ "sqlservr.exe" and RemotePort in (389, 636, 3268) and
HasPrecedingAdsiLoad, "MSSQL_ADSDSOObject_LinkedServer_LDAP_Query",
             InitiatingProcessFileName =~ "sqlservr.exe" and RemotePort in (389, 636, 3268),
"Uncorrelated_SQLServer_LDAP_Egress_LowConfidence_Hunt",
             "Candidate_SQL_Or_IIS_Unusual_Port_Egress_Hunt"
         ),
         ArtifactDetail = strcat(RemoteIP, ":", tostring(RemotePort))
    | 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_erp_sqlagent_and_bpmn_rce.yml

SIGMA

```
title: SQL Server Agent Spawning CmdExec Dropper or ProgramData Tunnel Binary
id: 95a1b201-4e11-4c92-8a15-202611160501
status: experimental
description: >
  Detects Microsoft SQL Server Agent (sqlagent.exe) or Database Engine (sqlservr.exe) directly
  spawning command shells or file-transfer utilities referencing C:\ProgramData\, reverse tunnel
  flags, diagnostic collectors, or Defender tampering queries, matching AA_mcp.sql, AB_diag.sql,
  AC_diag2.sql, and AD_svc.sql while avoiding routine TSQL maintenance noise.
author: Odin by Brandefense
date: 2026-11-16
logsource:
  category: process_creation
  product: windows
detection:
  selection_parent:
    ParentImage|endswith:
      - '\sqlagent.exe'
      - '\sqlservr.exe'
  selection_child:
    Image|endswith:
      - '\cmd.exe'
      - '\powershell.exe'
      - '\pwsh.exe'
      - '\curl.exe'
      - '\certutil.exe'
      - '\rs.exe'
      - '\svc.exe'
  selection_cli:
    CommandLine|contains:
      - 'C:\ProgramData\'
      - '--reconnect'
      - '--port 9001'
      - '43.230.171.146'
      - 'start_svc.ps1'
      - 'diag.txt'
      - 'Get-MpThreat'
  condition: selection_parent and selection_child and selection_cli
falsepositives:
  - Rare third-party DBA monitoring scripts explicitly staged inside C:\ProgramData\
level: critical
tags:
  - attack.persistence
  - attack.execution
```

```
    - attack.t1505.001
  ---
  title: Epicor Kinetic IIS Worker Writing BPM Verification File or ASPX Webshell
  id: 95b2c302-5f22-4d03-9b26-202611160502
  status: experimental
  description: >
    Detects IIS worker processes (w3wp.exe) hosting Epicor Kinetic application pools performing
    in-process CLR File.WriteAllText writes of BPM verification markers (rcx_bpm.txt) or ASPX
    files inside inetpub\wwwroot following SQL injection of XAML CustomCodeAction payloads into
    Ice.BpDirective. Pair general .aspx writes with maintenance-window change controls.
  author: Odin by Brandefense
  date: 2026-11-16
  logsource:
    category: file_event
    product: windows
  detection:
    selection_iis_writer:
      Image|endswith: '\w3wp.exe'
      TargetFilename|contains: '\inetpub\wwwroot\'
    selection_artifact:
      TargetFilename|endswith:
        - '\rcx_bpm.txt'
        - '.aspx'
    condition: selection_iis_writer and selection_artifact
  falsepositives:
    - Planned Epicor Kinetic application deployments or hotfix rollouts writing .aspx files during maintenance
  windows
  level: high
  tags:
    - attack.persistence
    - attack.t1505.003
  ---
  title: Activiti or n8n Workflow Runtime Spawning Shell or Empty-Argv Pkexec Escalation
  id: 95c3d403-6a33-4e14-8c37-202611160503
  status: experimental
  description: >
    Detects Apache Activiti / Flowable (Java) or n8n (Node.js) workflow runtimes spawning interactive
    POSIX shells (excluding npm package lifecycle scripts on EDRs populating ParentCommandLine), or
    local privilege escalation via CVE-2021-4034 (PwnKit) where pkexec is invoked via zero-argument
    execve (empty or null CommandLine, never bare 'pkexec') from a GCONV_PATH/writable staging directory
    or workflow parent alongside GCONV_PATH=. environment staging or /tmp/pwned_bash execution (pair
    with Linux auditd EXECVE argc=0 rules where process_creation omits empty-argv telemetry).
