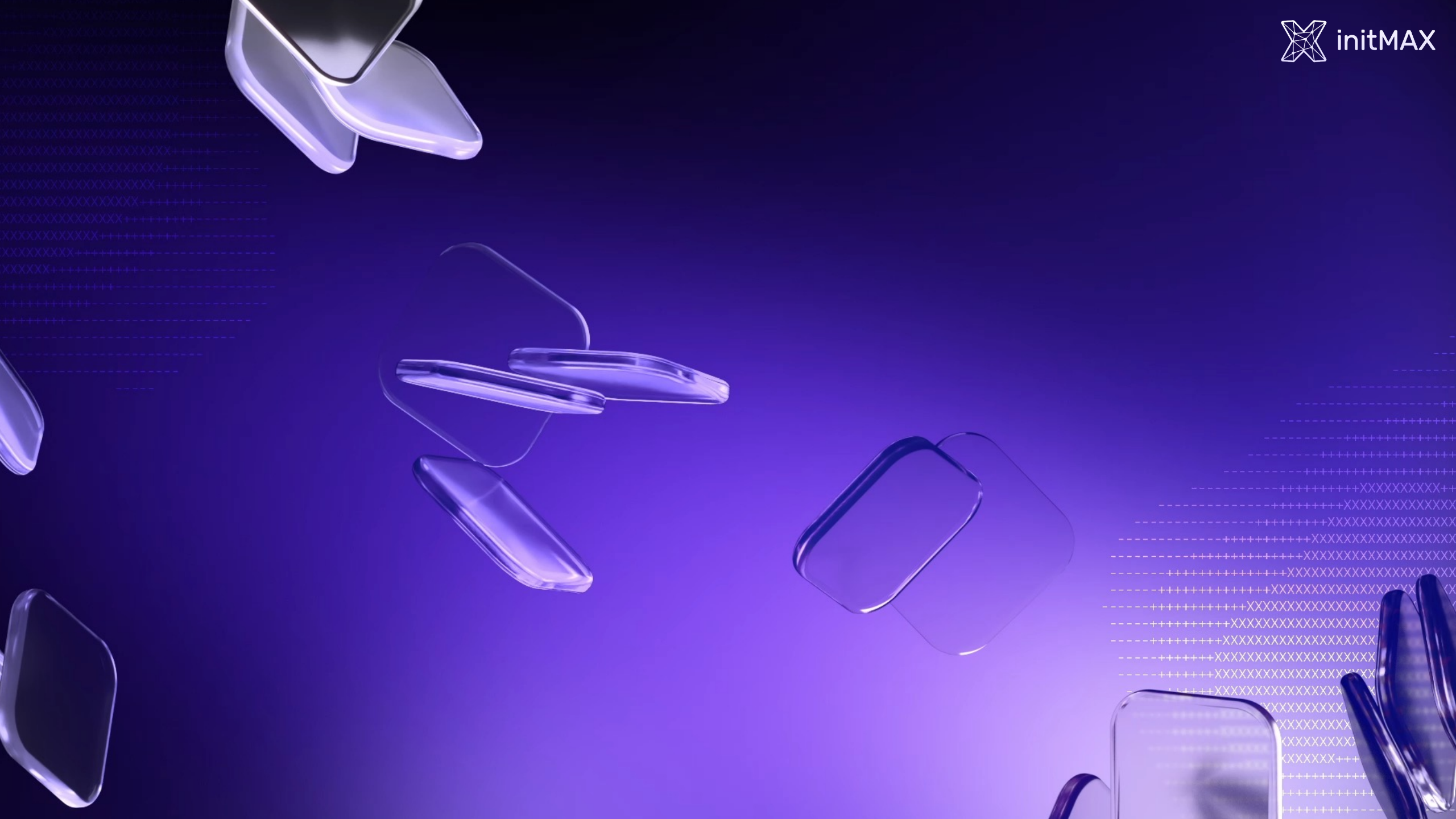

initMAX

initMAX

Securing Global Enterprise Environments with Zabbix

Tomáš Heřmánek

CEO & Founder | initMAX s.r.o.



initMAX

MAXimum trust, from engineers to enterprise

- Consulting
- Certified training courses
- Widgets & Modules
- Technical support
- All-in-one solutions
- Integration



initMAX

ZABBIX
PREMIUM DELIVERY PARTNER

ZABBIX
TRAINING PARTNER

PROXMOX

CYBERTEC
POSTGRESQL SERVICES & SUPPORT
OFFICIAL PARTNER

SMSEagle

wazuh.



initMAX

Where to find us
three offices, one team



USA

Czechia

Slovakia

initMAX



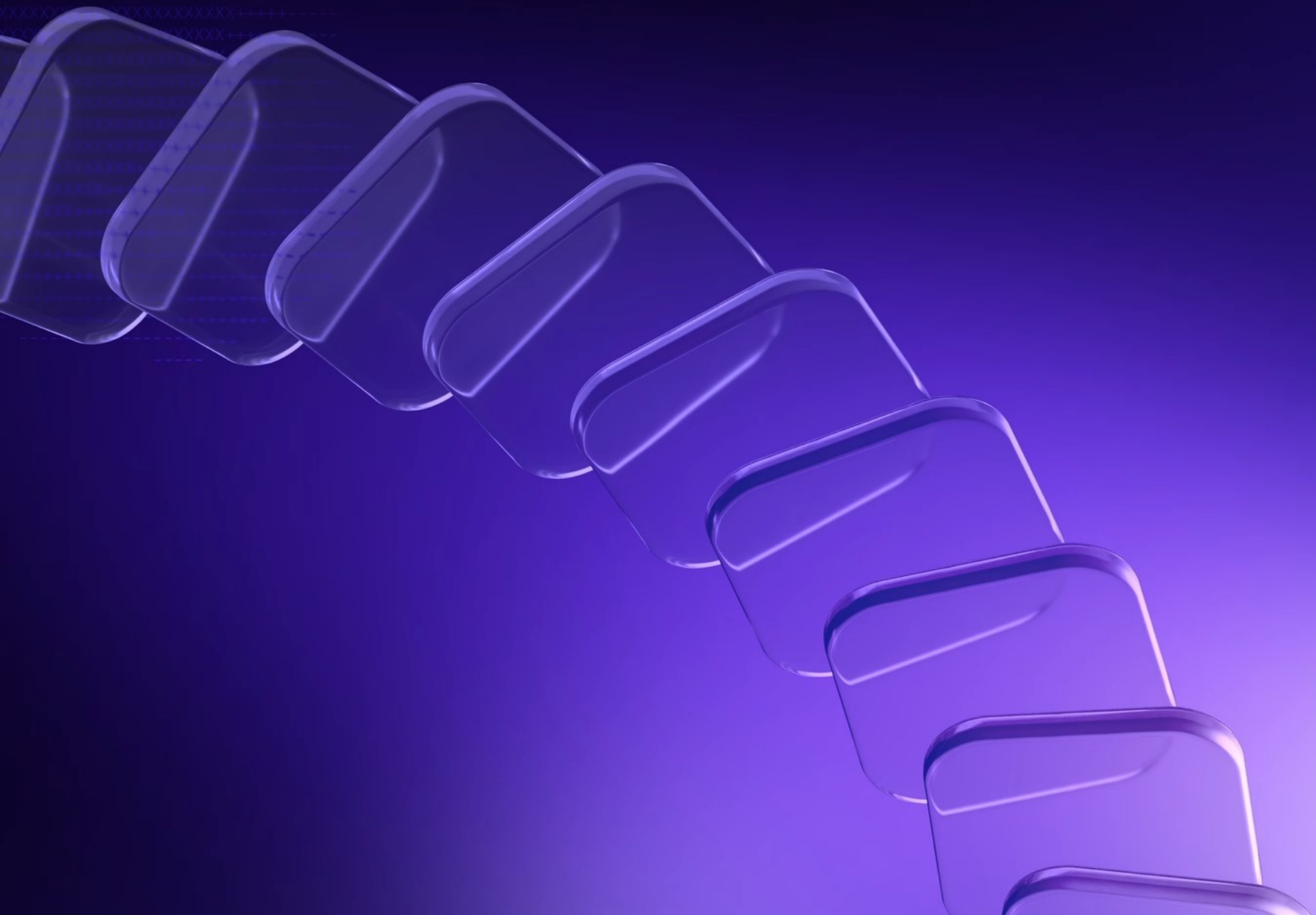
initMAX

NIS2: monitoring is now part of compliance

- **Monitoring = critical asset**
  Zabbix sees every host, credential and metric – breach the monitoring, breach everything.
- **NIS2 (EU 2022/2555)**
  Essential & important entities: risk management, incident handling, supply-chain security.
- **Article 21 requirements**
  Encryption, access control, MFA, secure authentication, logging and **continuous monitoring.**
- **Incident reporting**
  Early warning within 24 h, full notification within 72 h – Zabbix delivers the evidence.
- **AI assistants: no sensitive data, ever**
  A pasted secret or config cannot be recalled – chat history is a new entry point (NIS2 Art. 21, GDPR).
- **This talk**
  10 practical hardening steps for a Zabbix enterprise environment.



initMAX



initMAX

Encrypt every component – Zabbix 7.4+

- **Server – Proxy – Agent**
  TLS with certificates or PSK; TLSConnect / TLSAccept on every side, no unencrypted fallback.
- **Server / Proxy – Database**
  DBTLSConnect=verify_full with CA, client cert & key – PostgreSQL and MySQL/MariaDB.
- **Frontend – Server & DB**
  TLS on the native frontend–server connection (port 10051, new in 7.4); HTTPS only on the web tier and the JSON-RPC API (HSTS, TLS 1.2+ ciphers).
- **Web service (reports) & Vault**
  TLS for zabbix_web_service and HashiCorp / CyberArk connections.
- **Exception: Java gateway**
  JMX – Java gateway – server traffic is not TLS-encrypted – isolate it in a dedicated segment.
- **Data at rest**
  PostgreSQL has no built-in TDE – encrypt the data directory with LUKS, or use Percona pg_tde / CYBERTEC TDE. Encrypt and test backups – dumps contain macros and secrets.



initMAX

Encryption – our recommendations

- **Server – Proxy – Agent**
  Certificates or PSK on every hop – **never** reuse the same PSK key across hosts.
- **Database – all directions**
  Server, frontend and the DB-monitoring agent all talk to the DB – TLS on every one of them.
- **Frontend – Server (new in 7.4)**
  Frontend–server TLS was impossible before 7.4 – the last unencrypted gap is closed.
- **Web service & Vault**
  Use HashiCorp Vault; 8.0 adds AppRole auth. Watch out: token renewal must use POST, not GET.
- **Java gateway**
  Run it next to the server/proxy so JMX traffic stays local and isolated; gateway–application link can be TLS.
- **Data at rest**
  Encrypt the DB files on disk – open-source options exist, or CYBERTEC PostgreSQL TDE (our partner product).



