

What’s new in NSClient++

Two years. Forty-four releases.
The agent went quiet in 2019. This is what it has been doing since.

Michael Medin · NSClient++

nsclient.org · github.com/mickem/nscp



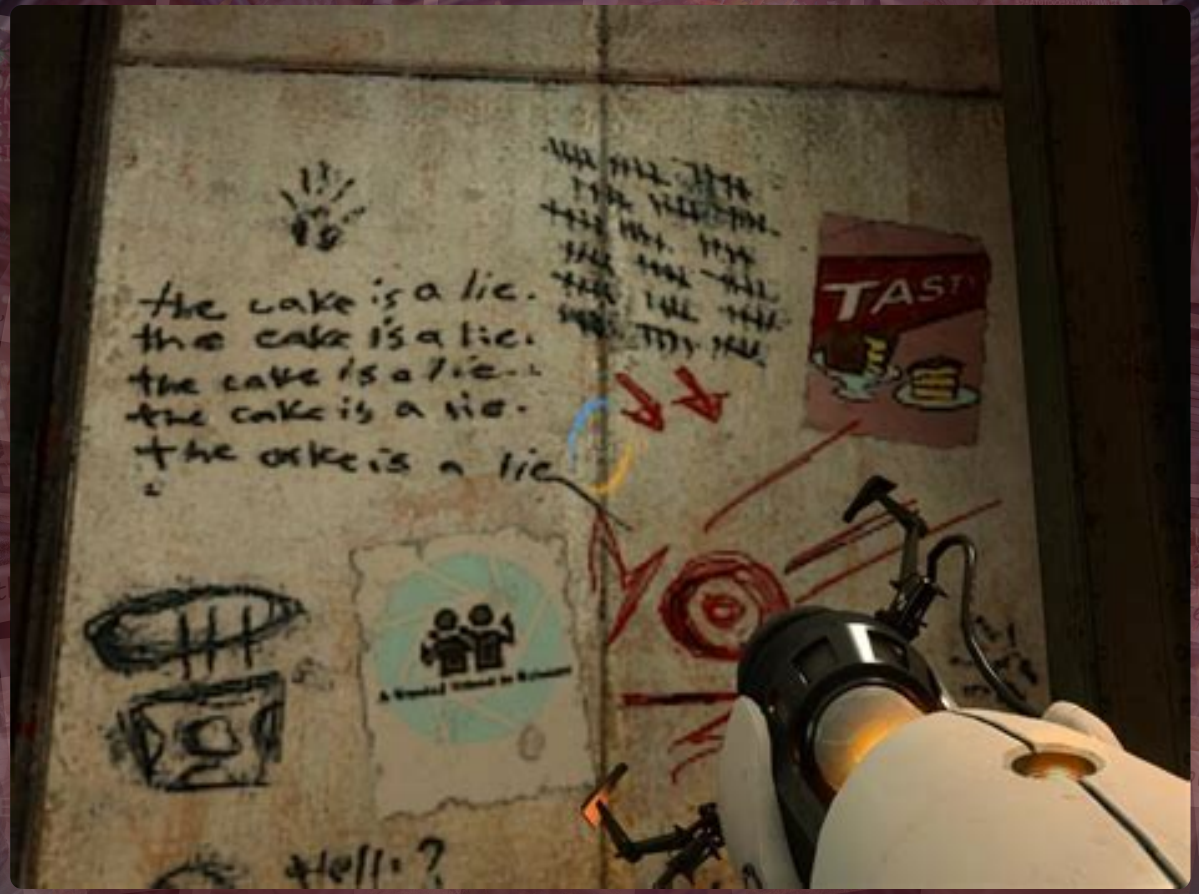
THURSDAY 19 NOVEMBER 2026

This is the worst day to give a presentation.

Half of you are in your hotel rooms, playing GTA VI.

The other half is probably playing it right here.

Grand Theft Auto VI cover art © Rockstar Games



Screenshot from Portal © Valve



1. The reboot

dealing with ten years of legacy



The NSClient++ update was a little late.

At least it was quicker than GTA VI.

Grand Theft Auto VI cover art © Rockstar Games



April 2018

The last release: NSClient++ 0.5.2.41

There were some sporadic updates...

...then dark for four years.

Why it went dark

The build machine was a Windows XP virtual machine

...and like every VM, it slowly died

A new one means a toolchain that still targets XP

...then updating every library

...then getting every library to build for XP as well

It never really went dark.
I started that climb several times, and each attempt failed somewhere along the chain.

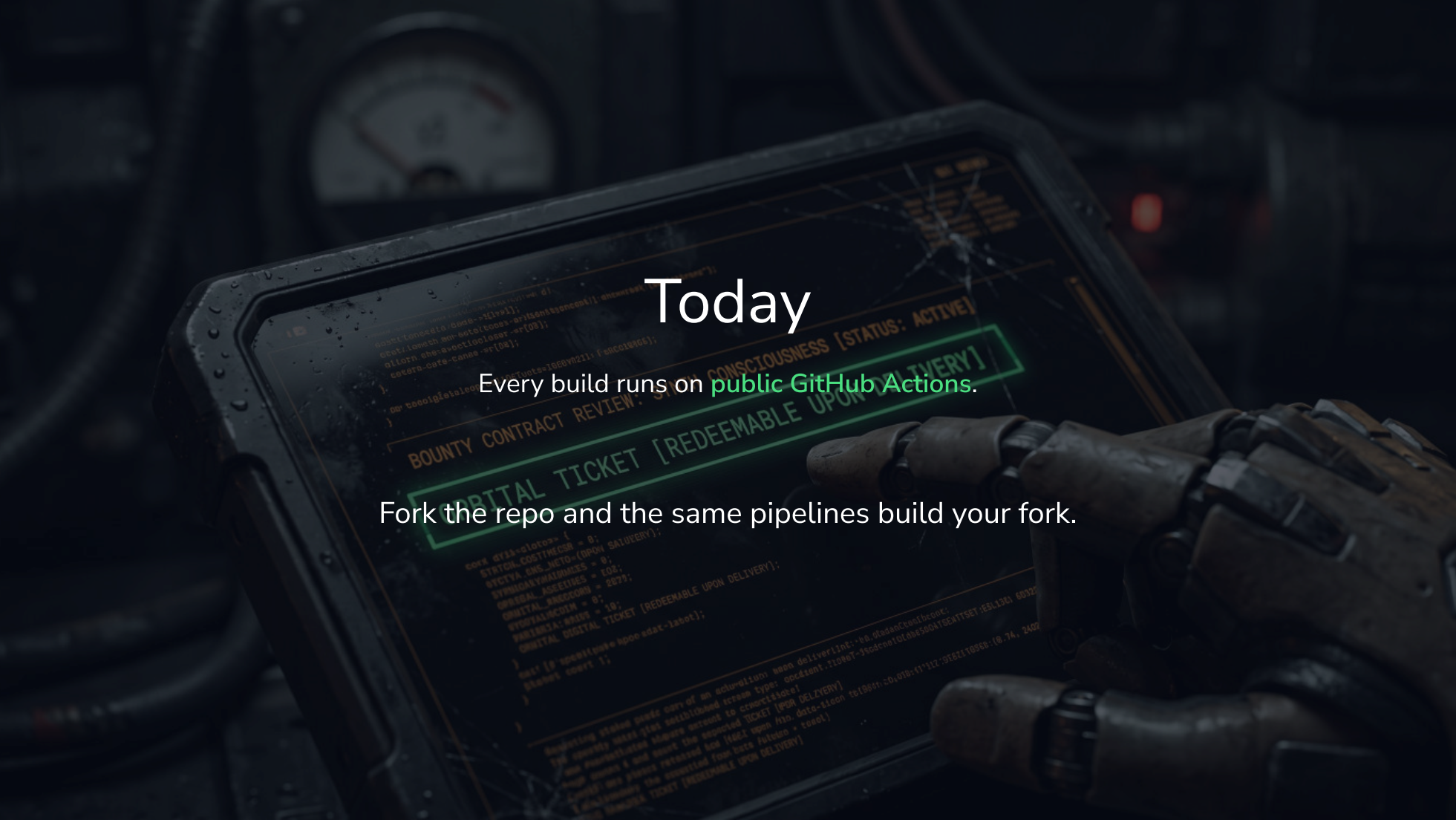
The solution was obvious: GitHub Actions

- The last resurrection attempt started with the pipelines, not the code
- ...and with a smaller scope: ship what builds, bring the rest back later

Gone in the first build `0.6.0.0`

- ✕ .NET modules
- ✕ Lua scripting
- ✕ The web UI (the server still answered)
- ✕ Older Windows
- ✕ NSCA encryption

They came back one release at a time.


Today
Every build runs on public GitHub Actions.
Fork the repo and the same pipelines build your fork.

Finding solid ground to stand on

The filter engine: answers that looked right

```
crit=state='hung' or count < 5
```

Went CRITICAL on the first row: while it was still counting, `count` was 1. `0.12.3`

The installer: whack-a-mole

Upgrading wiped your config.

"Possible fix for installer wiping config" `0.11.22`

"Yet another possible fix for installer deleting config on upgrade" `0.14.0`

The solution was again obvious: Tests

- ~9,200 unit tests in the C++ tree
- ~1,800 integration tests against a running agent: REST, and the *real* `check_nrpe`, `check_nt`, NSCA, Checkmk and Icinga

  There were tests before, too, but all NSClient++ talking to NSClient++. If NRPE was broken on one end, it was broken the same way on the other, and the test passed.
- ~30 Installer tests: that install over an existing config and check what's left
- Tests on real machines: fresh Windows and Linux VMs that install, enroll, run real MySQL and SQL Server

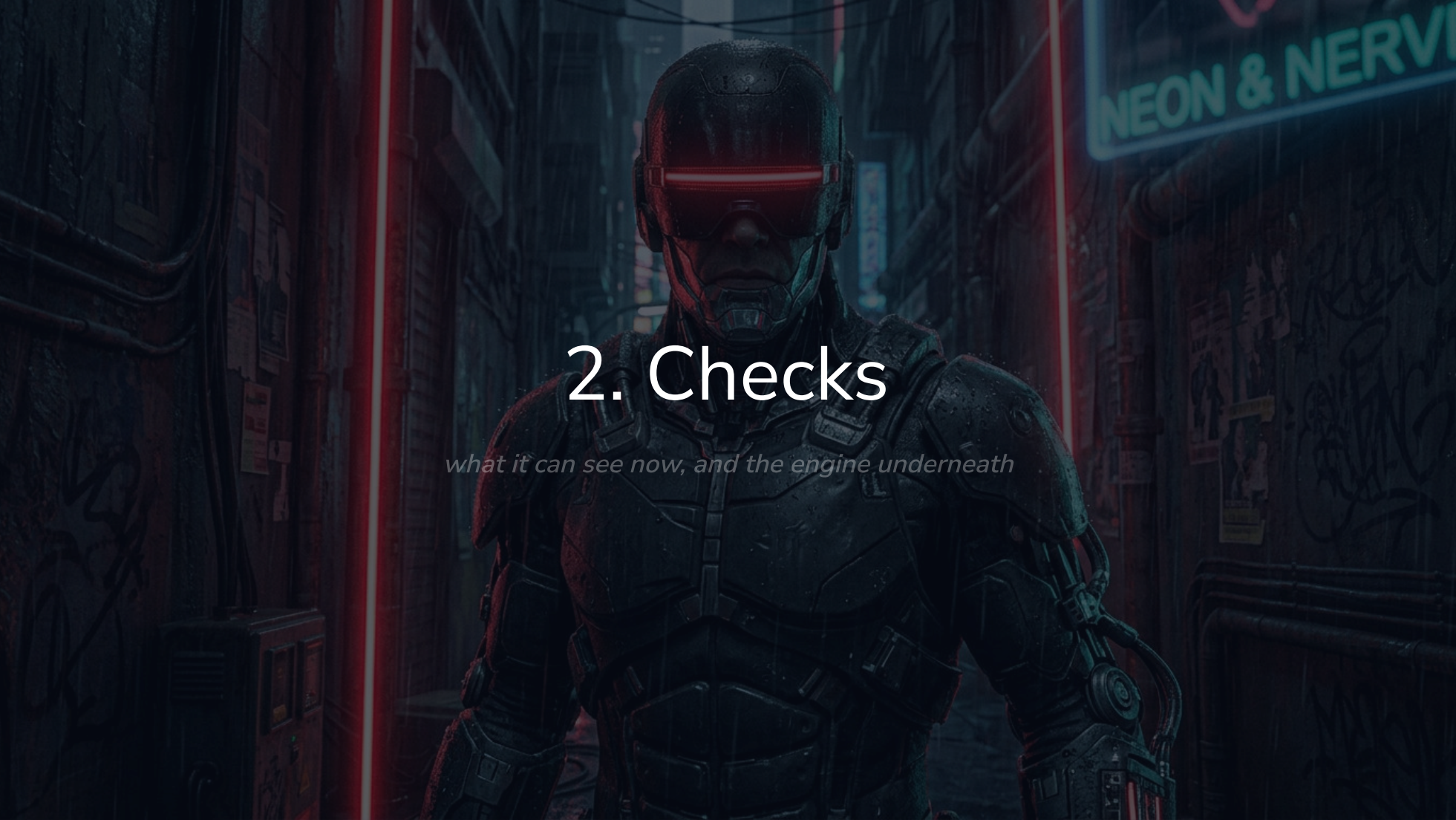
The tests run on the same public GitHub Actions. Fork the repo and you get them too.

