CrowdSec sperrt böswillige IP-Adressen

Wer Dienste im Internet betreibt, muss sich früher oder später mit unliebsamen Besuchern befassen. Zur Abwehr gibt es mit Fail2ban und Co. eine Reihe von Tools. CrowdSec ist eine modernere Alternative, die sich auch im Verbund nutzen lässt.

Von Lukas Grimm und Reto Kupferschmid

Öffentlich erreichbare Dienste wie Web- oder Mailserver werden seit jeher von vielen unerwünschten Besuchern heimgesucht. Darunter befinden sich einerseits Bots und Script-Kiddies, andererseits gibt es gezielte Angriffe. Viele Administratoren nutzen daher Werkzeuge wie Fail2ban, SSHGuard oder DenyHosts zur Absicherung. Mit CrowdSec existiert seit wenigen Jahren eine moderne Open-Source-Alternative. Dieser Artikel geht näher auf das Featureset ein und zeigt anhand zweier Beispiele, wie CrowdSec die bestehenden Tools ergänzen oder ablösen kann.

Im Gegensatz zu Fail2ban sind bei CrowdSec die Entscheidungen, welche Clients geblockt werden, nicht nur lokal verfügbar, sondern lassen sich über mehrere Systeme teilen. Und das nicht nur in einem lokalen Cluster, sondern mit der ganzen CrowdSec-Community.

Hier muss man sich allerdings Gedanken zum Datenschutz machen, denn die dazu erforderliche CrowdSec Central API ist per Default aktiv: CrowdSec teilt die angreifende IP, einen Zeitstempel und die Art der festgestellten Attacke mit der Community. Nutzer sollten dies bedenken und überlegen, ob sie die Crowd Sec Central API im eigenen Set-up besser deaktivieren.

CrowdSec hat noch zwei weitere Vorteile gegenüber Fail2ban: Zum einen kann es mehr Attacken erkennen, wie Brute-Force- und verschiedene Layer-7-Angriffe wie XSS, SQL Injections, Bot Scraping et cetera. Zum andern kann CrowdSec auf mehreren Ebenen blockieren. So kann es beispielsweise – anstelle einer Sperrung auf Firewallebene – dem User ein Captcha anzeigen, wenn es ein auffälliges Verhalten feststellt. So sperrt das Tool Menschen nie komplett aus. Im Blog von CrowdSec findet sich ein ausführlicher Vergleich zwischen CrowdSec und Fail2ban (siehe ix.de/z3bp).

Einen CrowdSec-Cluster kann man lokal aufbauen oder mit der zentralen CrowdSec Central API global organisieren. Über die zentrale API kann man die gesammelten Cyber-Threat-Daten teilen und erhält im Gegenzug die Daten aller anderen User. CrowdSec verspricht, die Informationen zu kuratieren und zu aktualisieren. Es gibt ein Webfrontend, über das man die eigenen Daten prüfen kann (siehe ix.de/z3bp).

CrowdSec-Komponenten

CrowdSec besteht aus mehreren Komponenten, die Hand in Hand zusammenarbeiten. Herzstück ist der CrowdSec Agent. Er analysiert die konfigurierten Logdateien auf dem System in Echtzeit und identifiziert potenzielle Angriffe. Treffer leitet er an die CrowdSec Local API (lapi) und je nach Konfiguration an die Central API weiter.

Bouncer sind die Komponenten, die sich um das Stoppen der Bedrohung kümmern. Sie beziehen die vom Agent ausfindig gemachten IP-Adressen über die CrowdSec API. Es gibt verschiedene Bouncer für unterschiedliche Dienste, so blockt der Firewall-Bouncer Angriffe auf der Hostfirewall eines Systems, während der nginx-Bouncer eine Fehlermeldung anzeigt oder Benutzern ein Captcha präsentiert.

Im Gegensatz zur CrowdSec Central API ist die Local API (lapi) nur lokal verfügbar. Die anderen Agents im Cluster verbinden sich mit der lapi und senden die lokalen Entscheidungen dorthin. Sollte es Datenschutzbedenken beim Einsatz der CrowdSec Central API geben, lässt sich so zumindest ein lokaler Crowd Sec-Cluster bilden.