  author: Odin by Brandefense
  date: 2026-11-16
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_workflow_parent:
      ParentImage|endswith:
        - '/java'
        - '/node'
      ParentCommandLine|contains:
        - 'activiti'
        - 'org.activiti'
        - 'flowable'
        - '/n8n'
        - 'n8n start'
        - 'task-runner'
    selection_workflow_shell_child:
      Image|endswith:
        - '/bash'
        - '/sh'
        - '/dash'
        - '/zsh'
        - '/curl'
    selection_pwnkit_empty_argv:
      Image|endswith: '/pkexec'
      CommandLine:
        - ''
        - null
    selection_pwnkit_cwd:
```

```
    CurrentDirectory|contains:
      - 'GCONV_PATH='
      - '/tmp/'
      - '/dev/shm/'
      - '/var/tmp/'
  selection_pwnkit_gconv_cli:
    CommandLine|contains:
      - 'GCONV_PATH=.'
      - 'CHARSET=PWNKIT'
  selection_pwnkit_dropper:
    Image|endswith: '/pwned_bash'
  condition: (selection_workflow_parent and selection_workflow_shell_child) or (selection_pwnkit_empty_argv and
(selection_pwnkit_cwd or selection_workflow_parent)) or selection_pwnkit_gconv_cli or selection_pwnkit_dropper
falsepositives:
  - Self-hosted n8n or Activiti servers intentionally configured to execute host shell commands
level: high
tags:
  - attack.execution
  - attack.privilege-escalation
  - attack.t1059.004
  - attack.t1068
```

suricata_erp_vcda_and_workflow_rce.rules

SURICATA

```
# Odin by Brandefense - Report #05 Suricata 7.x Enterprise ERP, VCDA/VCD & Workflow RCE Suite
# Classification: TLP:CLEAR
# Scope Note: Network delivery, HTTP SQLi staging, and exploitation-attempt signatures (inspect behind TLS
#             offload where HTTPS is used). Pair sid:9502602 staging detection with endpoint SQL Agent telemetry.

# 1. Epicor Kinetic Ice.BpDirective Source='DB' XAML CustomCodeAction or BpMethodCode SQL injection attempt
alert http any any -> $HOME_NET any (msg:"ODIN-ERP Epicor Kinetic Ice.BpDirective SQL Injection Attempt";
flow:established,to_server; http.request_body; content:"Ice.BpDirective"; nocase; fast_pattern; pcre:"/
(CustomCodeAction|BpMethodCode|'DB'|rcx_directive)/i"; metadata:created_at 2026_11_16, mitre_technique_id
T1505_001; classtype:web-application-attack; sid:9502601; rev:4;)

# 2. MSSQL Agent msdb.dbo.sp_add_jobstep CmdExec ProgramData dropper HTTP staging or SQLi delivery
alert http any any -> any any (msg:"ODIN-ERP MSSQL Agent sp_add_jobstep CmdExec Dropper HTTP Staging";
flow:established; file.data; content:"sp_add_jobstep"; nocase; fast_pattern; content:"CmdExec"; nocase; pcre:"/
(ProgramData|rs\.exe|svc\.exe|curl\.exe|--reconnect)/i"; metadata:created_at 2026_11_16, mitre_technique_id
T1505_001; classtype:trojan-activity; sid:9502602; rev:4;)

# 3. IIS ASPX RCX-211-OK or Epicor RCX-BPM-RCE webroot verification probe HTTP response
alert http $HOME_NET any -> any any (msg:"ODIN-ERP IIS ASPX RCX-211-OK or RCX-BPM-RCE Verification Marker";
flow:established,from_server; http.response_body; content:"RCX-"; fast_pattern; pcre:"/RCX-(211-OK|BPM-RCE)/";
metadata:created_at 2026_11_16, mitre_technique_id T1505_003; classtype:web-application-attack; sid:9502603; rev:
4;)