44 releases in two columns

0.6	×5	Jan–Feb 2025	web UI back · semver
0.7	×1	Mar 2025	Lua back · TLS 1.2+ default
0.8	×1	Apr 2025	secrets in Credential Manager
0.9	×4	Jul–Aug 2025	REST cleanup · installer rework
0.10	×3	Nov–Dec 2025	2-way TLS NRPE · OpenSSL 3.5
0.11	×8	Jan–Apr 2026	`check_nsclient` · Linux packages
0.12	×3	May 2026	Checkmk · Icinga 2 · permissions
0.14	×2	Jul 2026	Linux parity · web refuses cleartext
0.15	×1	Aug 2026	SQL Server · 17 checks
0.16	×3	Aug 2026	MySQL · Docker · modern layout
0.17	×1	Aug 2026	IIS · RDS · number rendering
0.18	×2	Aug 2026	client modules that did nothing, fixed
0.19	×1	Sep 2026	a real `nscp test` prompt
0.20	×1	Sep 2026	.NET back · passive result cache
0.21	×1	Sep 2026	NSClient Fleet · access modes
0.22	×1	Sep 2026	OpenMetrics · Mod-Gearman
0.23	×1	Sep 2026	paths · enforced TLS verification
0.24	×2	Sep 2026	host facts · macOS · AD · K8s
0.25	×1	Oct 2026	config validation
0.26	×1	Oct 2026	NCPA · `check_largest_files` · script root
0.27	×1	Nov 2026	failed checks · diagnostics · Fleet reports

2,900 commits · 480 pull requests merged · 300 issues closed, some from 2014

After feedback from the community, I stopped doing minor updates for new features.



2. Checks

what it can see now, and the engine underneath

What it checked in 2018

System	`check_cpu` · `check_memory` · `check_pagefile` · `check_uptime` · `check_os_version` · `check_service` · `check_process` · `check_pdh` · `check_network`
Disk	`check_drivesize` · `check_files`
Logs	`check_eventlog` · `check_logfile`
Other	`check_tasksched` · `check_wmi` · `check_ping` · `check_nscp` · `check_nscp_version`

Everything else was `check_wmi`, `check_pdh`, or a script you wrote, versioned, and copied to every box.

What it checks now

18 → 121

check_tcp check_http check_dns check_ssh check_ntp_offset check_connections check_domain check_radius
check_nsclient_web_online check_apache_status check_nginx_status check_phpfpm_status check_tomcat_status check_certificate
check_firewall check_firewall_rules check_antivirus check_defender check_bitlocker check_secureboot check_nla
check_activation check_file_security check_local_accounts check_group_members check_users check_mssql check_mssql_query
check_mssql_databases check_mssql_backup check_mssql_jobs check_mssql_availability_groups check_mssql_blocking
check_mssql_counters check_mssql_sessions check_mssql_transactions check_mssql_waits
check_iis_sites check_iis_app_pool check_iis_request_queues check_rds_sessions check_rds_session_load
check_rds_broker check_rds_license check_cluster_groups check_cluster_resources check_cluster_networks
check_nps_auth check_nps_accounting check_nps_counters check_load check_cpu_utilization check_cpu_frequency check_kernel_stats
check_kernel_memory check_swap_io check_temperature check_battery check_hardware check_hostname check_installed_software
check_os_updates check_pending_reboot check_patch_age check_w32time check_registry_key check_registry_value check_printqueue
check_printjobs check_process_history check_process_history_new check_disk_io check_disk_health check_disk_write check_mount
check_single_file check_uncpath check_share check_storagepool check_shadowcopy check_largest_files check_docker
check_docker_info check_docker_stats check_docker_restarts check_docker_df check_mysql check_mysql_query check_kubernetes
check_nodes check_pods check_workloads check_ad_replication check_kdc check_secure_channel check_hyperv_host
check_hyperv_vms check_hyperv_cpu check_nscp_update

Four checks that used to be `check_wmi`

```
# check_defender — signatures older than a week
check_wmi namespace=root\Microsoft\Windows\Defender
  "query=SELECT AntivirusSignatureAge FROM MSFT_MpComputerStatus" "crit=AntivirusSignatureAge > 7"

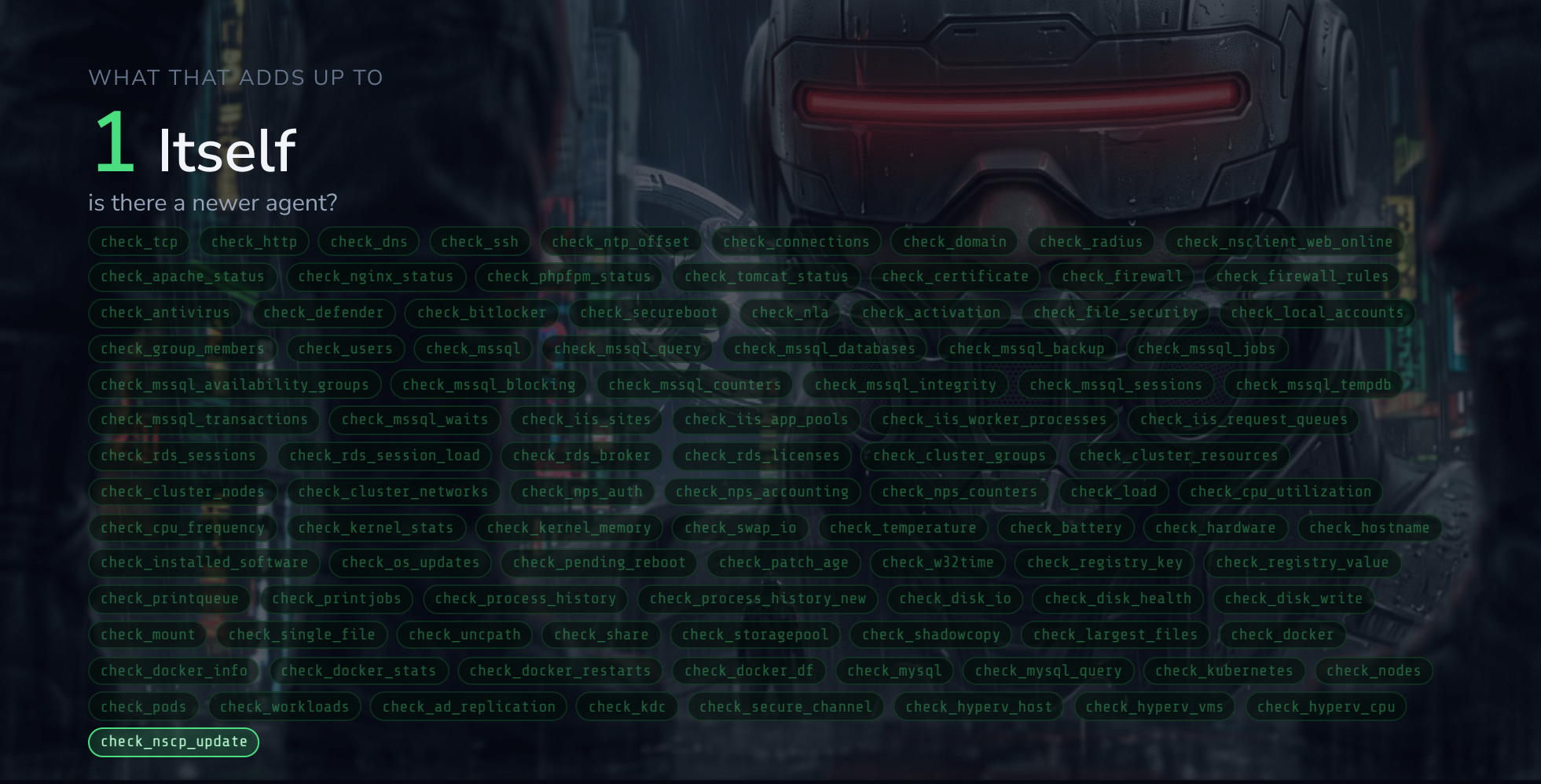
# check_bitlocker — a volume that isn't protected
check_wmi namespace=root\CIMV2\Security\MicrosoftVolumeEncryption
  "query=SELECT DriveLetter, ProtectionStatus FROM Win32_EncryptableVolume" "crit=ProtectionStatus != 1"

# check_activation — Windows not licensed
check_wmi "query=SELECT Name, LicenseStatus FROM SoftwareLicensingProduct WHERE PartialProductKey IS NOT NULL"
  "crit=LicenseStatus != 1"

# check_hyperv_vms — a VM that isn't running
check_wmi namespace=root\virtualization\v2
  "query=SELECT ElementName, EnabledState FROM Msvm_ComputerSystem WHERE Caption='Virtual Machine'"
  "crit=EnabledState != 2"
```

All of these worked in 2018. The new check adds **names instead of status codes**, defaults, docs and samples.

Convenience, not capability.