Interaktion mit CrowdSec per cscli

Das CLI zur Interaktion mit CrowdSec heißt `cscli`. Im Beispiel soll CrowSec anhand von nginx-Logdateien Angriffe und andere unerwünschte Besucher einer

iX-TRACT

- CrowdSec ist eine moderne Open-Source-Software zum Schutz vor Cyberbedrohungen, die anhand von Logdateien böswillige IP-Adressen erkennen und blockieren kann.
- Anders als traditionelle Tools wie Fail2ban ist CrowdSec darauf ausgelegt, im Verbund zu arbeiten und die erkannten Bedrohungen über mehrere Systeme hinweg zu teilen.
- Dank vorgefertigter Parser und Szenarios lässt sich die Software leicht in bestehende Systeme wie nginx oder OpenSSH integrieren.

Monitoring

Sowohl CrowdSec als auch der Firewall-Bouncer unterstützen Endpunkte, die Metriken im OpenMetrics-Format bereitstellen. Sie kann ein Tool wie Prometheus einsammeln und über Grafana visualisieren. CrowdSec bietet dazu im Repository crowdsecurity/grafana-dashboards vorgefertigte Dashboards an.

Webseite detektieren und die betroffenen IP-Adressen blockieren. Als Testumgebung dienen drei virtuelle Maschinen mit Rocky Linux 9, wobei crowdsec01 und crowdsec02 als Server arbeiten und mallory01 die Angriffe simuliert.

In einem ersten Schritt sind die nötigen Repositorys zu konfigurieren, um anschließend das Paket crowdsec über den Paketmanager zu installieren. Die Dokumentation von CrowdSec empfiehlt dabei die Konfiguration der Repositorys über ein Skript. Besser ist jedoch, das Repository manuell zu installieren und – wie in Listing 1 gezeigt – in /etc/yum.repos.d entsprechende Repo-Dateien anzulegen. Das ist nicht nur sicherer als die in der Dokumentation empfohlene Methode per `curl | sudo sh`, sondern lässt sich auch einfach über ein Konfigurationsmanagement wie Ansible automatisieren.

Sobald die Repositorys konfiguriert sind, installiert der Befehl `sudo dnf install crowdsec` die CrowdSec Security Engine. Per `cscli` lassen sich nun die nötigen Collections vom CrowdSec Hub installieren. Im Beispiel ist das `crowdsecurity/nginx`:

```
ansible@crowdsec01:~$ sudo cscli ↵
                collections install ↵
                  crowdsecurity/nginx
ansible@crowdsec01:~$ sudo ↵
            systemctl reload crowdsec
```

Diese Collection enthält sowohl einen Parser als auch einige grundlegende HTTP-Szenarien. Der Parser ist für die Analyse der nginx-Logdateien zuständig, während die Szenarien bestimmte Aktionen (beispielsweise Website-Scans oder -Crawls) erkennen können.

Im nächsten Schritt muss man CrowdSec mitteilen, wo es die nginx-Logs findet. Listing 2 zeigt das Erstellen der Datei nginx.yaml. Anschließend aktiviert ein CrowdSec-Neustart per `sudo systemctl restart crowdsec` die neue Konfiguration.

Mit diesem Paket sind alle nötigen Tools installiert, um Angriffe zu erkennen. Was nun noch fehlt, ist ein passender Bouncer. Diese Komponente nimmt die von CrowdSec generierten Entscheidungen entgegen und trifft auf dem System die Vorkehrungen, um weitere Angriffe dieser IP-Adresse zu unterbinden oder zu erschweren. Im Beispiel blockiert der Firewall-Bouncer die IP-Adresse auf der Hostfirewall der Linux-Maschine. Die Installation erfolgt über das im vorherigen Schritt angelegte Repository:

```
ansible@crowdsec01:~$ sudo dnf ↵
          install crowdsec-firewall-↵
                      bouncer-iptables
```

Bouncer und CrowdSec kommunizieren über die lapi. Bei der Installation des Bouncers registriert er sich automatisch über die API und ist damit bereit, Entscheidungen von CrowdSec entgegenzunehmen und IP-Adressen zu blocken. Über den Befehl `cscli bouncer list` lässt sich überprüfen, ob die Verbindung zwischen CrowdSec und dem Bouncer einwandfrei funktioniert (siehe Listing 3).