initMAX

Server to DB TLS and encrypted backups

```
# /etc/zabbix/zabbix_server.conf
```

```
DBHost=db.example.com
DBPort=5432
DBTLSConnect=verify_full
DBTLSCAFile=/etc/zabbix/tls/ca.crt
DBTLSCertFile=/etc/zabbix/tls/server.crt
DBTLSKeyFile=/etc/zabbix/tls/server.key
# DBHost must match the server certificate

# Separate Bash script: encrypted DB backup
set -o pipefail
pg_dump -Fc zabbix | age -r "$AGE_PUBKEY" > zbx.dump.age
# AGE_PUBKEY = your age PUBLIC key; set it first
# pg_dump exports DB; age encrypts the backup file
# Configure TLS for pg_dump separately
```

initMAX

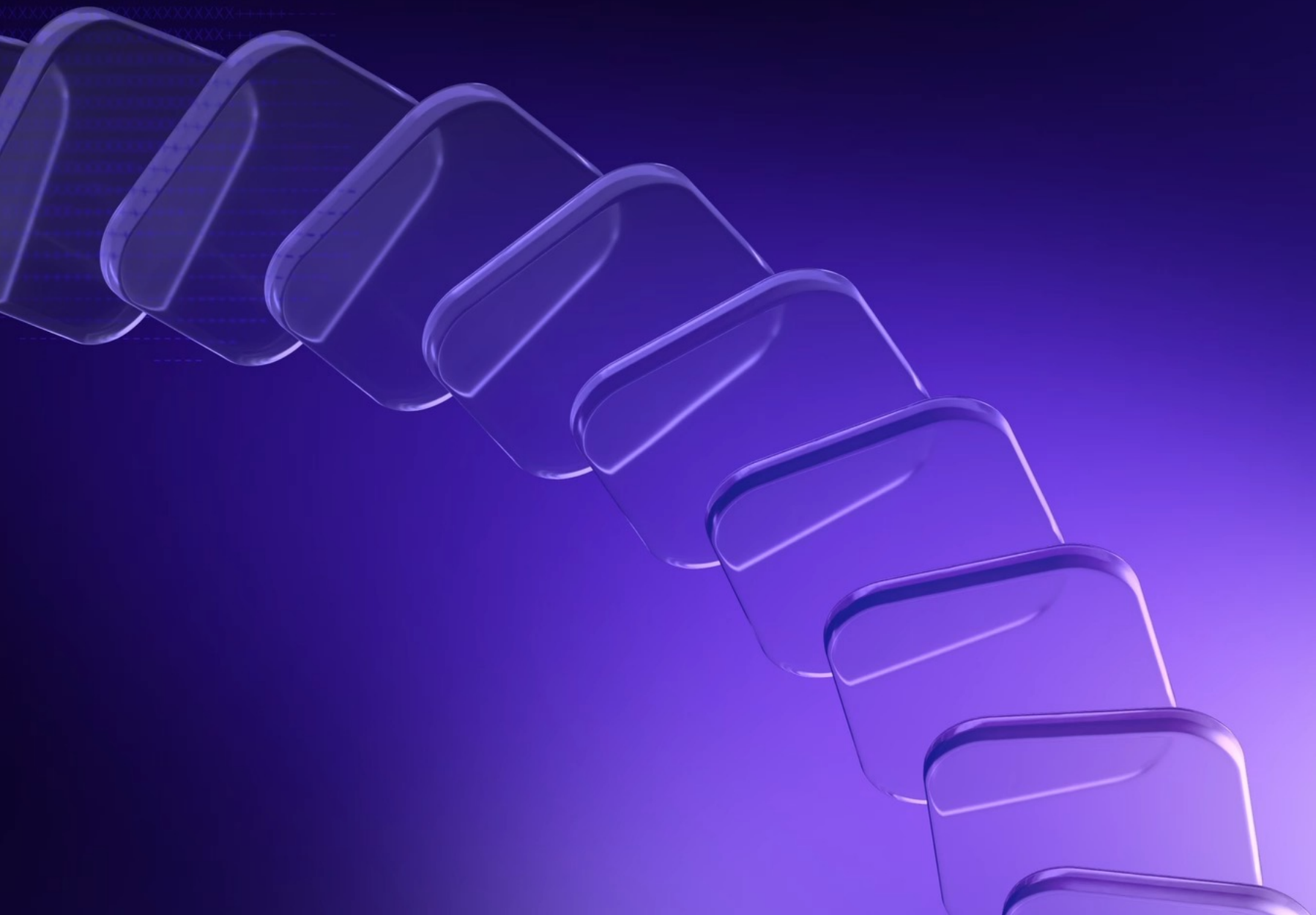
Frontend to DB: verified TLS connection

```
# /etc/zabbix/web/zabbix.conf.php
```

```
<?php
$DB['TYPE'] = 'POSTGRESQL';
$DB['SERVER'] = 'db.example.com';
$DB['PORT'] = '5432';
$DB['SCHEMA'] = '';

$DB['ENCRYPTION'] = true;
$DB['VERIFY_HOST'] = true;
$DB['CA_FILE'] = '/etc/zabbix/tls/ca.crt';
$DB['CERT_FILE'] = '/etc/zabbix/tls/frontend.crt';
$DB['KEY_FILE'] = '/etc/zabbix/tls/frontend.key';
$DB['CIPHER_LIST'] = '';
```

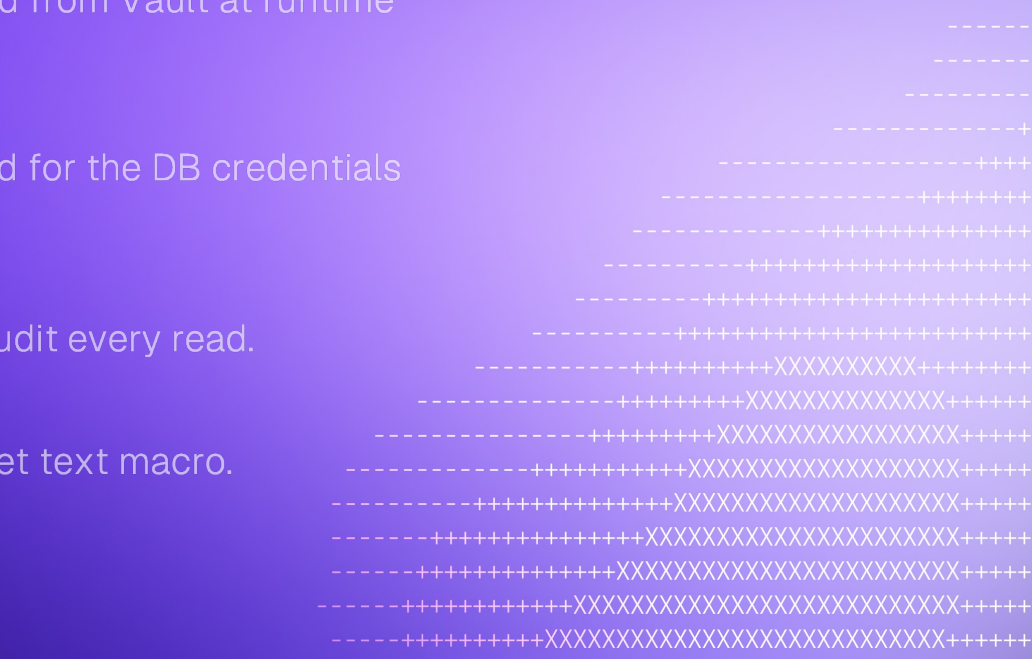
initMAX



initMAX

HashiCorp Vault vs. "secret text" macros

- **Secret text macro**
  Hidden in the frontend and API only – the value is still stored in plaintext in the globalmacro / hostmacro tables.
- **Who can read it**
  Anyone with DB access, a DB dump, a backup file or a compromised frontend server.
- **Vault macro**
  Zabbix stores only the path (secret/zabbix/db:password); the value is fetched from Vault at runtime and kept in server memory.
- **Supported backends**
  HashiCorp Vault (KV v2, AppRole / token) and CyberArk – for user macros and for the DB credentials themselves.
- **Operational rules**
  Short-lived tokens, dedicated policy per Zabbix instance, VaultTLSCertFile, audit every read.
- **Rule of thumb**
  If the credential unlocks something valuable, it belongs in Vault, not in a secret text macro.



initMAX

Secrets & Vault – our recommendations

- **Token renewal: POST, not GET**
  The most common failure – renewal silently fails, the token expires and Zabbix stops resolving macros. Plan it, monitor it.
- **Move to 8.0 AppRole as soon as it ships**
  Native AppRole login replaces hand-managed tokens – switch the moment 8.0 is available.
- **Until then: renew from inside Zabbix**
  Our workaround: a Script item calls the Vault API and renews the token – the same token that Zabbix uses for the DB credentials.
- **Vault is a hub, not an island**
  Auto-unseal and HSM keys from Azure Key Vault; Azure / cloud secrets engines issue short-lived credentials downstream.
- **Proxies can have their own Vault**
  Each proxy site keeps its secrets local – no central Vault exposed to every remote network.
- **Encrypt every backup**
  DB dumps and config backups may carry secrets – encrypt them whether you think they do or not.



initMAX

Vault integration – server config & macro

```
# /etc/zabbix/zabbix_server.conf   (secret keys: username, password)
```

```
Vault=HashiCorp
VaultURL=https://vault.example.com:8200
VaultDBPath=secret/zabbix/database
SSLCertLocation=/etc/zabbix/tls
SSLKeyLocation=/etc/zabbix/tls
VaultTLSCertFile=vault-client.crt
VaultTLSKeyFile=vault-client.key

# Token via systemd drop-in (NOT together with VaultToken= in .conf)
# /etc/systemd/system/zabbix-server.service.d/vault.conf
[Service]
Environment=VAULT_TOKEN=<AppRole token>

# Host macro {$DB.PASSWORD}, type Vault secret → next slide
```

initMAX

Vault integration – frontend DB configuration

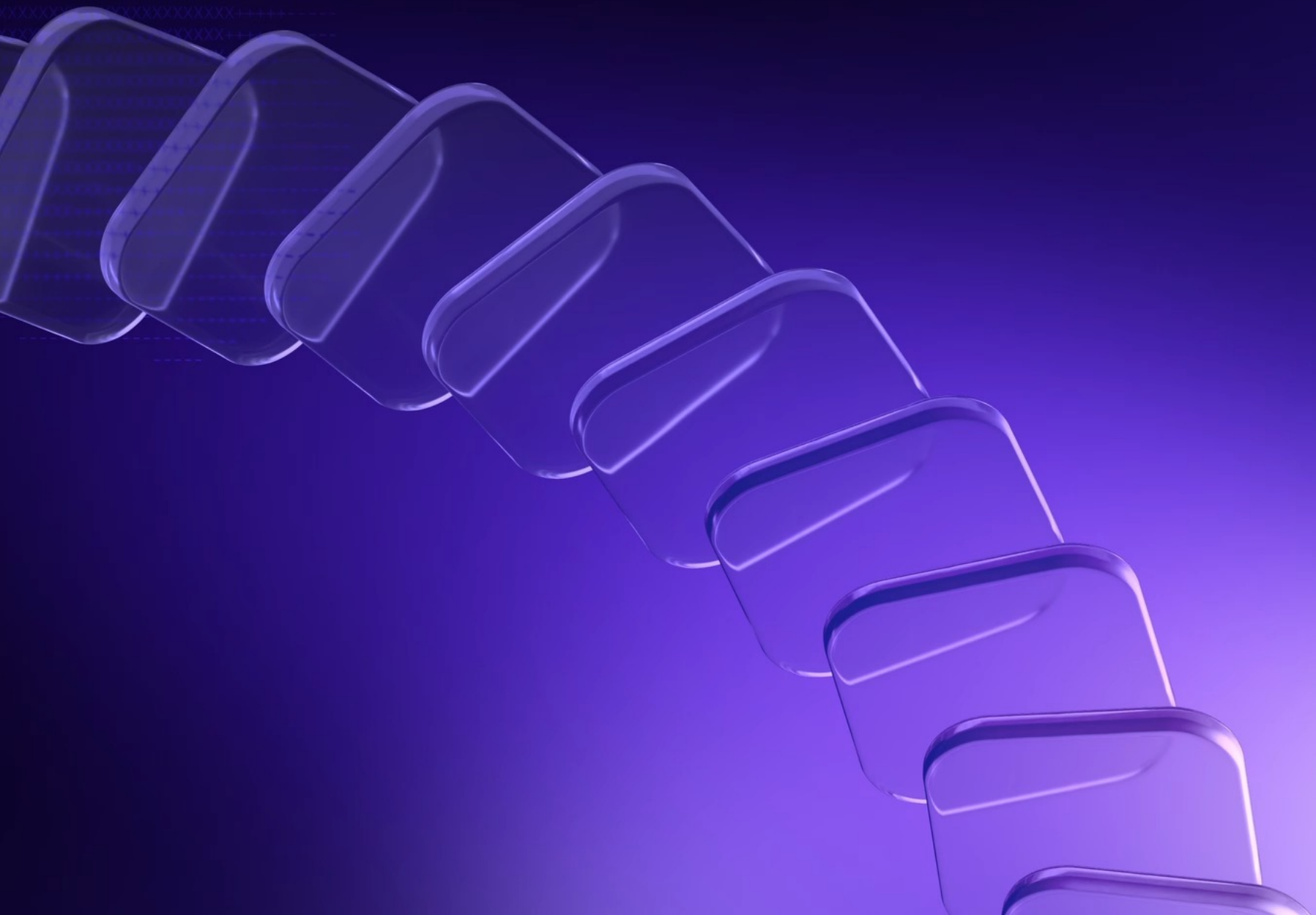
```
# /etc/zabbix/web/zabbix.conf.php   (secret keys: username, password)
```

```
$DB['VAULT']          = 'HashiCorp';
$DB['VAULT_URL']      = 'https://vault.example.com:8200';
$DB['VAULT_DB_PATH']  = 'secret/zabbix/frontend';
$DB['VAULT_TOKEN']    = '<frontend token>';   // no env var
$DB['USER']           = '';   // keep empty
$DB['PASSWORD']       = '';   // keep empty

// Vault secrets: secret/zabbix/database (server), /frontend (web)
// DB rights: SELECT INSERT UPDATE DELETE; upgrade needs owner

// Host macro {$DB.PASSWORD} - type Vault secret
// value: secret/zabbix/hosts/db01:password
```