WHAT THAT ADDS UP TO

1 Itself

is there a newer agent?

check_tcp check_http check_dns check_ssh check_ntp_offset check_connections check_domain check_radius check_nsclient_web_online
check_apache_status check_nginx_status check_phpfpm_status check_tomcat_status check_certificate check_firewall check_firewall_rules
check_antivirus check_defender check_bitlocker check_secureboot check_nla check_activation check_file_security check_local_accounts
check_group_members check_users check_mssql check_mssql_query check_mssql_databases check_mssql_backup check_mssql_jobs
check_mssql_availability_groups check_mssql_blocking check_mssql_counters check_mssql_integrity check_mssql_sessions check_mssql_tempdb
check_mssql_transactions check_mssql_waits check_iis_sites check_iis_app_pools check_iis_worker_processes check_iis_request_queues
check_rds_sessions check_rds_session_load check_rds_broker check_rds_licenses check_cluster_groups check_cluster_resources
check_cluster_nodes check_cluster_networks check_nps_auth check_nps_accounting check_nps_counters check_load check_cpu_utilization
check_cpu_frequency check_kernel_stats check_kernel_memory check_swap_io check_temperature check_battery check_hardware check_hostname
check_installed_software check_os_updates check_pending_reboot check_patch_age check_w32time check_registry_key check_registry_value
check_printqueue check_printjobs check_process_history check_process_history_new check_disk_io check_disk_health check_disk_write
check_mount check_single_file check_uncpath check_share check_storagepool check_shadowcopy check_largest_files check_docker
check_docker_info check_docker_stats check_docker_restarts check_docker_df check_mysql check_mysql_query check_kubernetes check_nodes
check_pods check_workloads check_ad_replication check_kdc check_secure_channel check_hyperv_host check_hyperv_vms check_hyperv_cpu
check_nscp_update

The engine underneath

Long-standing filter edge cases, fixed

- `warn=` / `crit=` evaluated after iteration — `count` is stable, mixed expressions work when nothing matches `0.12.3`
- text-vs-number comparisons are numeric — `value > 90` now matches `100` `0.17.0`
- fractional thresholds mean what they say — `count > 2.5`, `working_set > 1.5g` `0.17.0`
- an error inside a syntax template is reported, not rendered blank
- `empty-state` is explicit everywhere; a missing path is UNKNOWN, not a silent OK `0.11.33`
- `perf-config unit:` converts the value and its thresholds; it used to change only the label `0.17.0`

Functions, in thresholds too `0.12.5`

```
check_pdh counter=memory_bytes
  "warning=convert_bytes(value, 'MB') > 500"
  "detail-syntax=%(alias) = %(format_bytes(value))"
```

Output your frontend can use

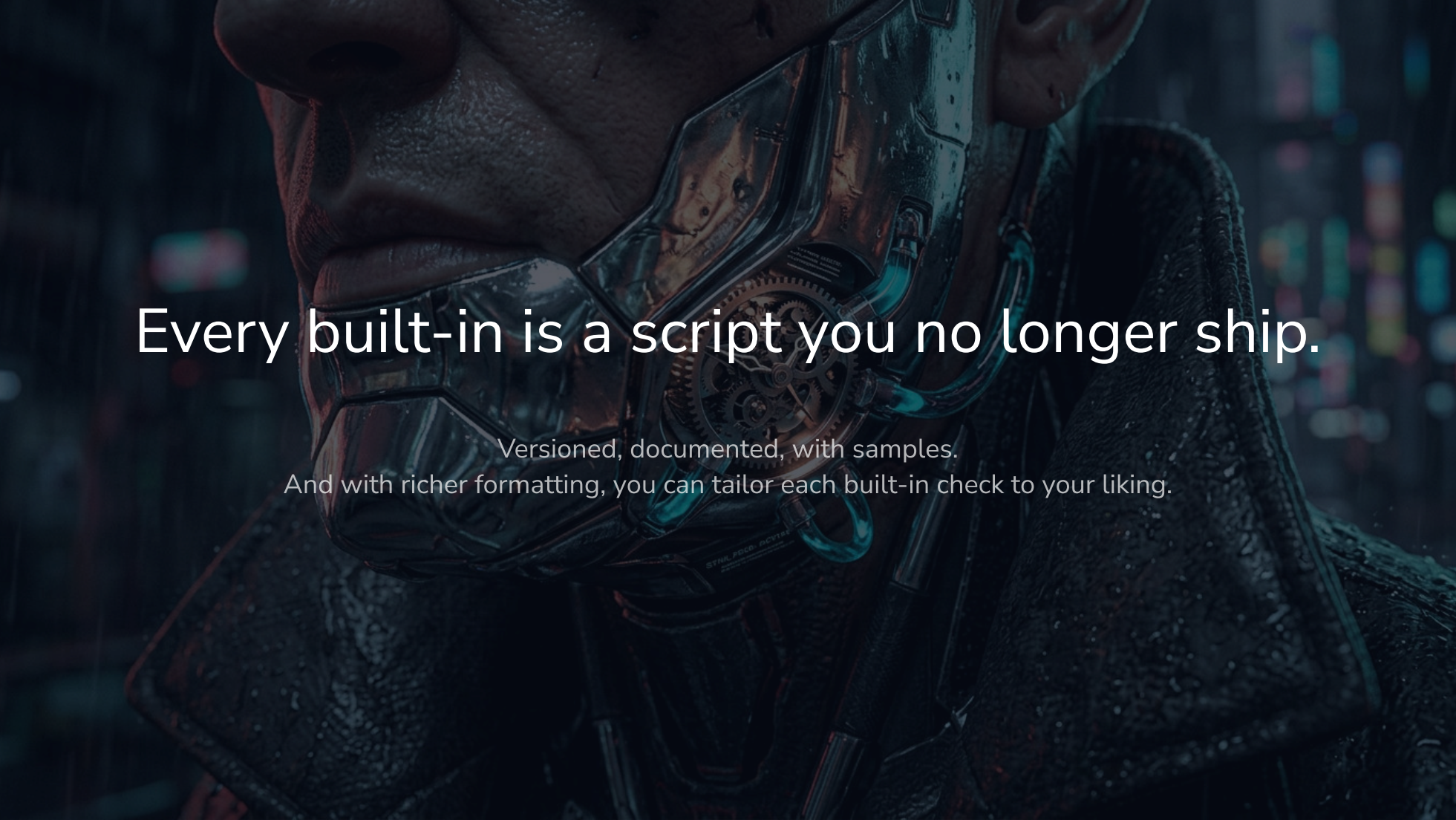
- `list-separator` — one item per line, which Nagios-style UIs show as summary + long output `0.15.0`
- Nagios range syntax in perfdata thresholds — `1:10`, `~:5`, `@10:20` `0.12.6`
- `warmup-state` — a collector-backed check reports OK instead of UNKNOWN for its first sample after a restart `0.25.0`

Number formatting `0.17.0`

```
check_drivesize drive=/ show-all=true
OK /: 141.085GB/0.983TB used
check_drivesize drive=/ show-all=true decimals=2 byte-unit=GB
  decimal-separator=, thousands-separator=.
OK /: 141,09GB/1.006,85GB used
```

Modernizing the existing checks

Check	What changed
`check_service`	overhaul `0.11.29`: trigger-start and clean-exit services are OK, `exit_code` keyword, Server 2025 service list · enumerates correctly on busy hosts `0.12.1` · a filter that matched nothing crashed the agent — fixed `0.19.0`
`check_process`	sees protected and cross-user processes as NETWORK SERVICE · case-insensitive real-time path · correct per-process CPU with `delta=true` `0.12.1` `0.13.2`
`check_cpu`	standard `user` / `system` / `idle` keywords · PDH fallback for the "negative load on 12+ cores" machines `0.7.0` · the PDH subsystem overhauled: crashes, counter retry, English-name fallback on localised Windows `0.12.6`
`check_files`	`max-depth=0`, non-ASCII paths, missing path → UNKNOWN, junction loops skipped `0.11.33` · checksums `0.14.0`
`check_eventlog`	`opcode` keyword, regex and `in` on numeric fields, `scan-range` fixed, event-ID matching fixed `0.11.30`
`check_logfile`	bookmarks — report a line once, not until it rotates · tailing `max-lines=N` without reading the file `0.16.1`
`check_network`	NIC teams, human-readable units, speed and percentages `0.12.6`
`check_drivesize`	one unavailable volume no longer aborts the check `0.11.31` · inodes `0.14.0` · `full_in` `0.16.0`
the collector	fixed cadence instead of `sleep 1` `0.11.27` · slow WMI providers on their own thread, so they can't stall `check_cpu` `0.15.0`


Every built-in is a script you no longer ship.
Versioned, documented, with samples.
And with richer formatting, you can tailor each built-in check to your liking.



3. Platforms

"it compiles on Linux"

Linux: from "it compiles" to near-parity

`0.11.9`	Jan 2026	first `.deb` / `.rpm` attached to a release — *"early, use with care"*
`0.11.24`	Mar 2026	official packages for Ubuntu and Rocky · Python and Lua scripts work · integration tests on both
`0.11.30`	Apr 2026	ARM64 for Ubuntu 24.04 and Rocky 9/10
`0.12.3`	May 2026	real-time metrics — the background CPU/memory/disk/network sampling Windows had for years
`0.14.0`	Jul 2026	native system checks from `/proc` and `/sys` · systemd-aware `check_service` · real-time filters · `CheckDisk` with inodes, LVM mapping, `check_mount` · FHS layout · `.deb` / `.rpm` for `/usr`
`0.22.0`	Sep 2026	Raspberry Pi / Debian 13 arm64
`0.24.0`	Sep 2026	service sandboxed under its own user and group

```
check_load "warn=load5 > 4" "crit=load5 > 8"
check_service service=mysql "warn=rss > 1G" "crit=rss > 2G"
check_drivesize drive=/ "warn=inodes_used_pct > 85" "crit=inodes_used_pct > 95"
check_mount mount=/data fstype=ext4
```

Which OS would you like to see packages for? Please let me know.

macOS, ARM, and yes, Windows XP

macOS `0.24.0`

- Apple silicon `.pkg` and `.tar.gz`
- runs as a launchd daemon
- all 21 Linux `CheckSystem` commands
- `check_service` checks launchd jobs
- experimental — not signed or notarized yet

ARM

- Linux ARM64: Ubuntu, Rocky `0.11.30`
- Raspberry Pi OS 64-bit `0.22.0`
- Windows ARM64 MSI is back `0.22.0` (no PythonScript, no bundled VC runtime)

Legacy Windows `0.10.0`

- a `Win32-legacy-xp` build
- verified on Server 2008
- *in theory* runs on XP
- the MSI may not — use the zip



Install from the package manager you already use: `winget` · Scoop `0.13.0`

Windows: a more secure layout

Legacy — the default

Everything under `C:\Program Files\NSClient++\`

- `nsclient.ini`, the TLS private key, the fleet identity
- readable by any logged-in user
- the service needs write access to its own install folder

Modern — opt-in `0.16.2`

Program stays put. Writable state moves to `%ProgramData%\NSClient++\`

- DACL: SYSTEM + Administrators only, inheritance broken
- config, `security\`, `fleet\`, `log\`, `cache\`
- junctions refused — the lockdown can't be redirected `0.19.0`

```
nscp settings --migrate-layout modern --dry-run     # see what would move
nscp settings --migrate-layout modern               # move it, record the choice
msiexec /i NSCP-0.26.0-x64.msi LAYOUT=modern        # or at install / upgrade
```

Legacy stays the default; an upgrade keeps whatever layout the host has. Modern is recommended for new installs.



Same agent, same syntax. Three platforms.

Nearly half of the new checks run on Windows and Linux alike —
`check_load` on Windows is *synthesised* from the run queue, because Windows never had one.

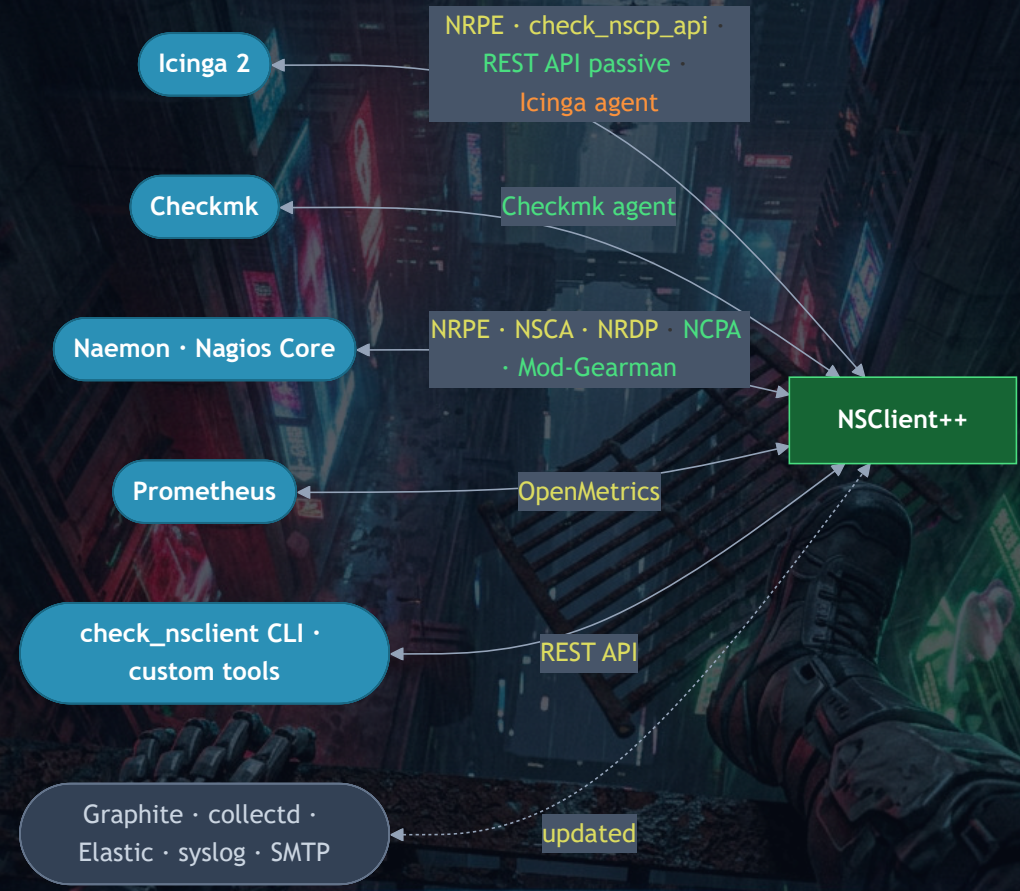
Load averages · mount points · swap I/O · kernel stats · thermal zones · Docker · MySQL · Apache, nginx, PHP-FPM, Tomcat status pages —
Unix ideas that landed on Windows, not Windows checks ported to Linux.