# 4. VMware Cloud Director Availability (VCDA) literal ..;/ semicolon path-traversal SSRF (ports
443/8443/8046/3030)
alert http any any -> $HOME_NET [443,8443,8046,3030] (msg:"ODIN-VCD VMware VCDA Pre-Auth Literal Semicolon Path-
Traversal SSRF"; flow:established,to_server; http.uri.raw; content:"..|3B|/"; fast_pattern; metadata:created_at
2026_11_16, mitre_technique_id T1190; classtype:attempted-admin; sid:9502604; rev:4;)

# 5. VMware VCDA POST /config/root-password Config-Secret reset or password-oracle attempt
alert http any any -> $HOME_NET any (msg:"ODIN-VCD VMware VCDA POST Root Password Reset with Config-Secret";
flow:established,to_server; http.method; content:"POST"; http.uri; content:"/config/root-password"; nocase;
fast_pattern; http.header_names; content:"Config-Secret"; nocase; metadata:created_at 2026_11_16,
mitre_technique_id T1190; classtype:attempted-admin; sid:9502605; rev:4;)

# 6. Apache Activiti BPMN 2.0 Plaintext Groovy ProcessBuilder RCE upload
alert http any any -> $HOME_NET any (msg:"ODIN-WF Apache Activiti BPMN 2.0 Plaintext Groovy ProcessBuilder RCE";
flow:established,to_server; http.request_body; content:"ProcessBuilder"; nocase; fast_pattern; content:"groovy";
nocase; metadata:created_at 2026_11_16, mitre_technique_id T1059; classtype:web-application-attack; sid:9502606;
rev:6;)

# 7. n8n workflow REST API JavaScript Code Node child_process command injection
alert http any any -> $HOME_NET any (msg:"ODIN-WF n8n Workflow REST API Code Node OS Command Injection";
flow:established,to_server; http.uri; content:"/workflows"; nocase; http.request_body; content:"n8n-nodes-
```

```
base.code"; fast_pattern; content:"child_process"; metadata:created_at 2026_11_16, mitre_technique_id T1059_007;
classtype:web-application-attack; sid:9502607; rev:4;)

# 8. VMware VCDA URL-encoded %3b/ semicolon path-traversal SSRF
alert http any any -> $HOME_NET [443,8443,8046,3030] (msg:"ODIN-VCD URL-Encoded Semicolon Path-Traversal SSRF";
flow:established,to_server; http.uri.raw; content:"%3b/"; nocase; fast_pattern; pcre:"/(\.\.|%2e%2e)%3b\//i";
metadata:created_at 2026_11_16, mitre_technique_id T1190; classtype:attempted-admin; sid:9502608; rev:2;)

# 9. Apache Activiti Modeler Stage-1 Pure-JVM or Base64 json_xml payload upload (StartNoneEvent + JVM recon/
Groovy)
alert http any any -> $HOME_NET any (msg:"ODIN-WF Activiti Modeler Stage-1 Pure-JVM or Base64 json_xml Upload";
flow:established,to_server; http.request_body; content:"json_xml"; fast_pattern; pcre:"/(StartNoneEvent|
U3RhcnROb25lRXZlbn|YXJ0Tm9uZUV2ZW50|dGFydE5vbmVFdmVu)/"; pcre:"/(System\.getProperty|InetAddress\.getLocalHost|
openStream|U3lzdGVtLmdldFByb3BlcnR5|c3RlbS5nZXRQcm9wZXJ0|eXN0ZW0uZ2V0UHJvcGVy|SW5ldEFkZHJlc3MuZ2V0TG9jYWxIb3N0|
ZXRBZGRyZXNzLmdldExvY2FsSG9z|bmV0QWRkcmVzcy5nZXRMb2NhbEhv|b3BlblN0cmVh|ZW5TdHJlYW|cGVuU3RyZWFt|Z3Jvb3Z5|b292e|
cm9vdn)/"; metadata:created_at 2026_11_16, mitre_technique_id T1059; classtype:web-application-attack; sid:
9502609; rev:3;)

# 10-13. Apache Activiti BPMN 2.0 Spring EL getScriptingEngines().evaluate RCE: plaintext (sid:9502610) and
Base64 alignments 0-2 (sid:9502611-9502613). Match the request body directly (direct XML, JSON-wrapped Base64 or
multipart); no filename is required.