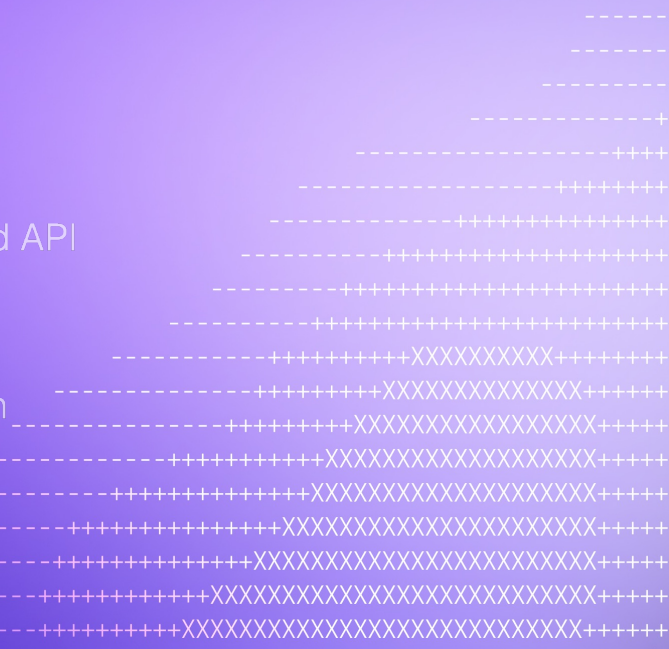
initMAX



initMAX

Proxies and DMZ – the server never talks to the edge

- **One proxy per zone**
  DMZ, OT / production, branch offices, cloud VPCs – the server only accepts connections from proxies.
- **Active proxies from inside out**
  Proxy initiates the connection to the server; nothing is initiated into the proxy zone – the server allows only 10051 from known proxy IPs.
- **Passive proxies in the DMZ**
  Server polls the proxy; agents in the DMZ never see the core network.
- **Firewall matrix**
  10051 server ← proxy only, 10050 agent ← its proxy only, 443 frontend ← reverse proxy / WAF.
- **Frontend isolation**
  Web tier behind a reverse proxy with WAF, IP allow-lists and rate limiting; API access via dedicated API users.
- **Bonus**
  Proxy keeps unsent data through outages (ProxyOfflineBuffer, ProxyBufferMode=hybrid) – RAM in normal operation, DB only as a safety net.



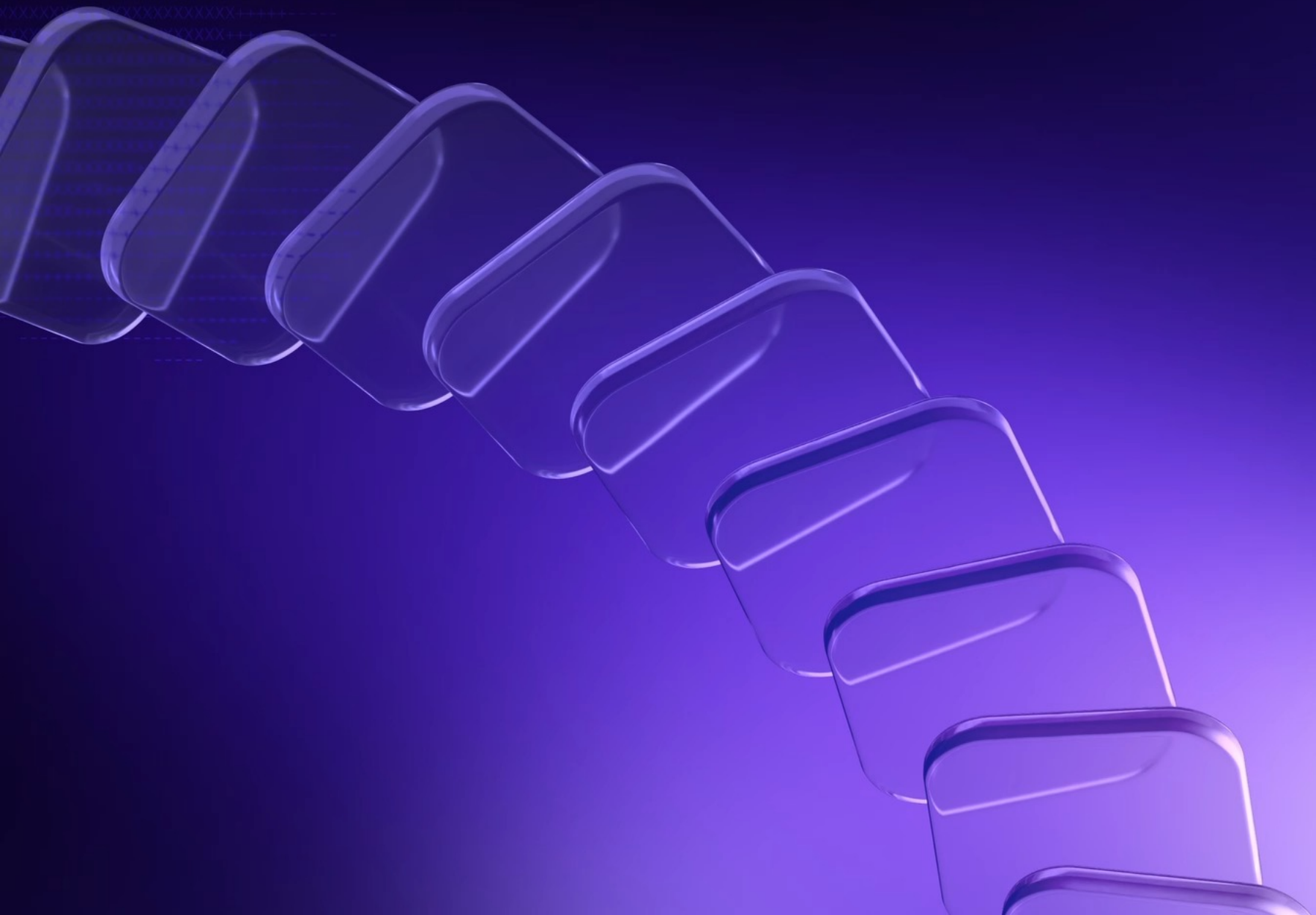
initMAX

Network segmentation – our recommendations

- **One proxy per zone – HA counts as one**
  HA group in the same zone with the same firewall rules behaves as one proxy and works fine.
- **Active proxies where you can go out, but not in**
  The proxy opens the connection from its zone to the server; one inbound rule on the server (10051 from proxy IPs), none into the proxy zone.
- **Passive proxies where nothing leaves the DMZ**
  Agents inside only see the proxy; the server reaches it over one explicitly defined rule.
- **WAF yes – but keep it in step with Zabbix**
  Inspection may block a new protocol version (XML to JSON) – update rules, test before upgrade.
- **Exposed frontend: lock it down and watch the API**
  API is open to every user today; 8.0 flips it: per role, off for new users, empty allow-list = none.
- **Buffer unsent data only – hybrid mode, encrypted DB**
  ProxyOfflineBuffer sized for the longest expected outage, ProxyLocalBuffer=0 (no need to keep sent data), ProxyBufferMode=hybrid – flushes to disk on shutdown; pure memory loses data on restart.
- **No SQLite on proxies in secured environments**
  SQLite file is unencrypted – use local MySQL / PostgreSQL with encryption, encrypt backups.



initMAX



initMAX

SAML / LDAP / MFA and hardcore user provisioning

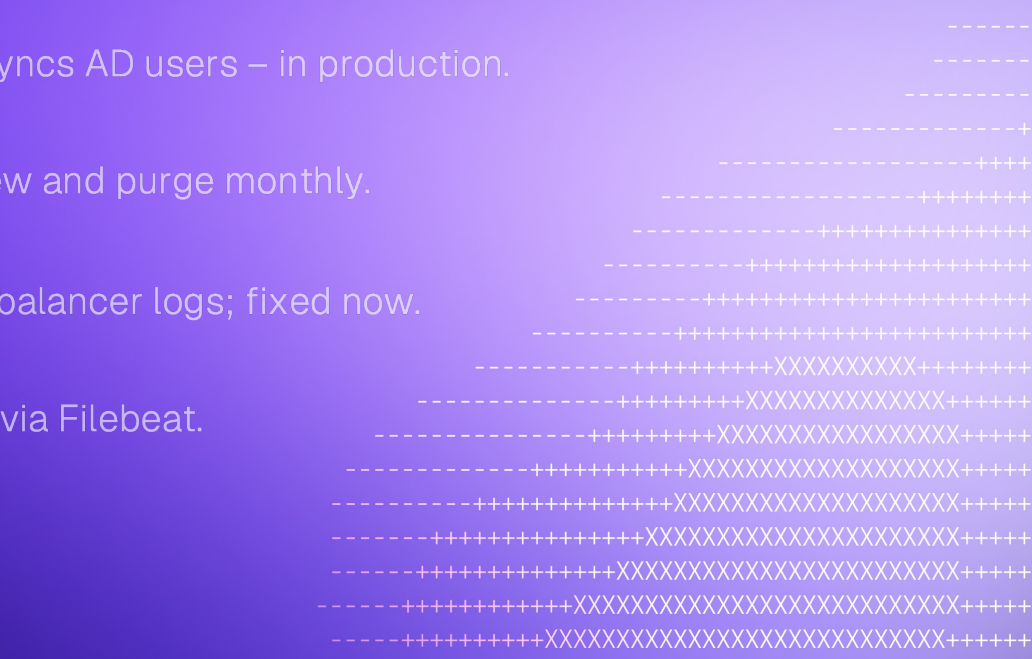
- **No local passwords**
  SAML (Entra ID, Keycloak, ADFS) or LDAP against Active Directory; keep one break-glass Super admin.
- **MFA everywhere**
  Built-in TOTP / Duo since 7.0 – enforce per user group, not just for admins.
- **JIT provisioning**
  SAML / LDAP user provisioning maps IdP groups and attributes to Zabbix roles, user groups and media.
- **Windows AD sync – hardcore mode**
  WMI / LDAP items pull group membership, JavaScript preprocessing builds the user object, API script creates and disables accounts.
- **Deprovisioning**
  Leavers are disabled the moment they drop out of the AD group – no orphaned admin account.