Als Test, ob der Bouncer tatsächlich IP-Adressen blockiert, dienen einige Anfragen auf eine nicht existierende URL von einem zweiten Server aus. Gleichzeitig beobachtet man die CrowdSec-Logs auf dem Zielsystem und prüft, ob durch die Anfragen das erwartete Szenario `http-crawl` getriggert und die IP-Adresse blockiert wird (siehe Listing 4).

Wie erwartet wird die IP-Adresse des Clients (5.102.145.23) erkannt und vom Firewall-Bouncer blockiert. Die in den Logs aufgeführte `ban time` von vier Stunden ist der in der Datei /etc/crowdsec/profiles.yaml definierte Defaultwert für die Dauer der Sperre. Sie lässt sich beliebig anpassen, wobei CrowdSec hier neben statischen Werten auch dynamische Ausdrücke erlaubt, um zusätzliche Features wie längere Sperren für Wiederholungstäter umzusetzen.

Troubleshooting

Neben den Hinweisen in den Logdateien lassen sich die blockierten IP-Adressen auch über das CLI auslesen und bei Bedarf

Listing 1: Repo-Dateien für CrowdSec anlegen

```
$ cat << 'EOF' | sudo tee /etc/yum.repos.d/crowdsec.repo
[crowdsec_crowdsec]
name=crowdsec_crowdsec
baseurl=https://packagecloud.io/crowdsec/crowdsec/el/9/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://packagecloud.io/crowdsec/crowdsec/gpgkey
       https://packagecloud.io/crowdsec/crowdsec/gpgkey/crowdsec-crowdsec-↵
                                                          EDE2C695EC9A5A5C.pub.gpg
       https://packagecloud.io/crowdsec/crowdsec/gpgkey/crowdsec-crowdsec-↵
                                                          C822EDD6B39954A1.pub.gpg
       https://packagecloud.io/crowdsec/crowdsec/gpgkey/crowdsec-crowdsec-↵
                                                          FED78314A2468CCF.pub.gpg
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
EOF

$ cat << 'EOF' | sudo tee /etc/yum.repos.d/crowdsec-source.repo
[crowdsec_crowdsec-source]
name=crowdsec_crowdsec-source
baseurl=https://packagecloud.io/crowdsec/crowdsec/el/9/SRPMS
repo_gpgcheck=1
gpgcheck=1
enabled=1
gpgkey=https://packagecloud.io/crowdsec/crowdsec/gpgkey
       https://packagecloud.io/crowdsec/crowdsec/gpgkey/crowdsec-crowdsec-↵
                                                          EDE2C695EC9A5A5C.pub.gpg
       https://packagecloud.io/crowdsec/crowdsec/gpgkey/crowdsec-crowdsec-↵
                                                          C822EDD6B39954A1.pub.gpg
       https://packagecloud.io/crowdsec/crowdsec/gpgkey/crowdsec-crowdsec-↵
                                                          FED78314A2468CCF.pub.gpg
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
EOF
```

Listing 2: Die Datei nginx.yaml erstellen

```
$ sudo mkdir /etc/crowdsec/acquis.d
$ cat << 'EOF' | sudo tee /etc/crowdsec/acquis.d/nginx.yaml
---
source: file
filenames:
  - /var/log/nginx/*.log
labels:
  type: nginx
EOF
```

wieder entsperren (siehe Listing 5). Der erste Aufruf zeigt die aktuell blockierten IP-Adressen an, der zweite gibt eine IP-Adresse wieder frei.