alert http any any -> any any (msg:"ODIN-WF Apache Activiti BPMN 2.0 Plaintext Spring EL getScriptingEngines
RCE"; flow:established,to_server; http.request_body; content:"getScriptingEngines"; nocase; fast_pattern;
content:"evaluate"; nocase; distance:0; pcre:"/getScriptingEngines\s*\(\s*\)\s*\.evaluate/i"; metadata:created_at
2026_11_18, mitre_technique_id T1059; classtype:web-application-attack; sid:9502610; rev:3;)
alert http any any -> any any (msg:"ODIN-WF Apache Activiti BPMN 2.0 Base64 Spring EL getScriptingEngines RCE
(Align 0)"; flow:established,to_server; http.request_body; content:"Z2V0U2NyaXB0aW5nRW5naW5l"; fast_pattern;
pcre:"/(ZXZhbHVhdG|V2YWx1YXRl|ldmFsdWF0Z)/"; metadata:created_at 2026_11_19, mitre_technique_id T1059;
classtype:web-application-attack; sid:9502611; rev:1;)
alert http any any -> any any (msg:"ODIN-WF Apache Activiti BPMN 2.0 Base64 Spring EL getScriptingEngines RCE
(Align 1)"; flow:established,to_server; http.request_body; content:"dldFNjcmlwdGluZ0VuZ2luZX"; fast_pattern;
pcre:"/(ZXZhbHVhdG|V2YWx1YXRl|ldmFsdWF0Z)/"; metadata:created_at 2026_11_19, mitre_technique_id T1059;
classtype:web-application-attack; sid:9502612; rev:1;)
alert http any any -> any any (msg:"ODIN-WF Apache Activiti BPMN 2.0 Base64 Spring EL getScriptingEngines RCE
(Align 2)"; flow:established,to_server; http.request_body; content:"nZXRTY3JpcHRpbmdFbmdpbmVz"; fast_pattern;
pcre:"/(ZXZhbHVhdG|V2YWx1YXRl|ldmFsdWF0Z)/"; metadata:created_at 2026_11_19, mitre_technique_id T1059;
classtype:web-application-attack; sid:9502613; rev:1;)
```

yara_erp_vcda_and_workflow_toolkits.yar

YARA

```
/*
   Odin by Brandefense - Report #05 YARA Enterprise ERP, VCDA/VCD & Workflow RCE Rules
   Classification: TLP:CLEAR
*/

rule ODIN_Epicor_BpDirective_XAML_RCE_Injector {
    meta:
        author = "Odin by Brandefense"
        description = "Detects SQL injection scripts and migration telemetry targeting Epicor Kinetic
Ice.BpDirective / Ice.BpMethod"
        classification = "TLP:CLEAR"
    strings:
        $tbl1 = "Ice.BpDirective" ascii wide nocase
        $tbl2 = "Ice.BpMethod" ascii wide nocase
        $xaml1 = "<DirectiveDefinition2" ascii wide
        $xaml2 = "CustomCodeAction" ascii wide nocase
        $m1 = "Ice.UserFile.Update" ascii wide nocase
        $m2 = "Ice.AlternateKey.Update" ascii wide nocase
        $err1 = "Invalid column name 'DisplayOrder'" ascii wide
        $err2 = "RC_UNKNOWN" ascii wide
        $src = "'DB'" ascii wide
        $rcx1 = "rcx_directive" ascii wide nocase
        $rcx2 = "RCX-BPM-RCE" ascii wide
        $rcx3 = "RCX-211-OK" ascii wide
    condition:
        filesize < 250KB and (
            (($tbl1 or $tbl2) and ($xaml1 or $xaml2 or $m1 or $m2) and ($src or $rcx1 or $rcx2))
            or ($err1 and $err2)
```

```
            or $rcx3
        )
}

rule ODIN_MSSQL_Agent_CmdExec_And_ADSI_Toolkit {
    meta:
        author = "Odin by Brandefense"
        description = "Detects MSSQL Agent sp_add_jobstep CmdExec/xp_cmdshell persistence scripts and
ADSDSOObject AD recon queries"
        falsepositives = "Authorized DBA Active Directory ADSI linked-server scripts querying sAMAccountName"
        classification = "TLP:CLEAR"
    strings:
        $j1 = "sp_add_jobstep" ascii wide nocase
        $j2 = "CmdExec" ascii wide nocase
        $x1 = "xp_cmdshell" ascii wide nocase
        $p1 = "C:\\ProgramData\\" ascii wide nocase