4. Talking to your monitoring system

Icinga 2, Checkmk, Naemon, Nagios, Prometheus — whatever you run, there's a door

One agent, every core in the room



■ new · ■ major update · ■ coming soon

Mod-Gearman without an open port `0.22.0`

```
[/modules]
GearmanClient = enabled

; the worker half: pull this host's checks
[/settings/gearman/worker]
; …address, key, queues — see the reference

; the result half: replaces NSCA
[/settings/gearman/client]
channel = GEARMAN
hostname = auto

[/settings/gearman/client/targets/default]
address = gearmand.example.com:4730
key = <shared secret>
queue = check_results

[/settings/scheduler/schedules/default]
channel = GEARMAN
interval = 5m
```

- `mode = agent` — answers for the host it runs on
- `mode = proxy` — runs a whole hostgroup's checks from one box
- the core keeps scheduling exactly as before; the agent connects out to `gearmand`
- results travel the same outbound connection — one daemon fewer, no NSCA, no mcrypt

Experimental, and not yet run at scale. Verified end to end against both flavours — ConSol's Mod-Gearman (Naemon) and Nagios-Mod-Gearman (Nagios Core). Put it on a slice of hosts first.

Icinga 2 and Checkmk, in config

Icinga 2 — passive, direct `0.12.1`

```
[/modules]
IcingaClient = enabled

[/settings/IcingaClient/targets/default]
address = https://icinga2.example.com:5665
username = nscp
password = secret
hostname = ${hostname}
```

```
nscp client --module IcingaClient \
  --command submit_icinga --command heartbeat \
  --result 0 --message "hello" --ensure-objects
```

Active checks via `check_nscp_api` keep working — its query-string password path is allow-listed by User-Agent `0.12.4`

Checkmk — a native agent `0.12.3`

```
[/modules]
CheckMKServer = enabled
CheckSystem = enabled    ; mem, ps, services
CheckDisk = enabled      ; df

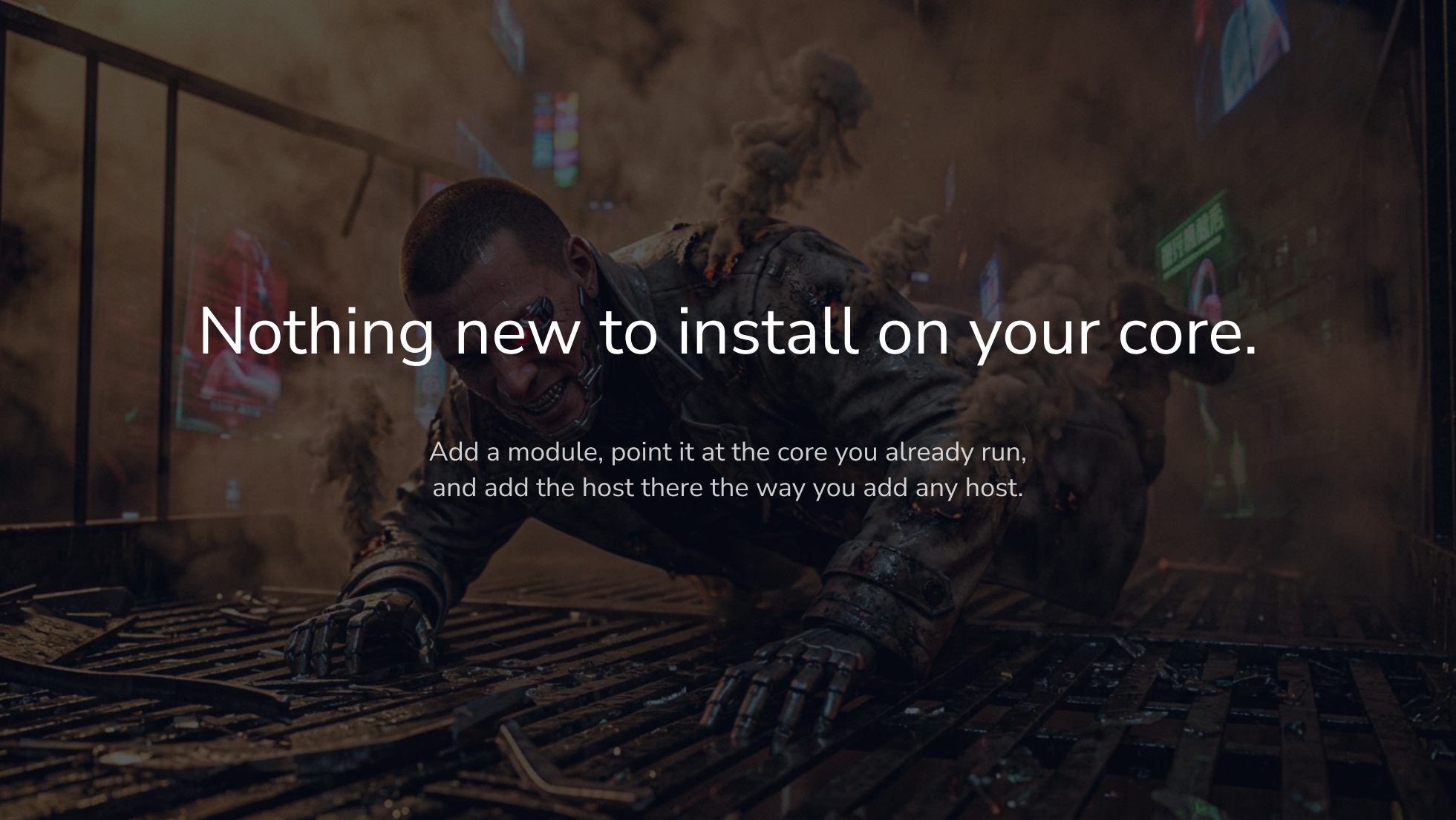
[/settings/check_mk/server]
port = 6556
allowed hosts = 127.0.0.1, 10.0.0.5
mrpe channel = check_mk-mrpe
local channel = check_mk-local
```

Register the host with `tag_agent = cmk-agent`, run discovery.

No key: agent encryption isn't supported yet, so the site talks plain text and `allowed hosts` is the gate. The check_mk *client* also verifies TLS by default now `0.23.0`

The other new doors

		Since
OpenMetrics	`/api/v2/openmetrics` is a conformant, typed, labelled exposition. `system_cpu_idle_percent{core="0"}` instead of `system.cpu.core 0.idle` . Every name changed.	`0.22.0`
Passive results over REST	the WEB server caches results until something polls `GET /api/v2/results` ; `check_nsclient results feed` hands them to Nagios as passive — one active check per host, no receiver.	`0.20.0`
NCPA	`NCPAServer` answers Nagios' own `check_ncpa.py` on 5693: token auth, `allowed hosts` , arguments off unless allowed. Serves `plugins/` (any check, alias or script); the `cpu/` , `memory/` , `disk/` tree comes later. *Experimental.*	`0.26.0`
NSCA-NG	a new, hardened `NSCANgClient` : PSK, AEAD-first ciphers. *Not in the Windows installer since 0.24.1: antivirus engines flag its DLL.*	`0.12.3`
NRDP	now supports a proxy	`0.11.33`
Elastic	fixed: works with Elasticsearch 8, with TLS verification and API keys	`0.18.1`
collectd · syslog	fixed: collectd end to end `0.19.0` · syslog, long-outstanding issues `0.17.0`	



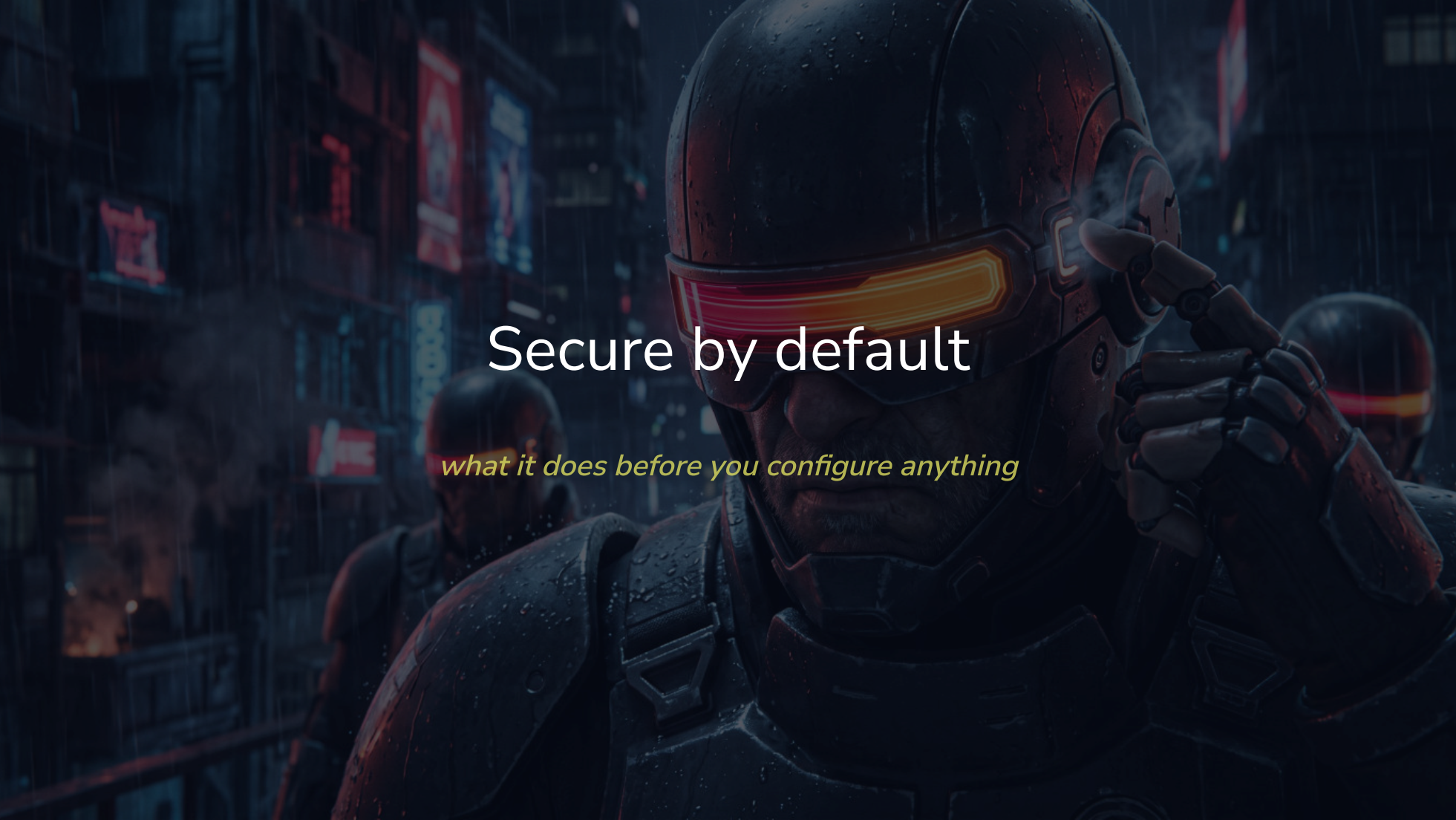
Nothing new to install on your core.