Blick hinter die Kulissen

Zusätzlich bietet `cscli` zahlreiche Funktionen. So lassen sich über den Unterbefehl `explain` anhand einer oder mehrerer Logzeilen die einzelnen Verarbeitungsschritte darstellen. So kann man prüfen, ob CrowdSec die Logzeilen korrekt parst und welche Szenarien die Logzeilen auslösen. Zur Veranschaulichung dient ein Beispiel, das die letzte Zeile aus der Datei access.log von nginx an den Befehl `cscli explain` übergibt (siehe Listing 6). Das Flag `type` definiert dabei, von welcher Applikation die Logzeile stammt. Im Normalbetrieb wird dieser Typ anhand der jeweiligen Acquisition-Konfiguration (im Beispiel aus der Datei /etc/crowdsec/acquis.d/nginx.yaml) automatisch gesetzt.

Anhand der Farben lässt sich erkennen, ob der jeweilige Schritt in der Pipeline auf die Logzeile angewendet wurde. So ist in Listing 6 der Parser crowdsecurity/syslog-logs rot dargestellt, da das Format der Logzeile nicht dem erwarteten Format des Parsers entspricht. Ist die Logzeile erfolgreich verarbeitet, erscheint dieser Schritt in Grün. Da ein Parser die Logs mit zusätzlichen Informationen anreichern kann, gibt cscli in diesem Fall die Zahl der hinzugefügten (beispielsweise +5) respektive veränderten (~8) Felder ebenfalls aus.

CrowdSec verarbeitet Logs in Pipelines, wobei die Pipeline in drei Abschnitte (Stages) unterteilt ist. Jede dieser drei Stages besteht aus einem oder mehreren Parsern, die die Logzeile in einzelne Felder auftrennen und die Logs optional mit zusätzlichen Informationen anreichern.

In der ersten Stage namens `s00-raw` findet zunächst eine grobe Einteilung der

Listing 3: Verbindungstest Bouncer – CrowdSec

```
$ sudo cscli bouncer list

Name                              IP Address  Valid Last API pull         Type                      Version                      Auth Type
cs-firewall-bouncer-1723724566  127.0.0.1   ✔     2024-08-15T12:24:21Z  crowdsec-firewall-bouncer  v0.0.28-el9-rpm-↵
                                                                                                    af6e7e25822c2b1a02168b99ebbf8458bc6728e5  api-key
```

Listing 4: Testlauf für Szenario http-crawl

```
# Auf dem Client:

$ for i in {1..200}; do curl --silent http://5.102.145.14/$RANDOM > /dev/null ; done

# Auf dem Server:

$ sudo tail -n0 -f /var/log/crowdsec-firewall-bouncer.log /var/log/crowdsec.log

==> /var/log/crowdsec.log <==
time="2024-08-15T14:47:14+02:00" level=info msg="Ip 5.102.145.23 performed 'crowdsecurity/http-crawl-non_statics' ↵
                                                 (47 events over 3.301494736s) at 2024-08-15 12:47:14.407318385 +0000 UTC"
time="2024-08-15T14:47:14+02:00" level=info msg="Ip 5.102.145.23 performed 'crowdsecurity/http-crawl-non_statics' ↵
                                                 (47 events over 3.302069432s) at 2024-08-15 12:47:14.407899792 +0000 UTC"
time="2024-08-15T14:47:14+02:00" level=info msg="(8b9b722869604cf8aad593a3ebf96eef3OhKRXTtkftzmcNS/crowdsec) ↵
                                                 crowdsecurity/http-crawl-non_statics by ip 5.102.145.23 (CH/59414) : 4h ban on Ip 5.102.145.23"
time="2024-08-15T14:47:14+02:00" level=info msg="(8b9b722869604cf8aad593a3ebf96eef3OhKRXTtkftzmcNS/crowdsec) ↵
                                                 crowdsecurity/http-crawl-non_statics by ip 5.102.145.23 (CH/59414) : 4h ban on Ip 5.102.145.23"

==> /var/log/crowdsec-firewall-bouncer.log <==
time="15-08-2024 14:47:21" level=info msg="1 decision added"
```