        $p2 = "start_svc.ps1" ascii wide nocase
        $p3 = "rcx_" ascii wide nocase
        $ad1 = "sp_addlinkedserver" ascii wide nocase
        $ad2 = "ADSDSOObject" ascii wide nocase
        $ad3 = "OPENQUERY(ADSI" ascii wide nocase
        $ad4 = "sAMAccountName" ascii wide nocase
    condition:
        filesize < 150KB and (
            ($j1 and ($j2 or $x1) and ($p1 or $p2 or $p3))
            or ($x1 and $p3)
            or ($ad1 and $ad2 and ($p3 or ($ad3 and $ad4)))
        )
}

rule ODIN_VMware_VCDA_SSRF_And_VCD_Secret_Decryptor {
    meta:
        author = "Odin by Brandefense"
        description = "Detects VMware VCDA 4.7.4 pre-auth SSRF / Config-Secret scripts and standalone VCD 10.6.0
master-key decryption tools"
        classification = "TLP:CLEAR"
    strings:
        $v1 = "X-VCAV-Auth" ascii
        $v2 = "/config/root-password" ascii
        $v3 = "Config-Secret" ascii
        $v4 = "SourceIpAuthorizer.class" ascii
        $v5 = "/api-client-auth" ascii
        $v6 = "docs/..;/" ascii
        $v7 = ":8046/config" ascii
        $v8 = "/is-configured" ascii
        $c1 = "AES.MODE_ECB" ascii
        $c1b = "modes.ECB" ascii
        $c2 = "key_unwrap" ascii nocase
        $c2b = "aes_key_unwrap" ascii nocase
        $c3 = "PBKDF2" ascii nocase
    condition:
        filesize < 250KB and (
            2 of ($v1, $v2, $v3, $v4, $v5, $v6, $v7, $v8)
            or (($c1 or $c1b) and ($c2 or $c2b or $c3))
        )
}

rule ODIN_Activiti_BPMN_Groovy_And_n8n_Workflow_RCE {
    meta:
        author = "Odin by Brandefense"
        description = "Detects Apache Activiti BPMN 2.0 Groovy/EL/Modeler RCE payloads (plaintext and Base64) and
n8n post-exploitation / pwnkit toolkits"
        classification = "TLP:CLEAR"
    strings:
        $a1 = "groovy" ascii nocase
        $a1_b64 = "groovy" ascii base64
        $a2 = "new ProcessBuilder(" ascii
        $a3_plain = "getScriptingEngines().evaluate(" ascii
        $a3_b64 = "getScriptingEngines().evaluate(" ascii base64
        $a4_b64 = "StartNoneEvent" ascii base64
        $a5_plain = "System.getProperty(\"user.name\")" ascii
        $a5_esc = "System.getProperty(\\\"user.name\\\")" ascii
```

```
        $a5_b64 = "System.getProperty(" ascii base64
        $a6_b64 = "InetAddress.getLocalHost(" ascii base64
        $a7 = "openStream(" ascii
        $a8 = "InetAddress.getLocalHost(" ascii
        $n1a = "credentials_entity" ascii
        $n1b = "/rest/workflows" ascii
        $n1c = "n8n-auth" ascii
        $n1d = "child_process" ascii
        $n2b = "COPY (SELECT" ascii nocase
        $n2c = "<?php" ascii nocase
        $n2d = "eval($_POST[" ascii nocase
        $n3a = "GCONV_PATH=." ascii
        $n3b = "/usr/bin/pkexec" ascii
    condition:
        filesize < 250KB and (
            ($a1 and ($a2 or (($a5_plain or $a5_esc) and ($a7 or $a8))))
            or ($a4_b64 and ($a1_b64 or $a5_b64 or $a6_b64))
            or $a3_plain or $a3_b64
            or (($n1a or $n1b or $n1c) and $n1d)
            or ($n1b and $n1c)
            or ($n2b and ($n2c or $n2d))
            or ($n3a and $n3b)
        )
}
```

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/enterprise-middleware-under-attack/
Short Read · brandefense.io/odin-research/blog/attackers-are-staging-toolkits-for-erp-vmware-and-workflow-engines/
Defense Kit · brandefense.io/odin-research/defense-kit/