initMAX

Authentication – our recommendations

- **SAML or LDAP – always with MFA**
  Whichever you use, pair it with a second factor; LDAP only over LDAPS.
- **Watch the certificates**
  LDAPS and SAML signing certs expire – monitor them in Zabbix, no surprise lock-out.
- **8.0: SAML certs uploaded in the frontend**
  Stored in the database, so every HA frontend node has them – one more reason to encrypt the DB.
- **Provisioning: SCIM exists, JavaScript works too**
  Most customers skip SCIM; where IdP sync is not an option, WMI item + JS syncs AD users – in production.
- **Deprovisioning disables, never deletes**
  Leavers may own dashboards or sit in notifications – disable them, then review and purge monthly.
- **Audit log: know its gaps**
  Older versions skipped some updates (e.g. tags) – we had to audit from load-balancer logs; fixed now.
- **Audit log lives only in the DB**
  Workaround: resolve IDs to user / IP / object, append to a file, ship to Wazuh via Filebeat.



initMAX

Microsoft Entra ID group sync – JavaScript + Zabbix API

Microsoft Graph → Zabbix API | sync logic excerpt

```
var adUsers  = JSON.parse(azureADGetMembers()).value;
var zbxUsers = JSON.parse(zabbixGetMembers(zabbixURL, groupId)).result;

for (var i = 0; i < adUsers.length; i++) {
    var user = adUsers[i];
    var exists = zbxUsers.some(function (m) {
        return m.username === user.userPrincipalName;
    });
    if (!exists) {
        userCreate(zabbixURL, user, groupId);
    }
}

var removed = zbxUsers.filter(function (m) {
    return !adUsers.some(function (u) {
        return u.userPrincipalName === m.username;
    });
});
disableUsers(zabbixURL, removed);
```

initMAX

Export enriched audit JSONL with PostgreSQL

- **1 · One paste in psql**
  Export all stored auditlog columns for the last 24 hours to a local JSONL file.
- **2 · Names and host context**
  Prefer recorded names; add current names for users, hosts, items and triggers, plus their host context.
- **3 · Preserve the complete audit record**
  Keep original IDs, HA CUIDs and full details, including old/new values. Protect sensitive output.
- **4 · Keep the scope explicit**
  One-off snapshot, not a live feed. Nested IDs stay unchanged; unavailable historical names cannot be recovered.



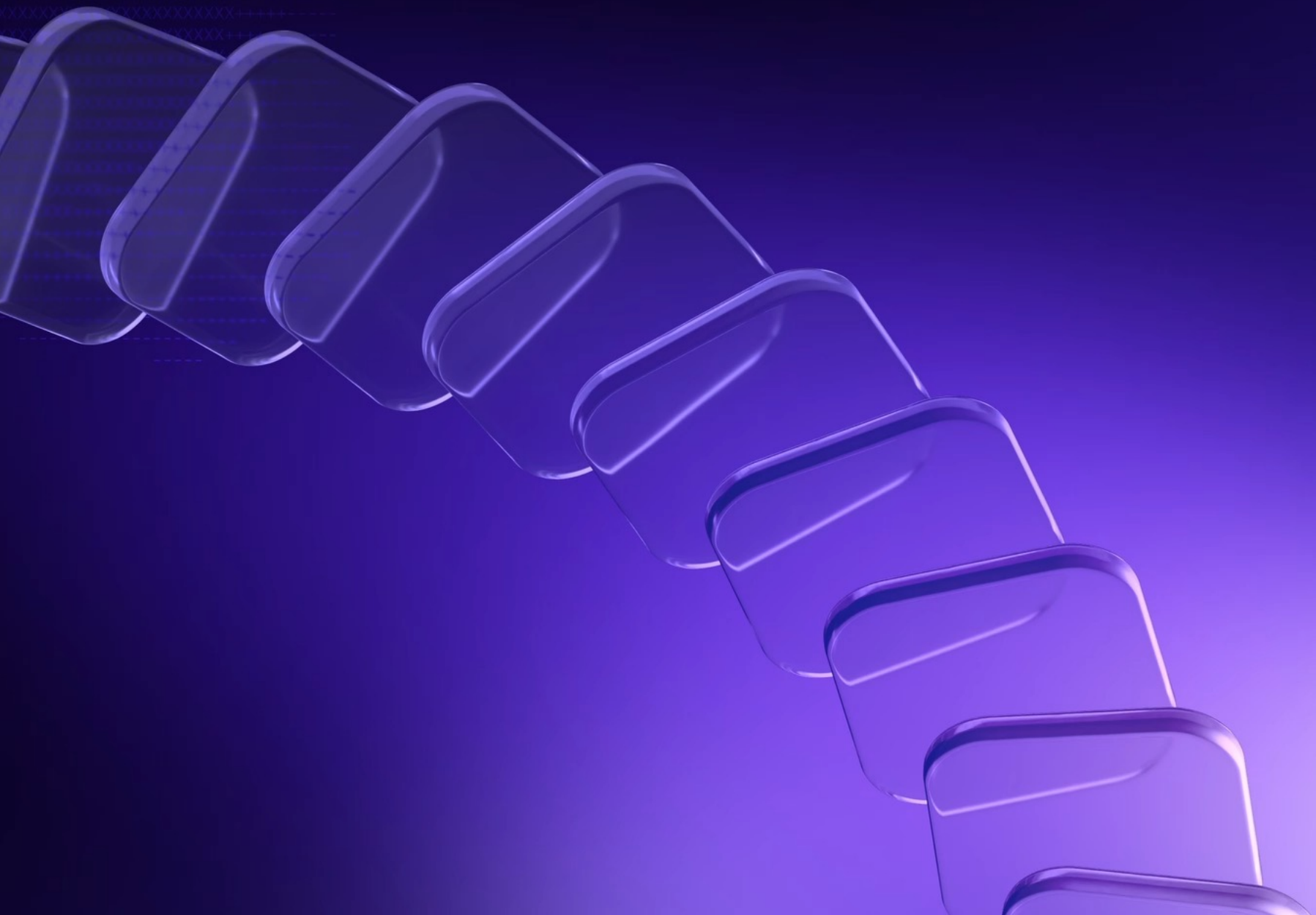
initMAX

One paste in psql → full audit JSONL

```
-- Connected to Zabbix PostgreSQL; verify DB TLS separately
```

```
\pset format unaligned
\pset tuples_only on
SELECT (to_jsonb(a)||jsonb_build_object('at',to_timestamp(a.clock),
 'action_name', coalesce((ARRAY['Add','Update','Delete',NULL,'Logout',
  NULL,NULL,'Execute','Login','Failed login','History clear',
  'Configuration refresh','Push'])[a.action+1], 'Action ' || a.action),
 'type_name','{"0":"User","4":"Host","13":"Trigger","15":"Item","47":"HA node"}'
  ::jsonb->>a.resourcetype::text,
 'name',coalesce(nullif(a.resourcename,''),h.name,i.name,t.description,u.username),
 'host_context_current',coalesce(th.n,h.name,ih.name)))::text FROM auditlog a
LEFT JOIN hosts h ON a.resourcetype IN (4,30,37) AND h.hostid=a.resourceid
LEFT JOIN items i ON a.resourcetype IN (15,36,52,56) AND i.itemid=a.resourceid
LEFT JOIN hosts ih ON ih.hostid=i.hostid
LEFT JOIN triggers t ON a.resourcetype IN (13,31) AND t.triggerid=a.resourceid
LEFT JOIN users u ON a.resourcetype=0 AND u.userid=a.resourceid
LEFT JOIN LATERAL (SELECT string_agg(DISTINCT ht.name,', ' ORDER BY ht.name) n
 FROM functions f JOIN items ti ON ti.itemid=f.itemid
 JOIN hosts ht ON ht.hostid=ti.hostid WHERE f.triggerid=t.triggerid) th ON true
WHERE a.clock>=extract(epoch FROM now())::bigint-86400 ORDER BY a.clock,a.auditid
\g '|umask 077; f=/tmp/zabbix-audit.jsonl; : >"$f" && chmod 600 "$f" && cat >"$f"'
```

initMAX

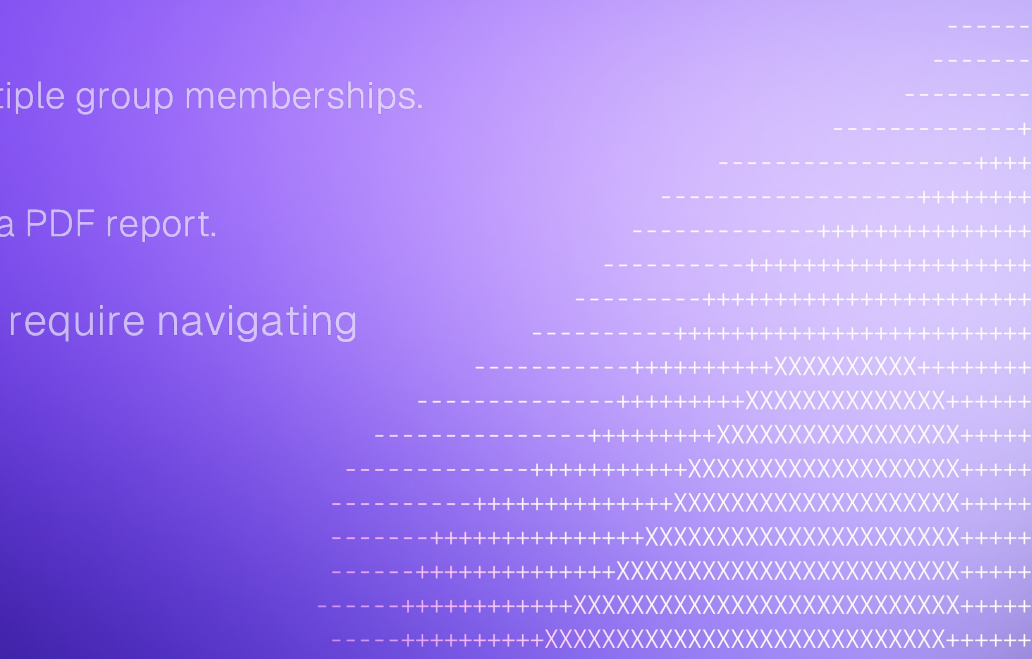


initMAX

permissionsMAX – Zabbix permissions in one place

- **See access at a glance**
  Instantly see what a user can access and what they can do.
- **Grant access to whole hierarchies**
  Quickly grant access to groups and subgroups without configuring each one individually.
- **Edit in the matrix**
  Change permissions directly in a clear matrix, without navigating separate forms.
- **Understand where access comes from**
  Understand where access comes from and why it is denied, even across multiple group memberships.
- **Complete user overview**
  Get a complete user overview – access, roles, notifications and activity, with a PDF report.