Listing 5: Blockierte IP-Adressen auslesen oder entsperren

```
# aktuelle blockierte Adressen auslesen:
$ sudo cscli decisions list
┌────────┬──────────┬──────────────────┬──────────────────────────────────────┬────────┬─────────┬──────────────────────────┬────────┬──────────┐
│   ID   │  Source  │   Scope:Value    │                Reason                │ Action │ Country │            AS            │ Events │ expiration │ Alert ID │
├────────┼──────────┼──────────────────┼──────────────────────────────────────┼────────┼─────────┼──────────────────────────┼────────┼──────────┤
│ 585019 │ crowdsec │ Ip:5.102.145.23 │ crowdsecurity/http-crawl-non_statics │ ban    │ CH      │ 59414 cloudscale.ch AG │ 41     │ 3h56m29.65362144s │ 58       │
└────────┴──────────┴──────────────────┴──────────────────────────────────────┴────────┴─────────┴──────────────────────────┴────────┴──────────┘
2 duplicated entries skipped

# IP-Adressen explizit entsperren

$ sudo cscli decisions delete -i 5.102.145.23
INFO 1 decision(s) deleted
```

Logs statt. Im Beispiel wird zwischen Syslogs und Non-Syslogs (Logs von Applikationen wie nginx, Apache oder MariaDB) unterschieden. Ein weiterer Anwendungsfall dieser Stage wäre das Extrahieren von Logzeilen aus JSON-Strukturen.

Die zweite Stufe, `s01-parse`, enthält die applikationsspezifischen Parser. Sie trennen die Logzeilen anhand des Logformats der jeweiligen Anwendung in einzelne Felder auf (bei den hier im Beispiel genutzen nginx-Logs etwa in HTTP-Statuscode, IP-Adresse, HTTP-User-Agent et cetera).

Nimmt man exemplarisch die in Listing 7 gezeigte Logzeile, extrahiert der Parser daraus die in Listing 8 gezeigten Felder. Sie können von den nachfolgenden Parsern genutzt werden und fließen aber später auch in die Entscheidung im Szenario ein. Sie beeinflussen damit, ob eine IP-Adresse blockiert wird oder nicht.

Als letzte Stufe der Pipeline reichert `s02-enrich` die Logs mit zusätzlichen Informationen wie Geodaten oder Reverse DNS der IP-Adresse an. Sobald die letzte Stage die Logzeile verarbeitet hat, vergleicht CrowdSec die angereicherten Logs mit den konfigurierten Szenarien. Im hier betrachteten Beispiel löst die Logzeile die beiden Szenarien `http-crawl-non_statics` und `http-probing` aus. Um mehr über die Konfiguration der einzelnen Szenarien zu erfahren, empfiehlt sich ein Blick in deren YAML-Definitionen. Den Pfad dorthin liefert das Kommando `cscli scenarios list`.

Listing 6: Verarbeitungsschritte per cscli explain nachvollziehen

```
$ sudo tail -n 1 /var/log/nginx/access.log | sudo cscli explain --type nginx ↵
                                                                    --file -

line: 5.102.145.23 - - [18/Aug/2024:19:49:28 +0200] "GET /15926 HTTP/1.1" 404 ↵
                                                     3332 "-" "curl/7.76.1" "-"
         ├ s00-raw
         |       ├ 🔴 crowdsecurity/syslog-logs
         |       └ 🟢 crowdsecurity/non-syslog (+5 ~8)
         ├ s01-parse
         |       └ 🟢 crowdsecurity/nginx-logs (+22 ~2)
         ├ s02-enrich
         |       ├ 🟢 crowdsecurity/dateparse-enrich (+2 ~2)
         |       ├ 🟢 crowdsecurity/geoip-enrich (+13)
         |       ├ 🟢 crowdsecurity/http-logs (+7)
         |       └ 🟢 crowdsecurity/whitelists (unchanged)
         ├-------- parser success 🟢
         ├ Scenarios
                 ├ 🟢 crowdsecurity/http-crawl-non_statics
                 └ 🟢 crowdsecurity/http-probing
```

CrowdSec auf verteilten Systemen

Aufbauend auf dem ersten Beispiel wird die Konfiguration für das nächste Set-up nun so angepasst, dass ein zusätzlicher Server Daten in die CrowdSec API einspeist. Der große Vorteil eines solchen Set-ups ist, dass die Entscheidungen über die zu blockierenden IP-Adressen direkt

Listing 7: nginx-Logzeile

```
5.102.145.23 - - [19/Aug/2024:16:47:23 +0200] "GET /15695 HTTP/1.1" 404 3332 "-" "curl/7.76.1" "-"
```