Add a module, point it at the core you already run,
and add the host there the way you add any host.



5. Security

Actually taking security seriously



Secure by default

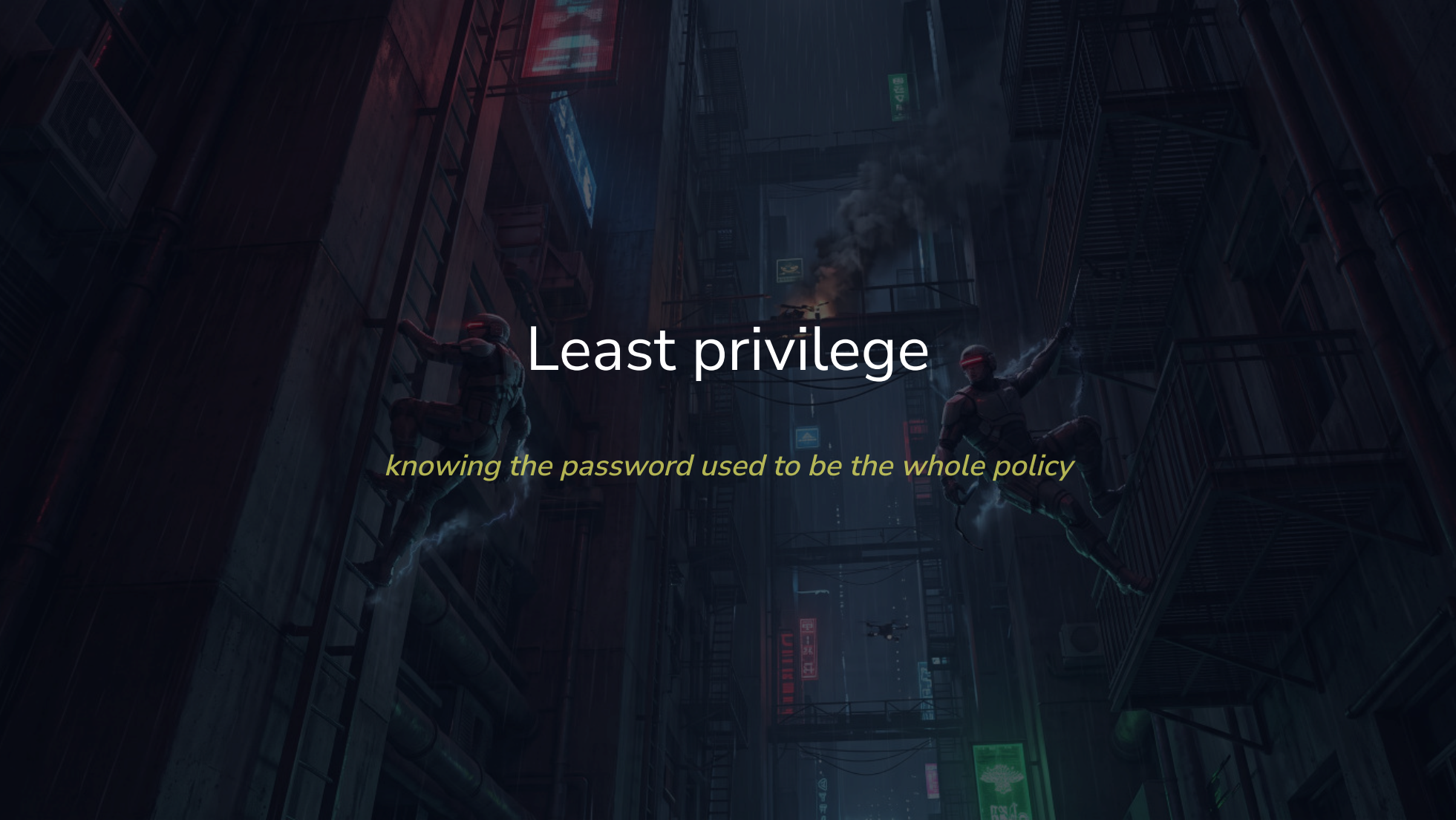
what it does before you configure anything

Defaults that flipped

EACH OF THESE USED TO BE THE PERMISSIVE WAY ROUND

sensitive values are redacted everywhere they're read

REST settings reads, `nscp settings --list`, the `nscp test` settings dump, settings urls in the log — all `***` `0.16.2` `0.21.0`. The `legacy` WEB role is no longer seeded `0.16.2`


Least privilege
knowing the password used to be the whole policy

Who may run what `0.12.6`

```
[/settings/permissions]
enabled = true
log allows = true        ; audit mode first

[/settings/permissions/policies]
; subject = module[:principal]   object = module.command
WEBServer:admin       = *
WEBServer:monitoring  = CheckSystem.check_cpu, CheckDisk.*
NRPEServer:icinga-cn  = CheckSystem.*, CheckDisk.*
```

- Rules are additive. No match → deny. There is no permissive fallback.
- `client identity source = cn` — the NRPE client certificate's CN is the principal. Your Icinga master's cert is its credential, scoped to the checks you granted.
- `allow exec` is a separate, single switch
- `disable admin user` — a monitoring-only web UI with no reconfiguration surface at all
- Off by default. Audit mode logs what *would* match before you enforce.

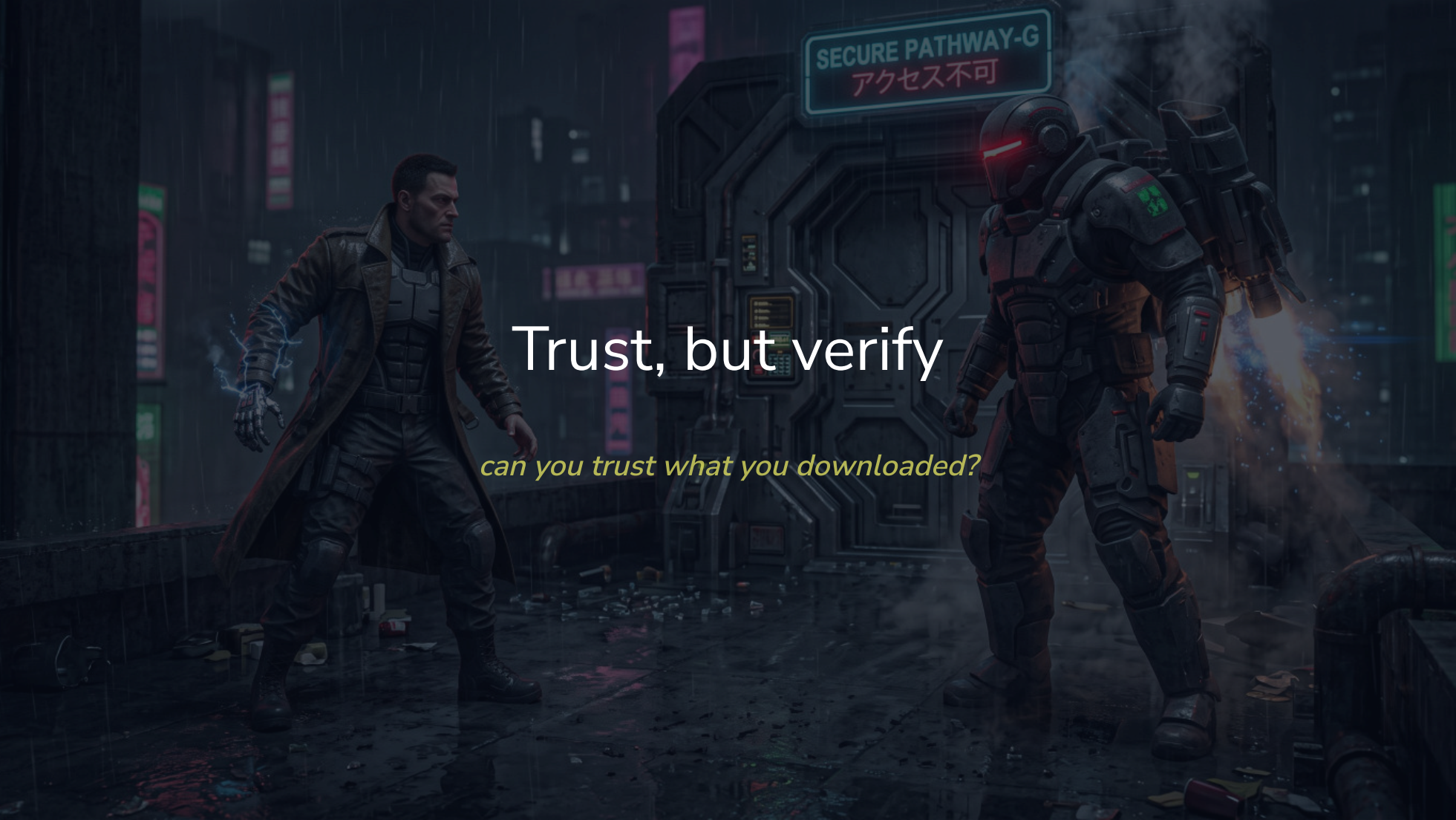
Reading less `0.21.0` `0.24.0`

A check that reads whatever its argument names is a read primitive for any caller who may choose the argument.

Mode	Meaning
`any`	the previous behaviour — default, nothing changes on upgrade
`predefined`	the caller may only use the names *you* configured
`allowed`	the argument must match an allow list

Applies to `check_logfile` · `check_files` · `check_single_file` · `check_disk_write` · `check_wmi` · `check_pdh` · `check_registry_key` · `check_registry_value` · `check_eventlog`

- Two WEB roles that can't be widened: `restricted` runs the checks the agent defines and refuses any argument — NRPE's `allow arguments = false`, for REST. `metrics` scrapes and nothing else.
- Bounded work per request `0.24.0`: nesting stops at 16, `check_multi` at 128 commands, a regex at 30 s, `check_logfile` at 64 MiB. A caller used to be able to pin a worker for minutes.
- Credentials stay with their target `0.19.0`: a query can no longer move a submission's `host=` and take the target's token with it.



Trust, but verify

can you trust what you downloaded?