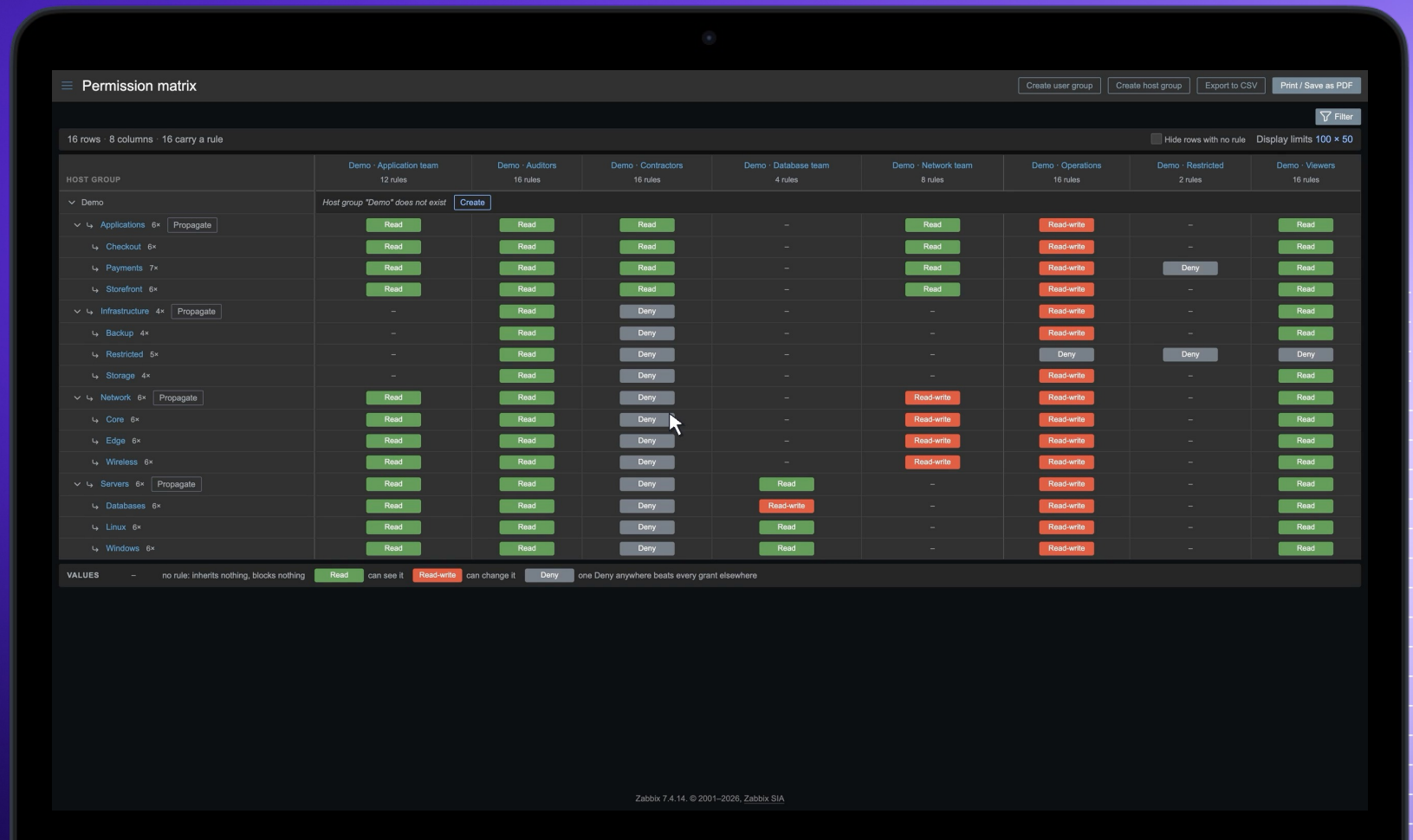
permissionsMAX brings together information and actions that normally require navigating several screens in Zabbix.



initMAX

permissionsMAX – Permission matrix

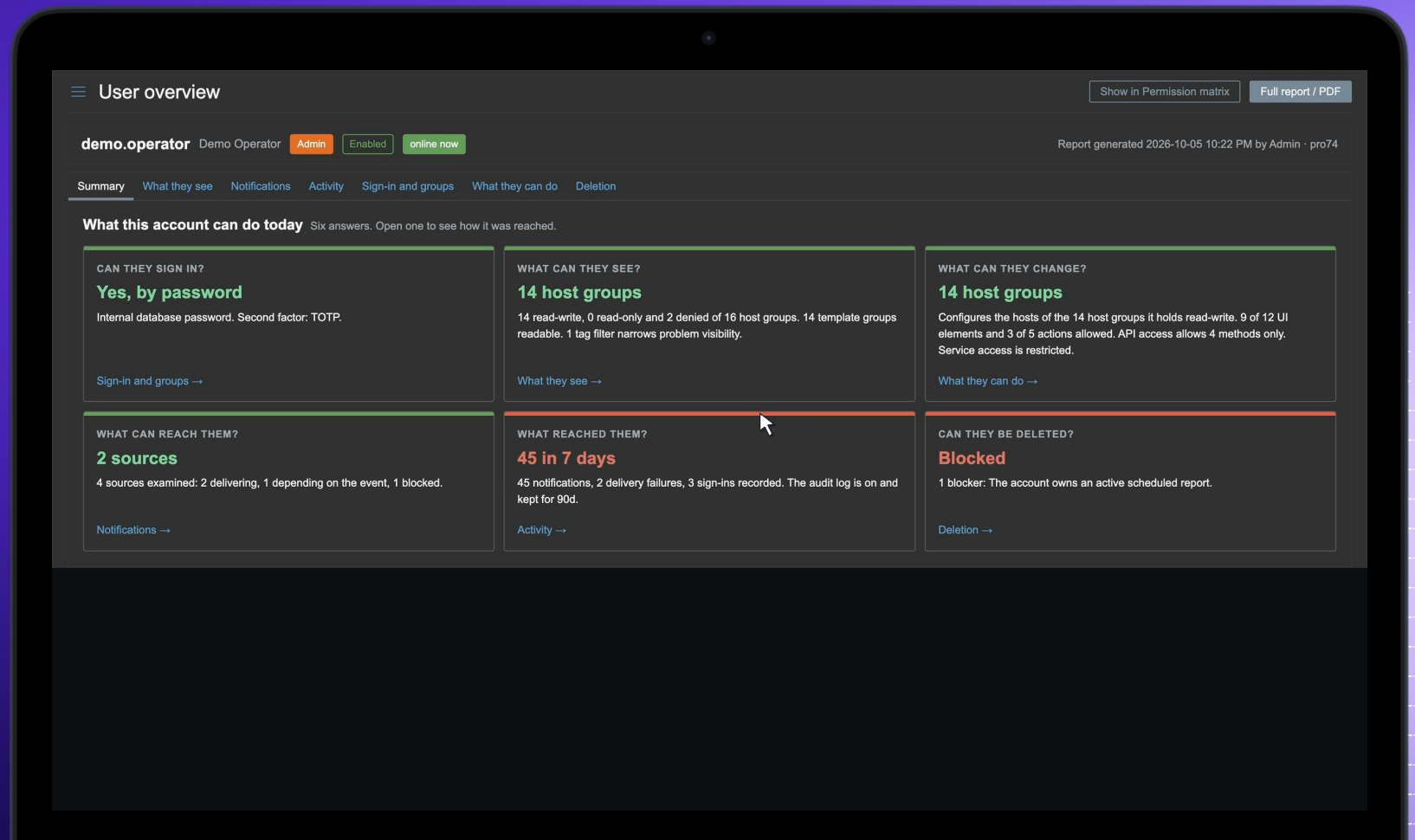
- **Matrix view**
  Access between user groups and host groups at a glance.
- **Click to change**
  Edit a permission directly in the matrix cell.
- **Group hierarchy**
  Groups and subgroups shown as a clear tree.
- **Propagate rules**
  Push rules down to subgroups, with confirmation first.



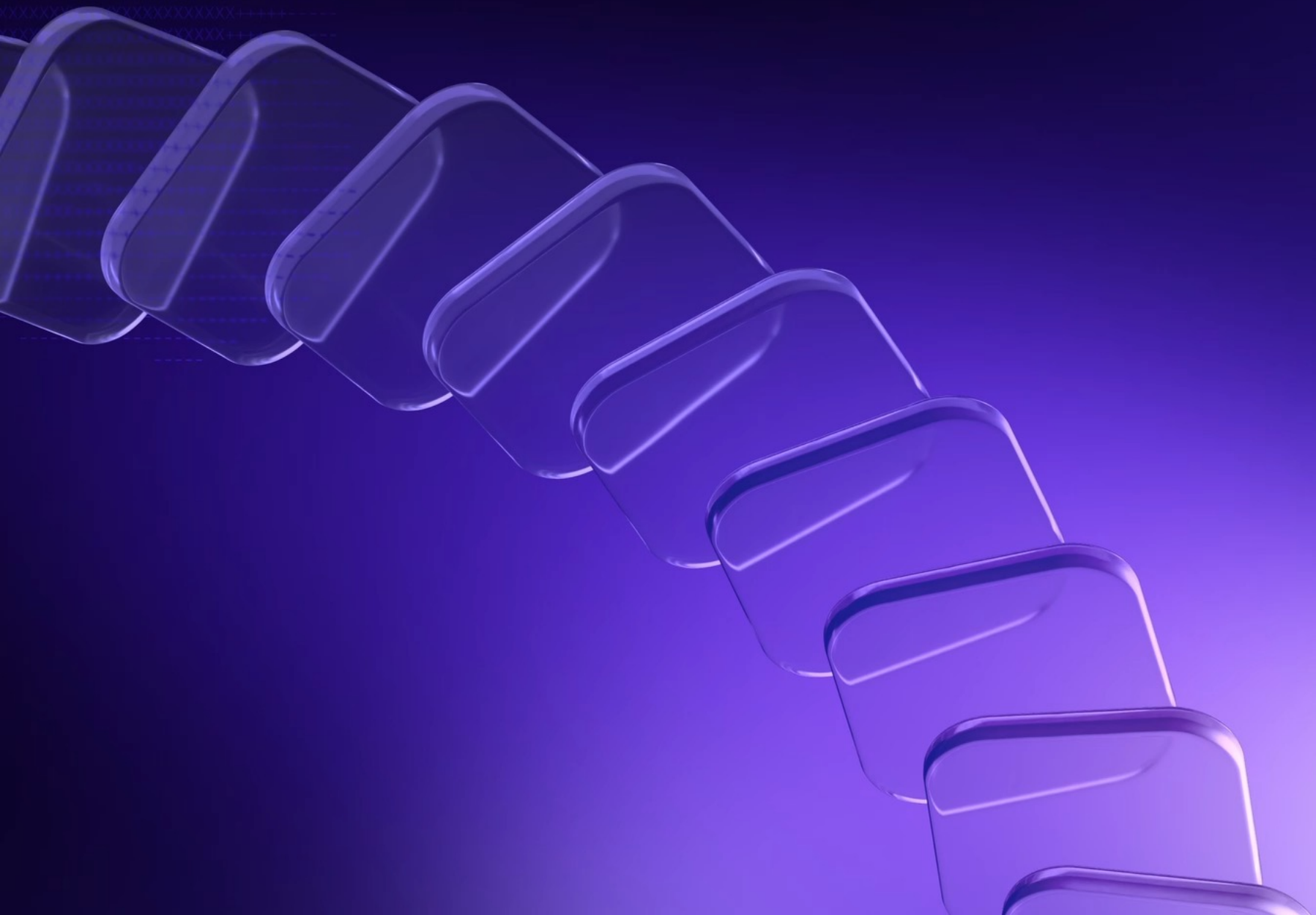
initMAX

permissionsMAX – User overview and PDF

- **What they see and change**
  What the user can view and what they can edit.
- **Where it comes from**
  Which groups and roles grant each permission.
- **Notifications**
  What the user receives and why some never arrive.
- **Activity and role**
  Sign-ins, recent activity and what the role allows.
- **PDF export**
  The full overview as a PDF for review or handover.