Listing 8: Vom Parser extrahierte Felder

```
evt.Parsed.http_version : 1.1
evt.Parsed.request : /15695
evt.Parsed.verb : GET
evt.Parsed.remote_addr : 5.102.145.23
evt.Parsed.request_length :
evt.Parsed.time_local : 19/Aug/2024:16:47:23 +0200
evt.Parsed.body_bytes_sent : 3332
evt.Parsed.http_referer : -
evt.Parsed.http_user_agent : curl/7.76.1
evt.Parsed.proxy_upstream_name :
evt.Parsed.remote_user : -
evt.Parsed.target_fqdn :
evt.Parsed.proxy_alternative_upstream_name :
evt.Parsed.request_time :
evt.Parsed.status : 404
evt.Meta.http_user_agent : curl/7.76.1
evt.Meta.http_verb : GET
evt.Meta.http_path : /15695
evt.Meta.http_status : 404
evt.Meta.log_type : http_access-log
evt.Meta.service : http
evt.Meta.source_ip : 5.102.145.23
```

Listing 9: Netzwerkkonfiguration CrowdSec API

```
$ sudo sed -i.bak 's#listen_uri: 127.0.0.1:8080#listen_uri: 10.110.0.41:8080#g'↵
                                                    /etc/crowdsec/config.yaml
$ sudo sed -i.bak 's#url: http://127.0.0.1:8080#url: http://10.110.0.41:8080#g'↵
                                          /etc/crowdsec/local_api_credentials.yaml
$ sudo sed -i.bak 's#api_url: http://127.0.0.1:8080/#api_url: ↵
   http://10.110.0.41:8080/#g' /etc/crowdsec/bouncers/crowdsec-firewall-bouncer.
yaml
```

auf allen beteiligten Maschinen angewendet werden. Dazu ist zuerst die CrowdSec API auf dem vorhandenden Server so zu konfigurieren, dass sie über das Netzwerk erreichbar ist; Listing 9 zeigt die Schritte .

In produktiven Umgebungen sollte man an dieser Stelle die TLS-Konfiguration gemäß der Anleitung vornehmen (siehe ix.de/z3bp). Das stellt sicher, dass die Übermittlung der verwendeten API-Keys an die CrowdSec Local API verschlüsselt erfolgt.

Zusätzlich ist in der Konfigurationsdatei /etc/crowdsec/config.yaml unter `api.server.trusted_ips` die IP-Adresse des zweiten Servers freizugeben. Anschließend ist ein Neustart von crowdsec und crowdsec-firewall-bouncer nötig:

```
sudo systemctl restart crowdsec.↵
  service crowdsec-firewall-bouncer.↵
                                    service
```

Den zweiten Server konfiguriert man analog zum ersten (Konfiguration der Repositorys, Installation der Pakete und der Collection crowdsecurity/nginx sowie Konfiguration der Acquisition). Damit der neue Server nun die Local API des ersten verwendet, wird er dort registriert:

```
sudo cscli lapi register -u ↵
                  http://<IP-Adresse-Server1>
```

Diese Registrierung muss noch bestätigt werden. Dazu benötigt man den Namen der neu registrierten Maschine und gibt ihn per `cscli machines validate` frei (siehe Listing 10).

Im letzten Schritt folgt die Konfiguration des Bouncers auf dem neuen Server.

WERTUNG

CrowdSec: Open-Source-Engine zur Verhaltenserkennung

- ⊕ Open Source
- ⊕ IPv6-kompatibel
- ⊕ schnell
- ⊕ einfache Inbetriebnahme
- ⊕ Austausch mit der Community
- ⊖ Communityaustausch datenschutzrechtlich bedenklich