Trusting the binary

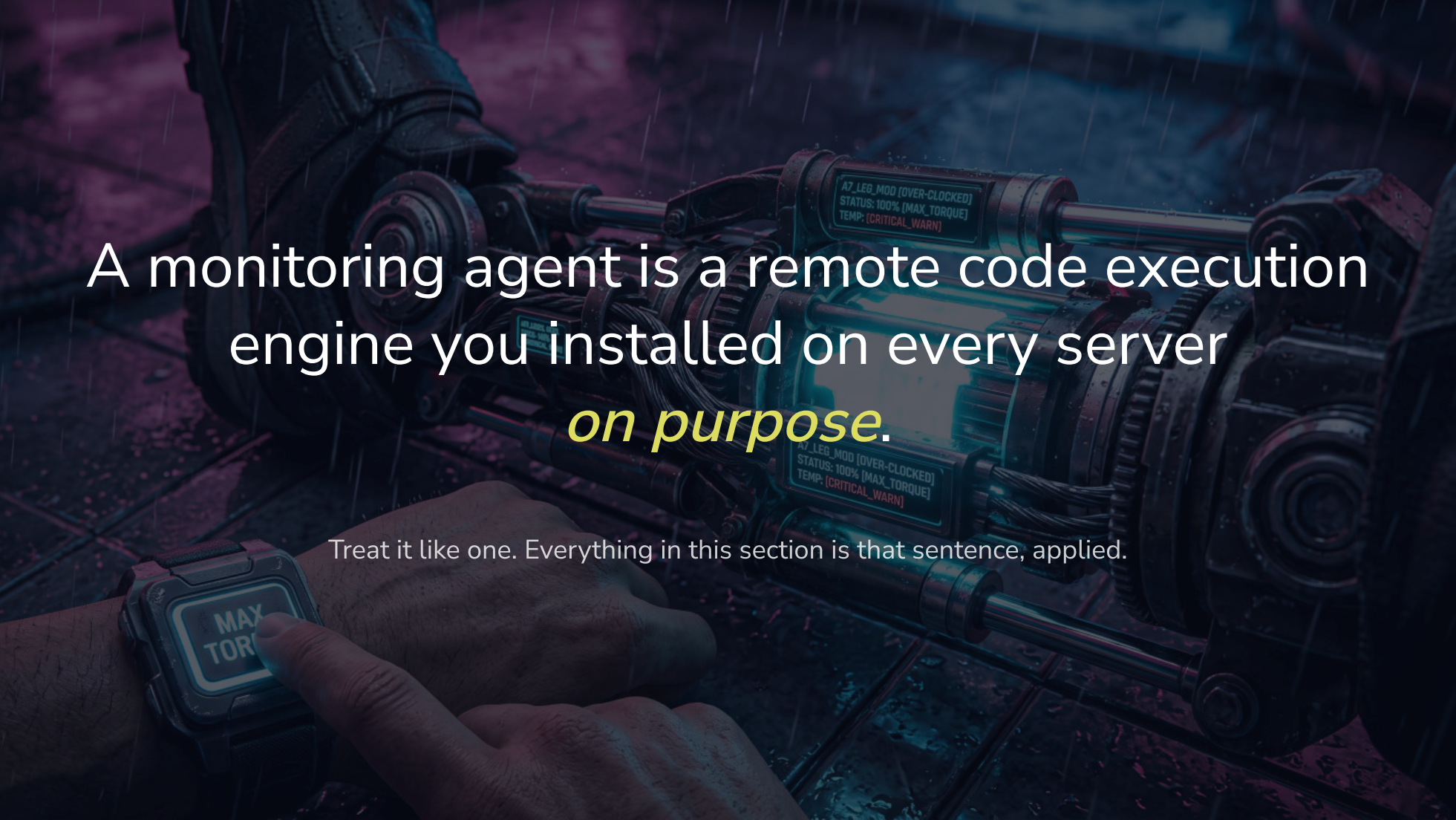
The releases

- Every shipped Windows binary signed — DLLs, executables, the MSI `0.24.0`
- a CycloneDX SBOM with each Windows release
- GitHub build attestations on every artifact
- dependencies pinned to digests; build actions pinned
- bundled OpenSSL tracked release by release — 3.5.9 `0.25.0`

```
gh attestation verify NSCP-0.26.0-x64.msi --repo mickem/nscp
```

Securing it

- Securing NSClient++, one page in the docs: *nsclient.org/docs/setup/securing*
- per protocol, the service account, passwords, the permission policy, what a check may read — and a quick checklist at the end
- 70+ hardening notices, one per change, each with affected versions, severity and the action: *nsclient.org/docs/security/notices*
- Antivirus false positives on 0.22–0.24: reported, documented, and the one module still flagged left the MSI `0.24.1`



A monitoring agent is a remote code execution engine you installed on every server *on purpose*.

Treat it like one. Everything in this section is that sentence, applied.



6. Operating it

a real prompt, a real CLI, a config that tells you when it's wrong

`nscp test` is a real prompt 0.19.0 0.21.0

- line editing, persistent per-user history
- tab completion against the command registry — `load check⇥` corrects to `load Check…`
- an unknown query or module name turns red before you press enter
- `filter=` , `warning=` , `detail-syntax=` are painted as you type, from the check's own keyword list 0.22.0
- log lines redraw *around* what you're typing
- `desc check_service` — defaults, the bare command line, what an alias runs
- `keywords check_drivesize` — every filter keyword

```
> load CheckSystem
> check_service "filter=start_type = 'auto'" \
      "crit=state != 'started' and exit_code != 0"
L   cli OK: All 71 service(s) are ok.
> debug=true check_drivesize drive=*
ignored  HarddiskVolume1   (filter)
match    C:                ok
warning  D:                free < 20%
```

`debug` returns a per-item verdict — `ignored` / `match` / `warning` / `critical` — instead of a wall of log 0.10.7

`check_nsclient` — the agent from anywhere 0.11.6

```
$ check_nsclient nsclient queries execute-nagios check_cpu
OK: CPU load is ok.|'total 5m'=10%;80;90 'total 1m'=10%;80;90

$ check_nsclient --output yaml nsclient \
      modules show CheckSystem
id: checksystem
title: CheckSystem
enabled: true
loaded: true

$ check_nsclient nsclient modules use CheckNet
Successfully loaded and enabled module CheckNet
```

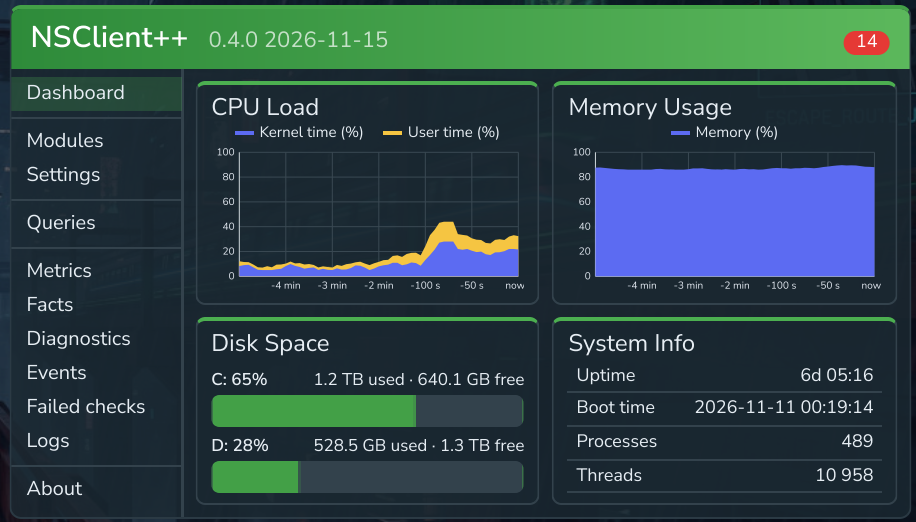
A stand-alone client, Windows and Linux, talking to the REST API. Now at 1.1.3

- profiles with credentials in the OS credential store — `auth login` once
- `modules list|use` · `queries execute` · `settings list|set` · `logs list` · `facts show`
- output as text, json, yaml or csv — pipe it into anything
- an interactive client for when you want the prompt without the RDP
- `results feed` — hand a host's cached passive results to Nagios 1.1.0
- `nrpe query` — speaks NRPE too 1.1.4

Bundled with the agent and released on its own. Growing into a universal client: NRPE is in, NSCA, NRDP and others are on the list.

The web UI, rebuilt

- Rewritten in a modern framework `0.6.4`; then a new theme, filterable lists, a settings diff `0.11.32`
- Dashboard: CPU, memory, disk and network widgets, PDH counters, real-time filters `0.11.26` `0.12.3`
- Filter expressions highlighted as you type `0.22.0`
- A Facts page `0.24.0` and a Configuration problems banner `0.25.0`
- Sessions survive a restart, stored only as hashes · failed logins back off `0.23.0` `0.19.0`
- Roles: `full` · `monitoring` · `restricted` · `metrics` — and `disable admin user` for a UI that can only look



On by default in the installer since `0.6.10`, on `https://localhost:8443`. On Linux: `sudo nscp web install`, then `sudo nscp web install-ui`.

Configuration that tells you when it’s wrong

Validation `0.25.0`

A misspelled section, a missing leading `/`, a key above the first section — all used to be ignored without a word.

```
$ nscp settings --validate
W  unused section: /settings/NRPE/sever
W  key outside any section: allowed hosts
exit 1
```

Also warned at every startup and reload, and shown in the web UI. (Output abridged.)

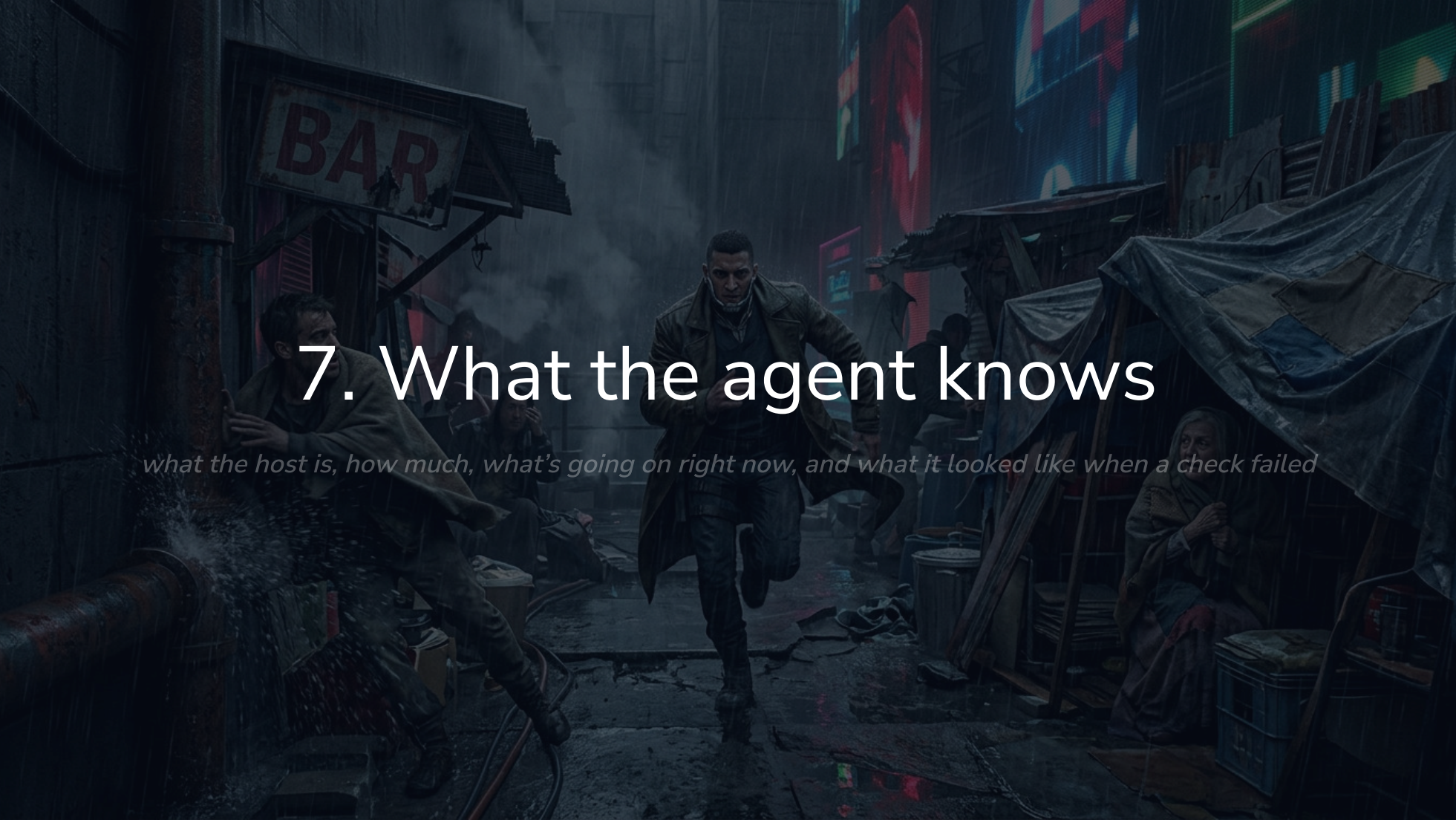
One `boot.ini`, a whole fleet `0.16.1`

```
[settings]
location = https://cfg.example.com/nsclient.ini?host=${host_lc}
```

`${hostname}` `${host}` `${domain}` (+ `_lc` / `_uc`) expand in settings urls, `[/includes]` and `[/attachments]` `0.17.0`. The query string is actually sent now. HTTP proxy support for the loader and every HTTP client `0.11.33`.