initMAX



initMAX

Five templates that turn Zabbix into a security sensor

- **SELinux template**
  mode (enforcing / permissive) and AVC denials via zabbix_agent2 log[] (active checks) – one problem per denial with comm / class / pid tags, auto-closed after {$TRIGGER.CLOSE.TIME}; trigger on any switch to permissive
- **Microsoft Defender template**
  real-time protection state, signature age, last scan, detected threats via PowerShell / WMI (MSFT_MpComputerStatus)
- **ClamAV template**
  clamd & freshclam status, database age, scan results and infected-file counter from the log
- **CVE Zabbix template**
  Zabbix CVEs evaluated for the effective version – taken from zabbix[version] or overridden by macro – CVSS score as severity
  Feed endpoint: https://assets.zabbix.com/files/zabbix_security_advisories.json
- **Windows Update template**
  pending Windows updates, update count and reboot-required flag via Zabbix Agent – alerts on missing update data



initMAX

Security templates available on our GitLab

01
SELinux

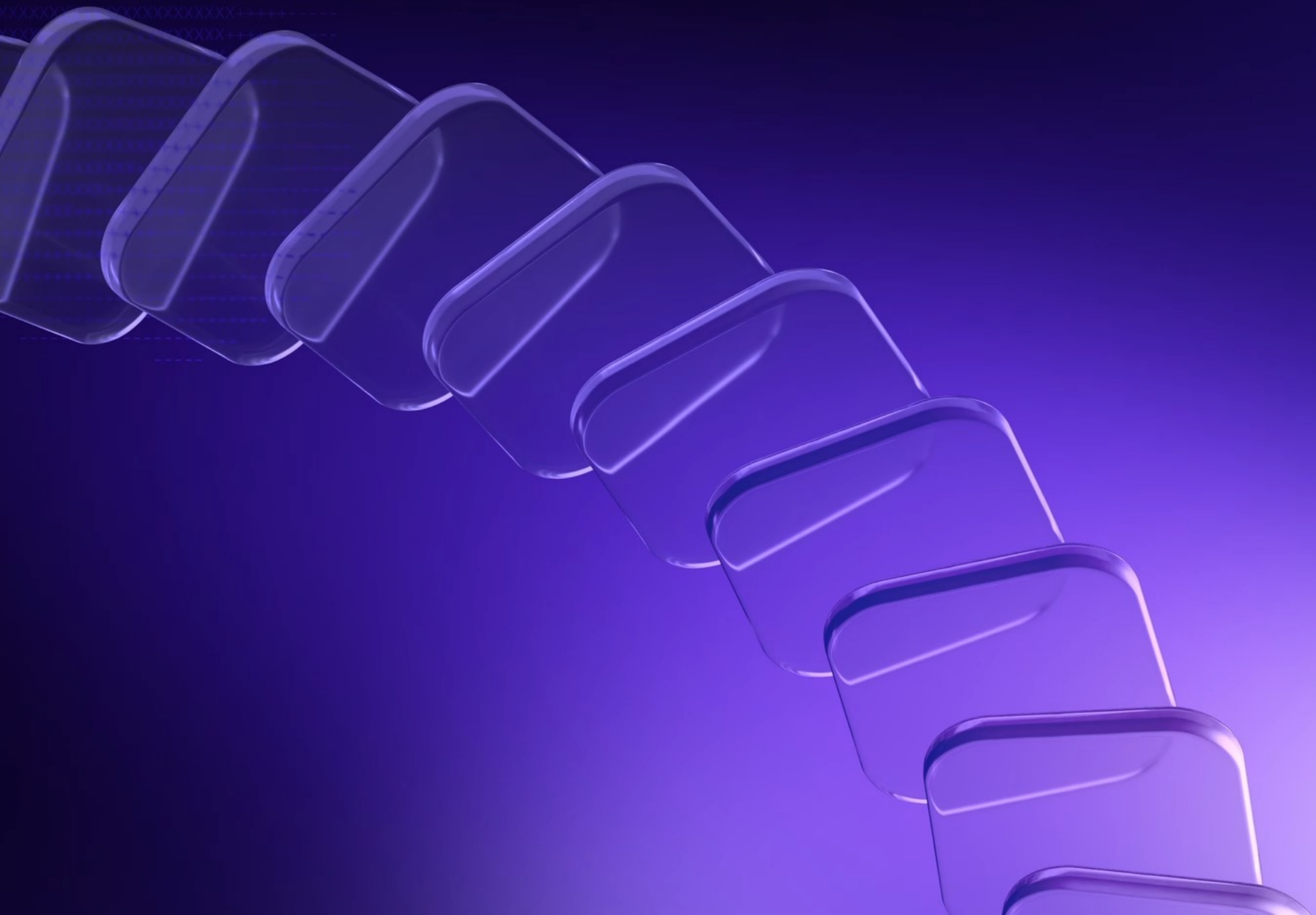
02
Microsoft Defender

03
ClamAV

04
CVE Zabbix

05
Windows Update

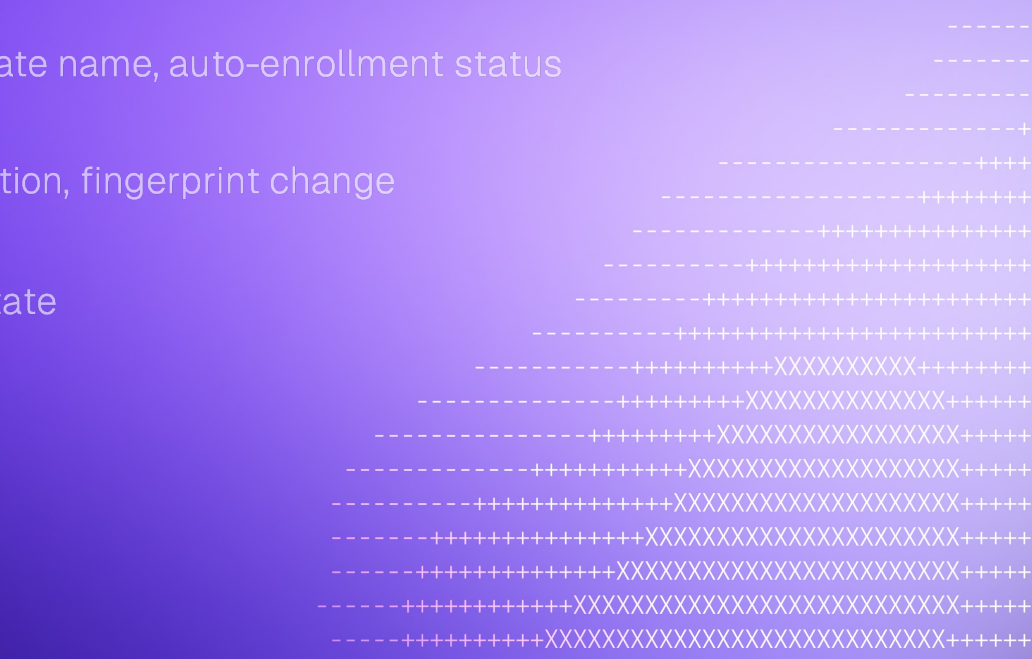
initMAX



initMAX

Certificates – from OS keychains and from the web

- **Web certificates**
  zabbix_agent2 web.certificate.get[host,port] – validity, issuer, SHA-256 fingerprint, chain result; hosts from a list macro
- **System certificates only**
  OS trust stores and service certs – not user / browser certificates (privacy + noise)
- **Linux keychains**
  /etc/pki/ca-trust, /etc/ssl/certs, NSS DBs – LLD over openssl x509 -enddate, tag by path and purpose
- **Windows keychain**
  Cert:\LocalMachine\My & Root via PowerShell – NotAfter, thumbprint, template name, auto-enrollment status
- **Triggers**
  expiry < 30 / 14 / 7 days, unexpected issuer (rogue CA), self-signed in production, fingerprint change
- **Dashboard**
  expiring-soon table + matrixMAX column control:certificate for the whole estate



initMAX

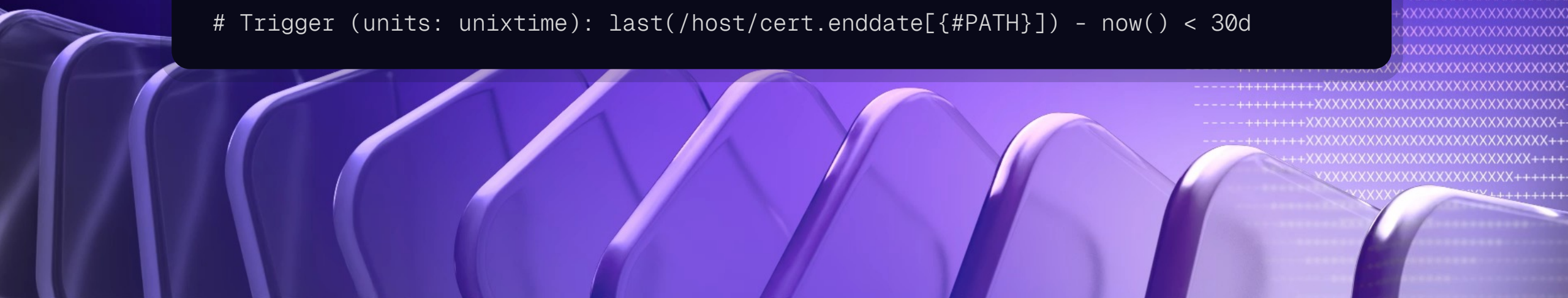
Certificate items – web, Linux keychain, Windows keychain

```
web.certificate.get[{#HOST},{#PORT}]  # agent2 → $.x509.not_after.timestamp
```