Listing 10: API-Registrierung bestätigen

```
$ sudo cscli machines list

 Name                                              IP Address   Last Update           Status  ↵
                                                     Version                                    Auth Type  Last Heartbeat
 8b9b722869604cf8aad593a3ebf96eef3OhKRXTtkftzmcNS  10.110.0.41  2024-08-19T14:02:00Z  ✔↵
                                                     v1.6.2-rpm-pragmatic-amd64-16bfab86-linux  password           31s
 8b9b722869604cf8aad593a3ebf96eefr0Hcx7aIQndVXodL  10.110.0.42  2024-08-19T13:59:57Z  🚫 ↵
                                                                                                password           ⚠ 2m34s

$ sudo cscli machines validate 8b9b722869604cf8aad593a3ebf96eefr0Hcx7aIQndVXodL

 INFO machine '8b9b722869604cf8aad593a3ebf96eefr0Hcx7aIQndVXodL' validated successfully
```

Listing 11: API-Keys für Crowd-Betrieb einrichten

```
$ sed -i 's#api_url.*#api_url: http://10.110.0.41:8080#' /etc/crowdsec/bouncers/crowdsec-firewall-bouncer.yaml
$ sed -i 's#api_key.*#api_key: 3/peGU/6P1h2pUkp2wb/E/1XZi5uNsggQc6HnVwfQP0#' ↵
                                                        /etc/crowdsec/bouncers/crowdsec-firewall-bouncer.yaml
$ systemctl restart crowdsec-firewall-bouncer.service
```

Wie im ersten Beispiel bereits gezeigt, registriert sich dieser bei der Installation an der lapi auf dem installierten Server. Für den Fall dieses Beispiels soll sich der Bouncer jedoch mit der API des ersten Servers verbinden. Das erfordert dort zunächst eine Registrierung:

```
$ sudo cscli bouncers add cs-↵
          firewall-bouncer-crowdsec02
API key for 'cs-firewall-bouncer-↵
                          crowdsec02':
  3/peGU/6P1h2pUkp2wb/↵
              E/1XZi5uNsggQc6HnVwfQP0

Please keep this key since you will ↵
          not be able to retrieve it!
```

Den angezeigten API-Key überträgt man in die Konfigurationsdatei /etc/crowdsec/bouncers/crowdsec-firewall-bouncer.yaml des neuen Servers und startet anschließend crowdsec-firewall-bouncer neu (siehe Listing 11). Wichtig ist hier aber, stets den API-Key aus der lokalen CrowdSec-Installation zu verwenden.

Nun ist alles bereit für einen erneuten Test. Dazu dient wieder der schon in Listing 4 genutzte Befehl auf dem Client, der Anfragen auf eine nicht existierende URL ausführt. Nach wenigen Sekunden erscheint dann in den Logs beider Firewall-Bouncer der erwartete Eintrag `1 decision added`. Zusätzlich kann man prüfen, ob die IP-Adresse des Clients (5.102.145.23) im ipset erscheint:

```
$ sudo ipset list | grep↵
                          "5.102.145.23"
5.102.145.23 timeout 14053
```

Fazit

CrowdSec bietet einen beeindruckenden Funktionsumfang. Mit den im Hub verfügbaren Parsern und Szenarios lassen sich gängige Applikationen mit wenig Aufwand integrieren. Durch das Anreichern der Logs mit Metadaten (etwa Geoinformationen) sowie der Möglichkeit, eigene Szenarios zu schreiben, lassen sich auch komplexere Anwendungsfälle wie progressiv längere Sperren oder Impossible Travel Detection umsetzen (siehe ix.de/z3bp). Für Testsysteme können die aus der CrowdSec-Community geteilten Entscheidungen über blockierte IP-Adressen sicherlich interessant sein. Für produktive Umgebungen sollte man sich jedoch gründlich überlegen, ob man die CrowdSec Central API anbinden möchte oder nicht. (avr@ix.de)

Quellen

Detaillierte Informationen zu CrowdSec und den im Artikel vorgestellten Komponenten: ix.de/z3bp

LUKAS GRIMM

ist Lead Systems Architect bei Puzzle ITC, verantwortet die interne IT-Infrastruktur, treibt die strategische Entwicklung voran, leitet Ansible-Schulungen und interessiert sich für Observability.

RETO KUPFERSCHMID

arbeitet als Systems Architect bei Puzzle ITC Schweiz und beschäftigt sich hauptsächlich mit Ansible Automatisierung und Observability-Themen rund um Prometheus.