Installer

- `IMPORT_CONFIG=https://…/nsclient.ini` — copy a config once, at install `0.10.7`
- `CERTIFICATE` / `CERTIFICATE_KEY` / `CERTIFICATE_CA` — your own TLS material, no self-signed fallback `0.18.0`
- `ALLOW_CONFIGURATION=0` — touch nothing `0.11.24`
- `nscp nrpe install --verify=peer-cert …` — 2-way TLS NRPE in one command `0.10.7`



7. What the agent knows

what the host is, how much, what's going on right now, and what it looked like when a check failed

Host facts: what the host is 0.24.0

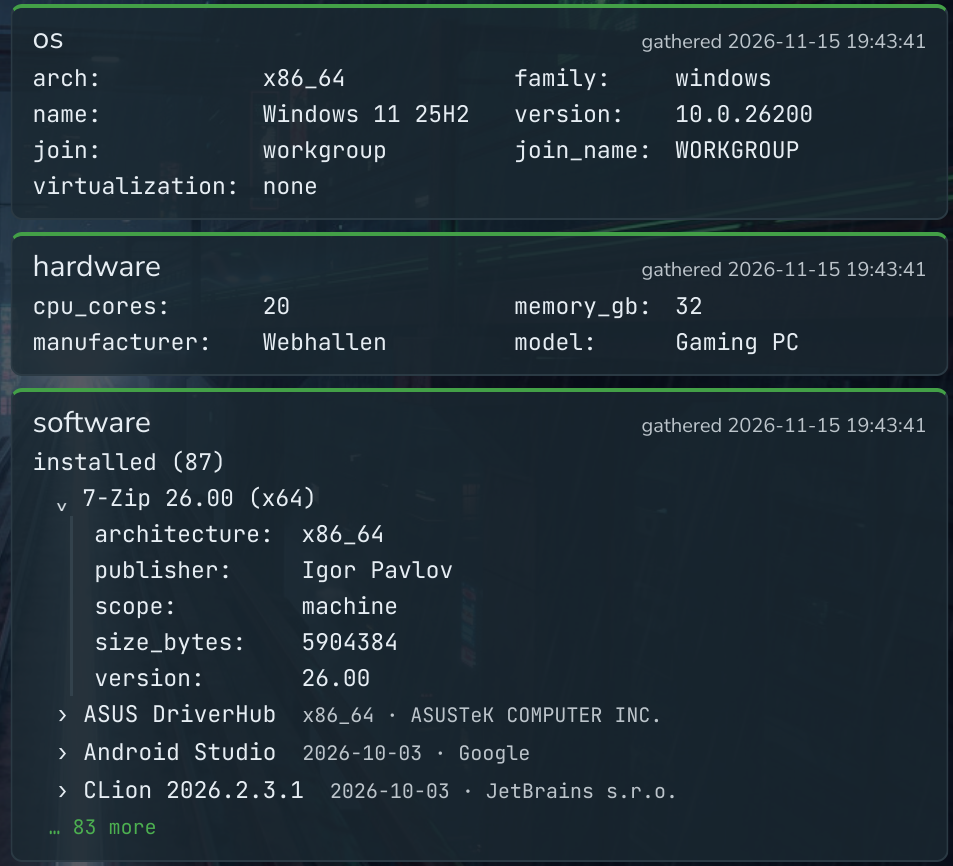
An opt-in inventory document per host, from the modules that already check those things:

`os` · `hardware` · `network.interfaces` · `storage.volumes` · `software.installed` · `services.installed` · `tasks.scheduled` · `agent` · `docker` · `mysql` · `mssql` · `hyperv.vms`

An inventory, not a dump

- every set is off until you enable it
- describes the machine, not its configuration: no credentials, no command lines
- volumes omit free space, services omit runtime state: those are checks

Read it over REST, in `nscp test`, with `check_nsclient`, or on the web UI's Facts page: that's mine on the right.



Metrics: how much

One snapshot, many readers

`0.22.0`

Every collecting module publishes into one snapshot, refreshed every 10 seconds: CPU, memory, disks, network, uptime. About 500 values on a Windows host.

- `GET /api/v2/metrics`, and `?meta=1` for type, unit and help
- `/api/v2/openmetrics`: typed and labelled, for Prometheus
- Graphite, collectd and Elastic submitters
- the Metrics page in the web UI
- a `metrics` role that can scrape and nothing else

Metrics — my desktop · 2026-11-16

Metric	Description	Value
disk › free › C: drive=C:		
free	Free space on the volume	640.8 GiB
free_pct	Free space as a share of the volume	34
total	Size of the volume	1.8 TiB
used	Used space on the volume	1.2 TiB
used_change_1h	Change in used space over the last hour	729.0 MiB
used_pct	Used space as a share of the volume	65
user_free	Free space the querying user may actual…	640.8 GiB
disk › io › _Total disk=_Total		
avg_queue_length	Average number of requests outstandin…	0.04
percent_disk_time	Share of the interval the disk was busy	0
percent_idle_time	Share of the interval the disk was idle	100
read_latency	Average time one read took over the int…	0.221 ms
total_latency	Average time one operation took over th…	0.238 ms
write_latency	Average time one write took over the int…	0.367 ms

… about 500 values in all · green rows are new in the next release

Diagnostics: what's going on right now

`main · next release`

`processes.top_memory` CheckSystem showing the first 10

The processes with the largest working set

name	pid	cpu %	working_set	private_bytes
clion64.exe	14908	0.4 %	2.6 GiB	3.4 GiB
idea64.exe	272732	0.2 %	1.6 GiB	2.6 GiB
Memory Compression	3868	0.0 %	1.5 GiB	23.0 MiB
rustrover64.exe	277764	0.3 %	1.5 GiB	3.5 GiB
claude.exe	22588	0.0 %	804.1 MiB	1.1 GiB
Rider.Backend.exe	212136	0.6 %	609.8 MiB	13.0 GiB

... 4 more rows

`disk.growth_spots` CheckDisk captured 48s ago

Size of the places that usually fill a disk

path	kind	size	note
C:\Users\micha\AppData\Local\Temp	temp	15.9 GiB	-
C:\Windows\Temp	temp	19.7 MiB	-
C:\Windows\Logs	logs	576.7 MiB	-
C:\pagefile.sys	page_file	44.3 GiB	-
C:\	shadow_storage	36.4 GiB	-

... 3 more rows

`services.problems` CheckSystem captured 1m ago · 2026-11-15 19:44:13

name	state	exit_code	binary_path	account
MapsBroker	stopped	50	C:\Windows\System32\svchost.exe	NT AUTHORITY\NetworkService

who is using the memory · where the disk went · which service isn't where it should be

Tables, not metrics. Also `processes.top_cpu` · `services.events` · `eventlog.errors` · `failures.recent`

- collected with the metrics round; a failure collects nothing
- bounded: 10 rows a table, 1 KB a cell, 16 tables
- off twice, behind its own grant · never a command line

Failed checks: what it looked like when it failed

The log `main · next release`

The core keeps the last 50 checks that didn't come back OK, from any transport, including the failures it makes itself: unknown command, permission denied.

- `GET /api/v2/failures`, filter by status, command, time
- a Failed checks page and a dashboard widget
- arguments: by default only the safe ones are kept (thresholds, filter, what was checked)

```
[/settings/core/failed checks]
enabled = true
max entries = 50
record arguments = safe    ; none · safe · all
capture metrics = true
capture diagnostics = true
```

What it keeps `main · next release`

- *System at the time*: the metrics snapshot the check failed under
- *Diagnostics at the time*: the tables from the last slide
- one snapshot is shared by every failure in the same interval
- ~70 KB of metrics each, under a 4 MiB budget for both kinds: enough for a full 50-entry log
- the oldest is released first; the entry stays, marked *evicted*
- `prune metrics` trims zeros, empties and duplicates
- both opt-in: `capture metrics`, `capture diagnostics`

By the time a human logs in, the spike is gone.
The snapshot isn't.

What the agent knows

Facts

what the host is

`0.24.0`

Metrics

how much

`0.22.0`

Diagnostics

what's going on right now

`main · next release`

Failed checks

what it looked like when it failed

`main · next release`



8. NSClient Fleet

the server that brings it all together

Fleet: a central server `Fleet 0.1`

- **One binary, or one container**: web UI, SQLite and TLS built in. `ghcr.io/mickem/nsclient-fleet`, on your own hardware.
- **Agents enroll themselves**: one-time token, CSR, client certificate back. After that the certificate *is* the identity.
- **Agent-initiated, mTLS, no inbound port.** The agent asks; the server never pushes. Most polls: *nothing changed*.
- **Nothing changes unless you enroll.** An agent that isn't enrolled is exactly the agent it was.

```
msiexec /qn /i NSCP-0.26.0-x64.msi ^
    FLEET_SERVER=https://fleet.example.internal:8443 ^
    FLEET_TOKEN=<one-time token> ^
    FLEET_CA=C:\certs\fleet-ca.pem
```

```
sudo nscp enroll --server https://fleet.example.internal:8443 \
                 --token <one-time token> \
                 --ca /etc/nsclient/security/fleet-ca.pem
```

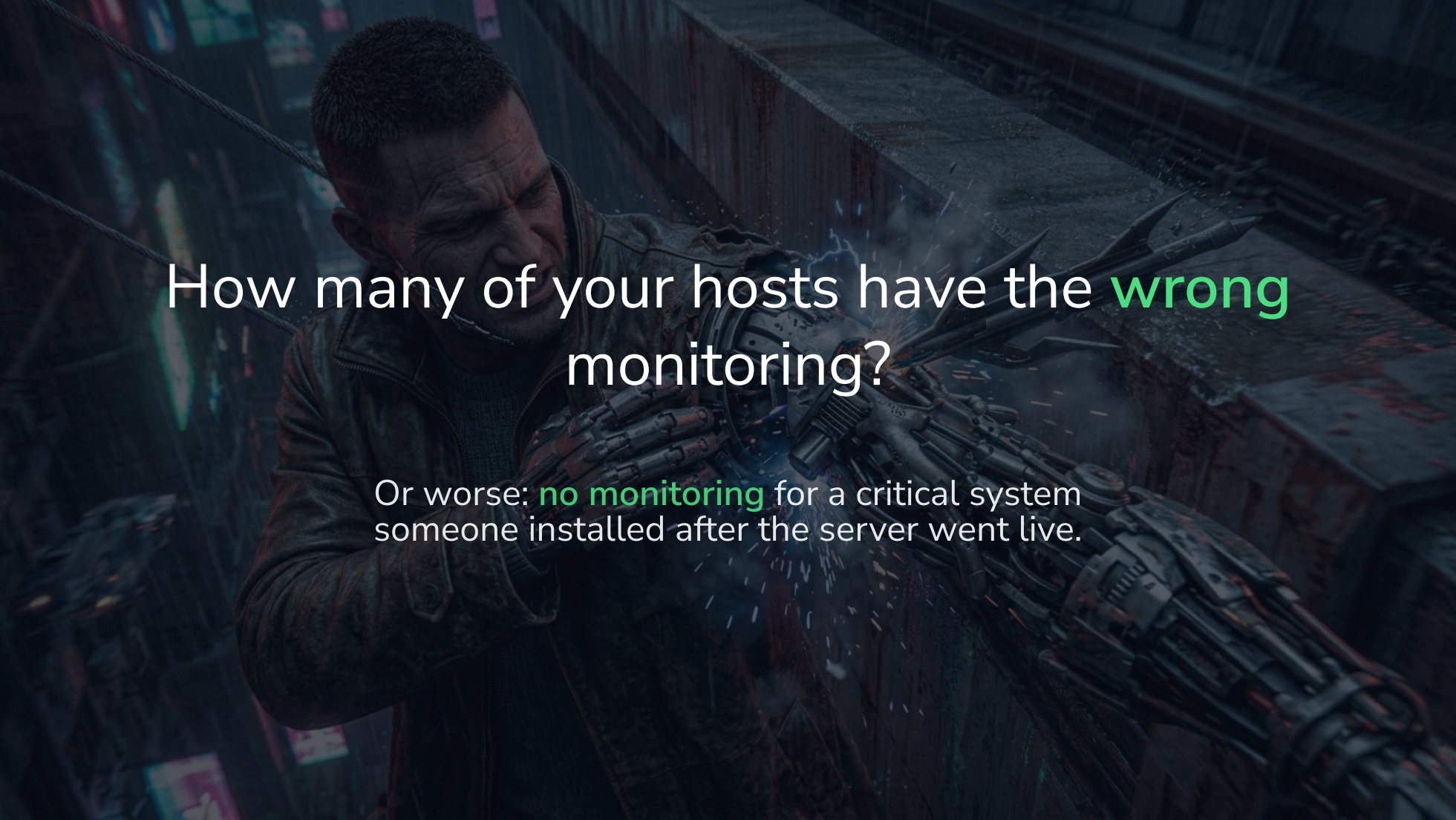
Agent side `0.21.0`+, use 0.22 or later. With `FLEET_SERVER` the MSI writes **no local config** `0.26.0`.