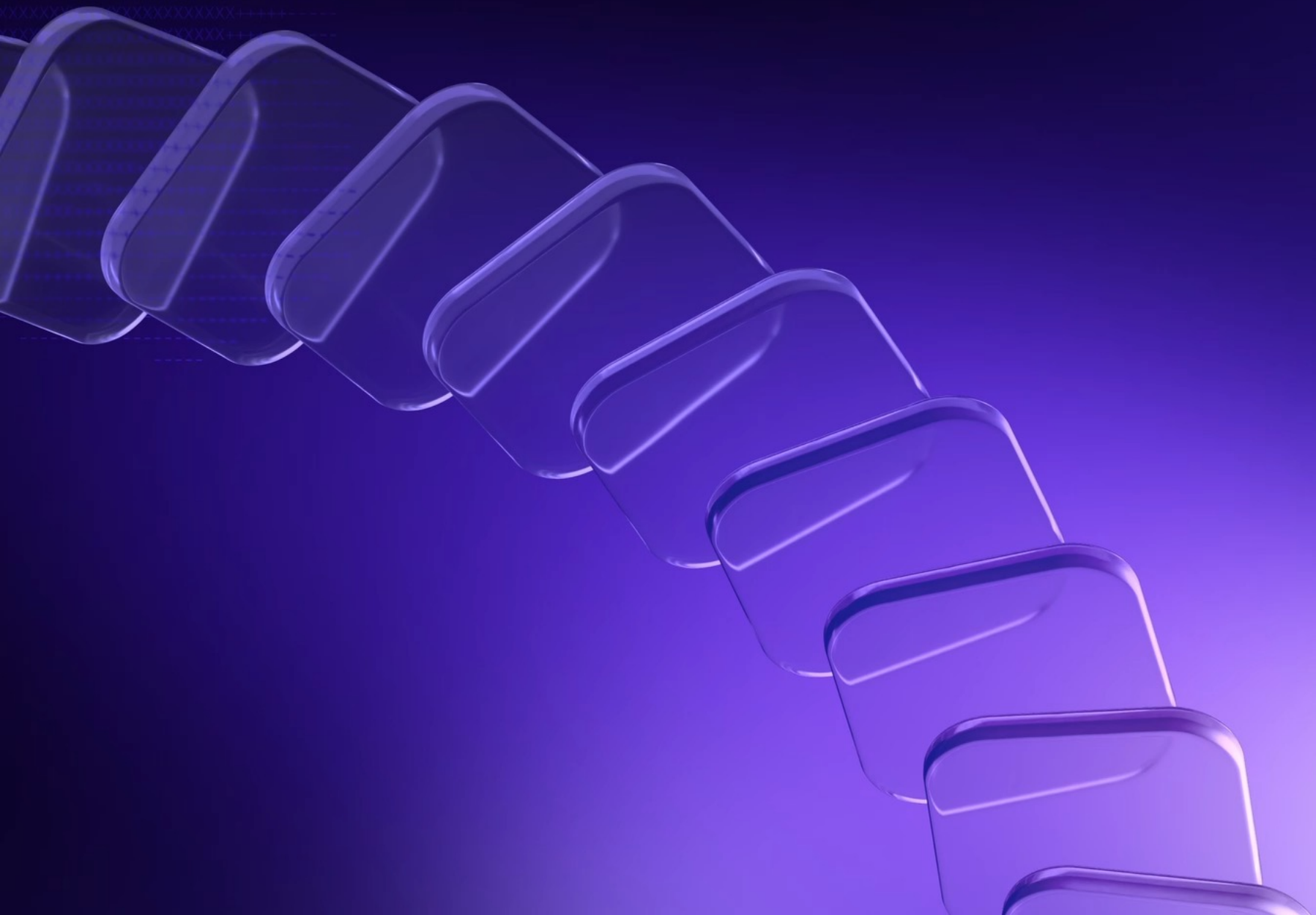
```
# Linux system trust store (UserParameter → unixtime, item type Numeric)
UserParameter=cert.enddate[*],/usr/local/bin/zbx-certcheck $1
# zbx-certcheck: openssl x509 -in "$1" -noout -enddate|cut -d= -f2|date -f- +%s

# Windows machine store (PowerShell, system certs only, NotAfter → unixtime)
Get-ChildItem Cert:\LocalMachine\My,Cert:\LocalMachine\Root |
  Select Subject,Thumbprint,Issuer,
    @{n='NotAfter';e={([DateTimeOffset]$_.NotAfter).ToUnixTimeSeconds()}} |
  ConvertTo-Json -Compress

# Trigger (units: unixtime): last(/host/cert.enddate[{#PATH}]) - now() < 30d
```



initMAX



initMAX

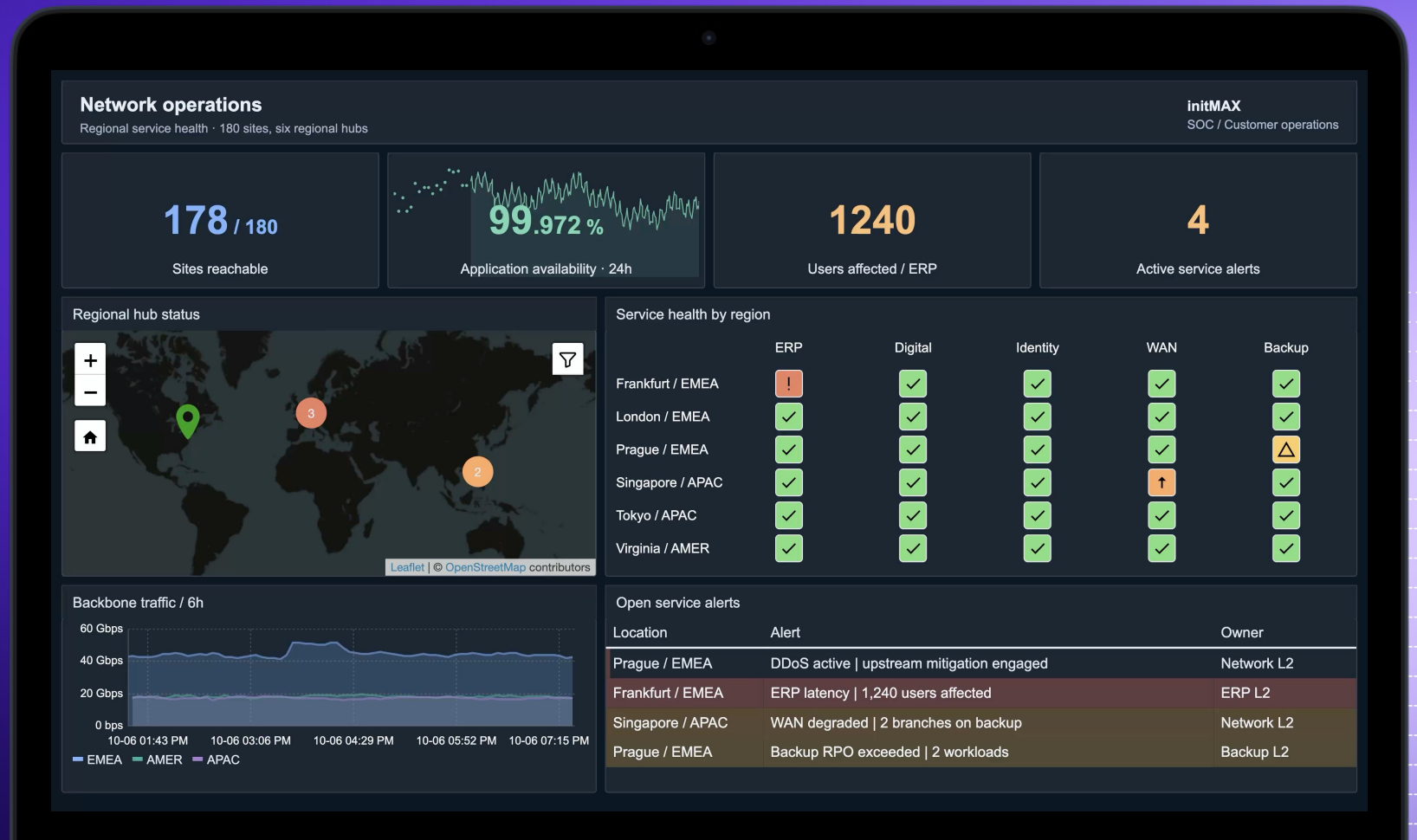
Visualization turns monitoring noise into faster response

- **Data collection is only the start**
  Dashboards turn raw signals into an operational picture.
- **Separate signal from noise**
  In busy environments, critical events can disappear among routine alerts.
- **See the active section at a glance**
  Across dashboards and views, spot where attention is needed first.
- **Investigate and mitigate fast**
  Open the relevant dashboard; navigatorMAX helps locate the affected host.
- **Response time and outcomes matter**
  AI can support triage and prioritization so teams act faster and resolve more effectively.



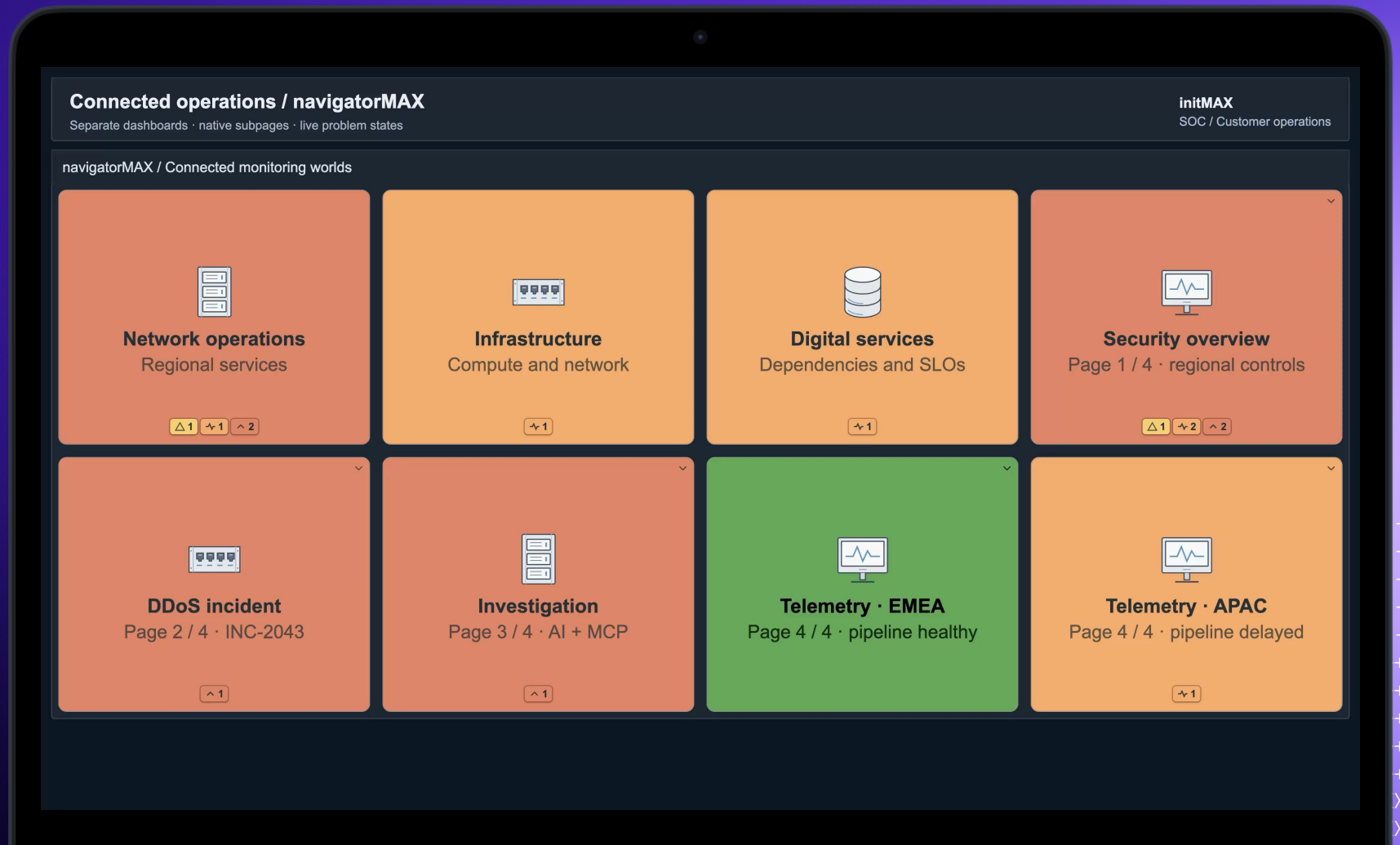
Security & HA dashboards – modules

- Detect the affected service
- Open the incident dashboard
- Investigate the source with MCP
- Approve and verify the response
- Create a Jira ticket and Zabbix ACK



initMAX

Security & HA dashboards – modules and widgets (navigatorMAX)





initMAX

Agent tricks, sudo, gMSA, process whitelisting and SNMPv3

- **Agent runs unprivileged**
  AllowKey / DenyKey whitelist, no system.run[*], no AllowRoot; single commands via sudoers NOPASSWD for exact binaries only.
- **Windows: gMSA instead of local admin**
  Agent under a group Managed Service Account – no stored password, automatic rotation, least-privilege WMI / event log rights.
- **New non-whitelisted process**
  proc.get[] / Win32_Process discovery compared with an allow-list macro → trigger on unknown running executables. Sampling, not enforcement – short-lived processes need auditd / Sysmon, blocking needs fapolicyd / AppLocker.
- **Agent integrity**
  Monitor agent binary checksum (vfs.file.cksum), config file changes and agent version drift across the estate.
- **SNMPv3 only**
  authPriv with SHA-256 / AES-256, unique engine users per device class; SNMPv1/v2c communities disabled.
- **SNMP ACL**
  Device-side ACL / views limit the proxy IP to read-only OIDs; monitor USM auth failures (usmStatsWrongDigests, usmStatsUnknownUserNames).