Central configuration

- **Bundles**: INI fragments and scripts, attached to **groups** with priorities. The higher priority wins a collision.
- **Scripts ship with the config**: batch, shell, Python, Lua `Fleet 0.1.3`, run in place with `script root = fleet` `0.26.0`
- **Applied atomically**: staged, then swapped. A half-written config never runs.
- **Encrypted bundles**: sealed in your browser; the server stores a fingerprint and can't open them.
- **Status is two hashes**: *in sync* · *out of sync* · *offline* · *lost*. Drift isn't detected; it's computed.
- **Local still wins, and says so**: a key in the host's own `nsclient.ini` overrides the fleet value, and the host reports that it did.
- **Config in git**: export groups and bundles as a directory of files, or serve them straight from one with `CONFIG_SOURCE_DIR` `Fleet 0.1.3`

One place to change it. One place to see whether it arrived.



How many of your hosts have the wrong monitoring?

Or worse: no monitoring for a critical system someone installed after the server went live.

Where the facts come from

Fleet has to know what a host is before it can decide what the host gets. Four sources, kept apart.

Host facts

`agent` `0.24.0`

the inventory document from section 7: `os`, `hardware`, `software.installed`, `services.installed`, …

a hash on every poll, uploaded only when it changed · Fleet keeps the history

Tags

`agent`

short `key = value` answers from the modules: `os`, `os_family`, `arch`, `virtualization`, `sqlserver = detected`

`service-tags`: a running service becomes a tag of your choosing

Scripts

planned

your own script reports facts and tags: what no module knows, like the application's version or the role of the box

ships in a bundle, like any other script

Import

`import:cmdb` `Fleet 0.1.3`

what no agent can know: owner, environment, criticality. A CMDB export or a spreadsheet, one row per host

CSV, JSON or YAML · previewed before anything is written

Nobody types it in, and nobody keeps it up to date.

From facts to config

Fleet 0.1

Groups are rules, not lists of machines. Read it left to right: what a host *is* decides what it *gets*.



Someone installs SQL Server. The fact changes, the host joins SQL Servers, and on its next poll it has the MSSQL checks. Uninstall it, and it leaves.

Nobody edited a config file, and nobody opened a ticket.

Bringing it all together

Config

ON ONE BOX

an `nsclient.ini` you edit by hand · `${host}` placeholders · `--validate`

ACROSS THE FLEET

bundles on groups, chosen by what the host is

Host facts

ON ONE BOX

`facts os` in `nscp test`

ACROSS THE FLEET

an inventory nobody types in, which also decides the groups

Failed checks

ON ONE BOX

the last 50 failures, with a metrics snapshot and diagnostics next release

ACROSS THE FLEET

a first guess at what went wrong

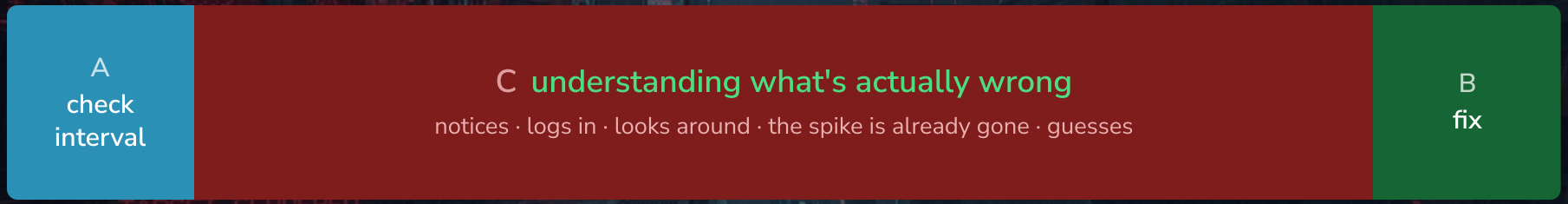
in review

Really automate your monitoring.

Not the checks. Everything around them: which checks, where, and what a failure means.

The one number that matters

time from error to recovery. What takes the longest?



minutes — hours — minutes

a typical incident as I've lived it, not a measurement

It isn’t measured in check intervals.
It’s measured in how long a human needs to get the context.

From `WARNING:` to a first guess

WHAT THE MONITORING SHOWS

Host	Service	Status	Status Information
faults-osl-14	check_cpu	CRITICAL	CRITICAL: 1m: 87%

WHAT FLEET SAYS ABOUT IT

```
$ check_nsclient fleet explain --host faults-osl-14 \
    --check check_cpu "CRITICAL: 1m: 87%"

faults-osl-14: The 1-minute CPU load [...] is almost entirely because
acme-indexer.exe is holding the host's only CPU core at 100% [...]
The evidence [...] does not show why the indexer is consuming the
core [...] Restarting it will clear [...]

Likely causes:
  - The acme-indexer.exe process is using 100% of the single CPU
    core, which accounts for the near-total CPU load [...]
  - The host has only one logical processor [...] so a single
    process at full use saturates the whole machine [...]

Confidence: high. Written by anthropic/claude-haiku-5-5; [...]
```

- the check result you were paged with
- the host facts: one core, a Hyper-V VM
- the snapshot and diagnostics: they name the process
- → Fleet asks a model: Claude, anything OpenAI-compatible, or a local Ollama

- off by default: nothing leaves the server until you configure a provider
- the evidence is untrusted input: credentials masked, no tools, schema-checked answer
- one description per problem, not per alert · a daily call budget per tenant

And if you don't like AI,
you can feed the same context to a human.

Your monitoring stays the same. This is just pretty-printing at the end.

What a first guess costs

One explanation: about 11,000 tokens of evidence in, about 800 tokens of answer out.

Claude Opus 5.5

6 cents

$59 per 1,000

Claude Sonnet 5.5

3 cents

$31 per 1,000

Claude Haiku 5.5

0.15 cents

$1.50 per 1,000

A local model

free

on your own hardware

Ollama, or anything OpenAI-compatible you host · the evidence never leaves your network

One explanation per problem, not per alert, under a daily budget: the bill has a ceiling.



And if you get a local AI model...

...you can always game on it when there are no incidents.

Grand Theft Auto VI cover art © Rockstar Games

Experimental today, and what's next

NSClient++

TODAY

- 84 commands are still marked experimental; their options may change
- Mod-Gearman: verified end to end, not yet run across a thousand hosts
- macOS: not signed or notarized yet

NEXT

- modern layout becomes the default on Windows
- experimental → stable for those 84 commands
- macOS: signed and notarized
- failed checks, snapshot and diagnostics in a release: on main today

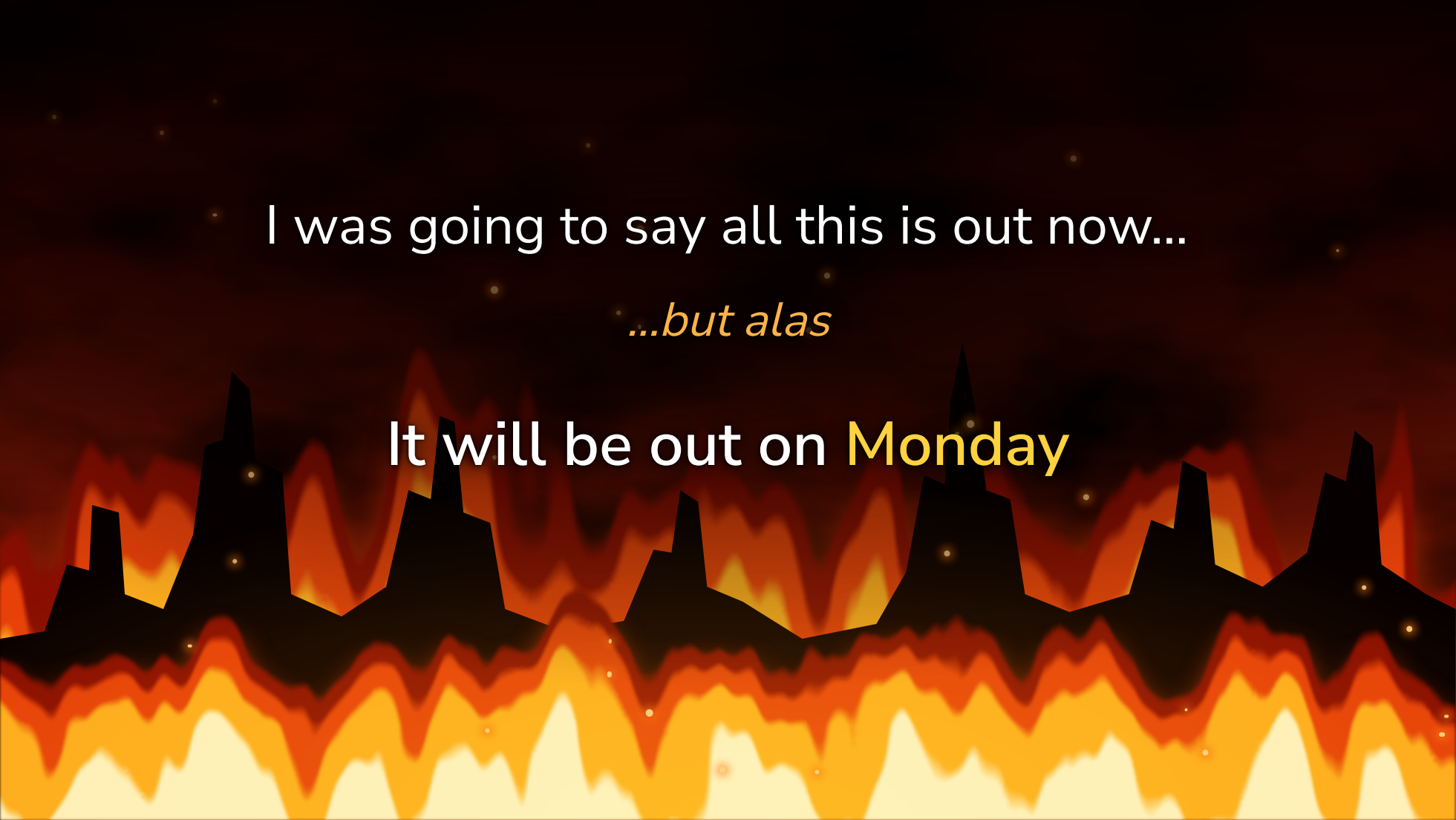
Fleet

TODAY

- the UI is a bit bare bones and will have bugs
- fleet config, bundle key included, is clear text on each host — use the modern layout
- without encrypted bundles the server can run code as SYSTEM or root on enrolled hosts, *by design*
- a TLS-inspecting proxy breaks the agent (pinned cert)

NEXT

- agent upgrades through Fleet, once the modern layout is the default; there are none yet
- adapters that write Icinga / Naemon / Nagios config from what the agent reports


I was going to say all this is out now...
...but alas
It will be out on Monday

Thank you

nsclient.org/download — the agent, `check_nsclient` and NSClient Fleet

nsclient.org/docs — the documentation: the agent, Fleet, `check_nsclient`

nsclient.org/docs/setup/upgrading — per-version actions, coming from 0.5.2

github.com/mickem/nscp — source, releases, discussions



linkedin.com/in/mickem



nsclient.org