Agent whitelist, sudo and SNMPv3 in practice

```
# /etc/zabbix/zabbix_agent2.conf
```

```
AllowKey=system.run[/usr/local/bin/zbx-diskcheck]  # exact, no *
AllowKey=system.run[/usr/local/bin/zbx-certcheck]
DenyKey=system.run[*]   # first match wins → Allow goes first
TLSConnect=cert
TLSAccept=cert
TLSServerCertIssuer=CN=initMAX CA

# /etc/sudoers.d/zabbix  (exact binary + pinned arguments)
zabbix ALL=(root) NOPASSWD: /usr/sbin/semanage port -l, \
    /usr/bin/clamdscan --no-summary --infected /srv/upload

# SNMPv3 (net-snmp) - authPriv only
createUser zbxmon SHA-256 "<auth>" AES-256 "<priv>"
rouser zbxmon priv -V systemonly   # view = ACL on OIDs
```



initMAX

Guy behind all of this...





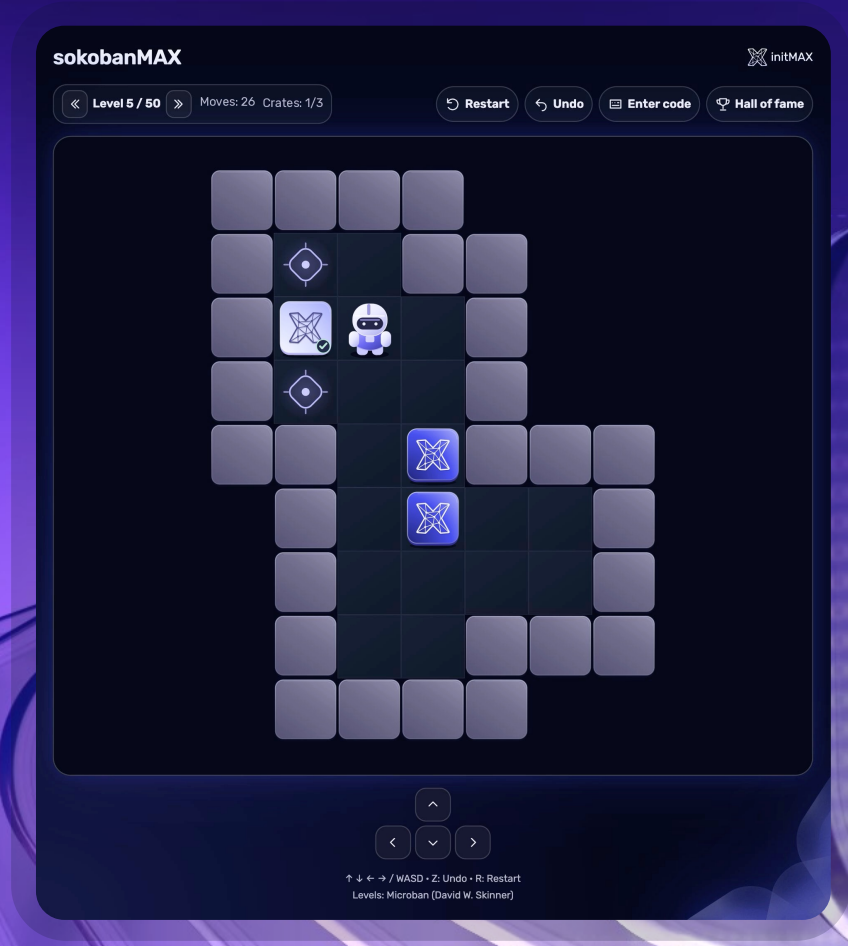
initMAX

Same as every year a few gifts for you

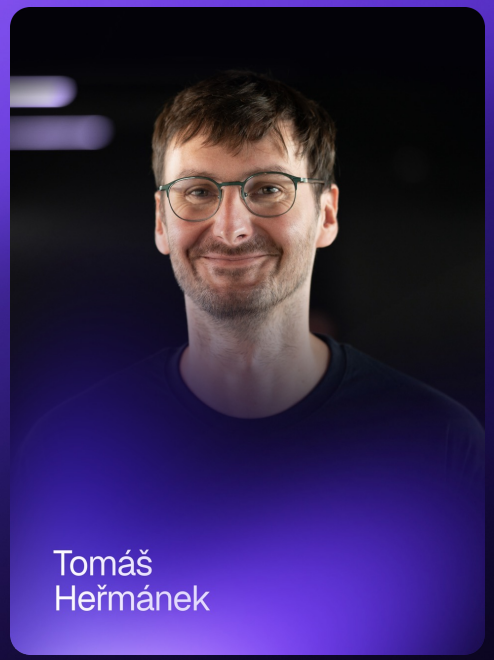
- **chessMAX**
  Chess widget for your Zabbix dashboard – play against bundled Stockfish (4 levels), invite a colleague or share one screen, optional chess clock.

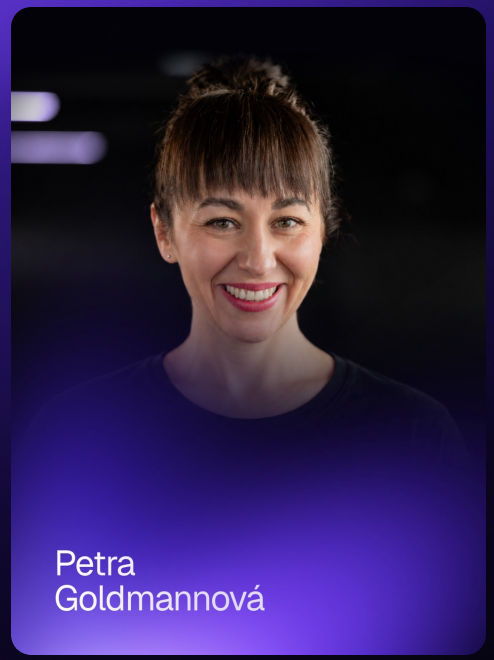
- **sokobanMAX**
  Sokoban puzzle widget for your Zabbix dashboard – classic box-pushing levels, runs entirely in the browser, no external services.

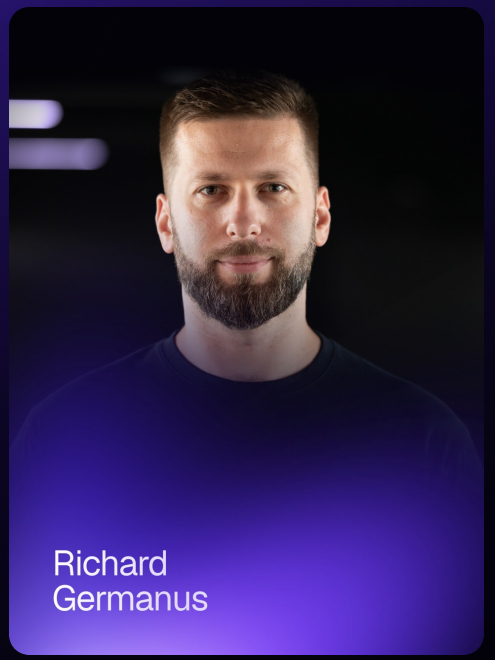
- **Download for free from the initMAX Portal**
  https://portal.initmax.com/

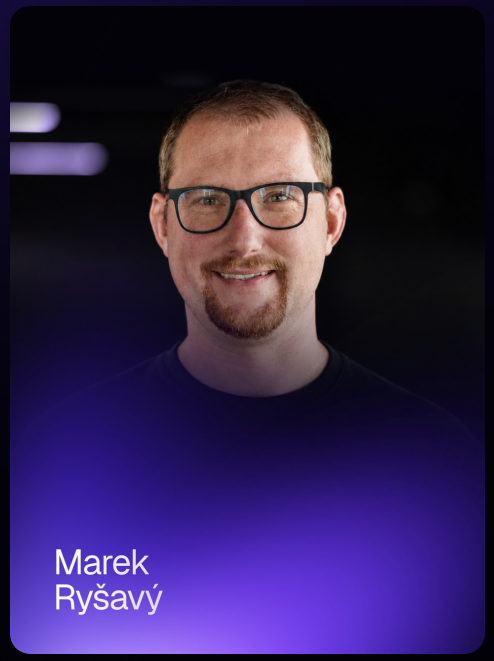


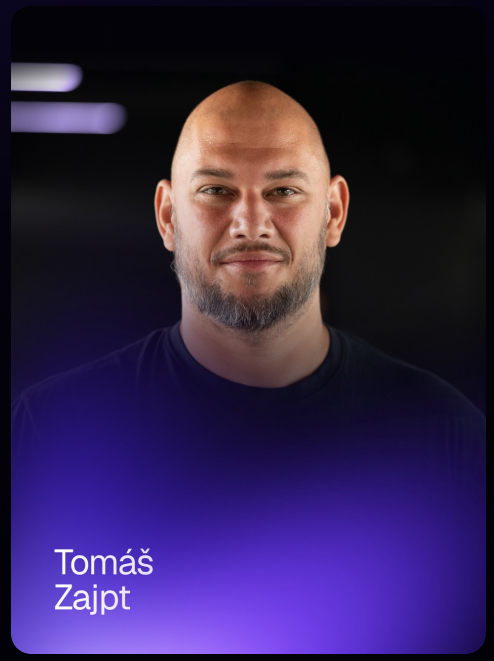
Visit us at our stand!

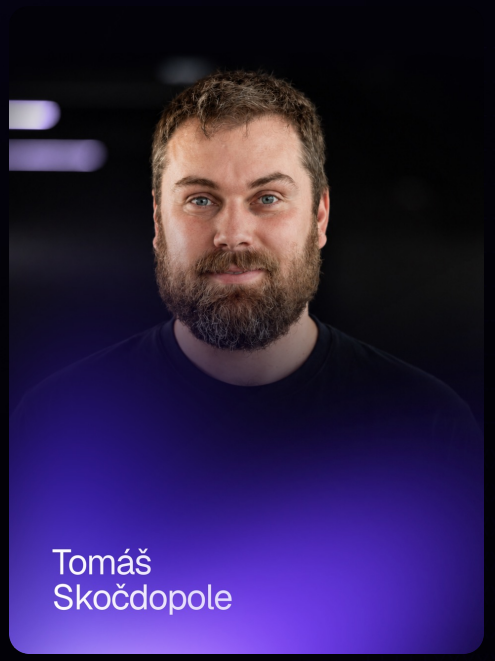












initMAX

VISIT OUR WEBSITE
WWW.INITMAX.COM

WEBINARS / WIKI / E-SHOP

VISIT OUR PORTAL
PORTAL.INITMAX.COM

MODULES / WIDGETS / RELEASES

VISIT OUR GITHUB
GITHUB.COM/INITMAX

TEMPLATES / SCRIPTS / TOOLS

initMAX

THANK YOU

initMAX

Q&A