SCALE COMMERCE

KI GEGEN BOTS

WIE AI IN DER BOT-ABWEHR WIRKLICH HILFT

WEBINAR TEIL 2

AGENDA

01

KI-Angriff in der Praxis

Live-Daten: ChatGPT-User Bot mit 139.582 Hits in 15 Minuten

02

Threat Intelligence

CrowdSec, Cloudflare Bot Management, BunnyCDN Bot-Shield

03

Anomalie-Erkennung

Fake vs. echte User-Agents: identische Strings, unterschiedliche Signale

04

LLM-Tools zur Log-Analyse

Claude, Mistral, Llama: Cloud-API und lokale Modelle mit Ollama

05

Datenschutz und DSGVO

Log-Daten und LLMs: Anonymisierung, lokale Verarbeitung, Compliance

RECAP: BOTS, SICHERHEIT & PERFORMANCE

Bot-Landschaft

42% des Webtraffics sind Bots. Davon sind 65% als "bad bots" klassifiziert, die scrapen, credential-stuffing betreiben oder DDoS-Attacken fahren.

Auswirkungen

Performance-Degradation, verzerrte Analytics, erhoehte Infrastrukturkosten, Datenverlust durch Scraping und Account Takeover.

Defense Layers

WAF, Rate Limiting, Geo-Blocking, Bot-Scores, JavaScript-Challenges. Mehrere Schichten sind Pflicht.

Heute gehen wir tiefer: Wie hilft KI konkret bei Erkennung und Abwehr?

01

KI-ANGRIFF IN DER PRAXIS

LIVE-DATEN: ChatGPT-User/1.0 BOT

139.582

Hits in 15 Minuten

~155

Requests pro Sekunde

1x IP

Einzelne Quell-Adresse

Was ist passiert?

User-Agent: ChatGPT-User/1.0
OpenAI-Bot crawlt aggressiv Produktseiten
Kein robots.txt Respekt trotz Eintrag
Origin-Server unter Volllast (CPU 98%)
Cache-Bypass durch dynamische URL-Parameter

Sofortmassnahmen

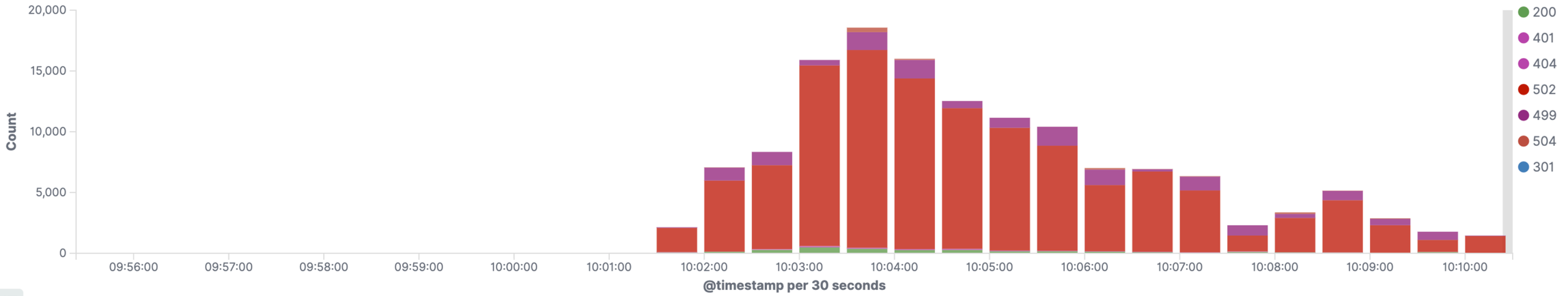
1. Proof-of-Work-Challenge auf User-Agent Basis
2. IP-Block der Quell-Adresse
3. CrowdSec Bouncer mit automatischem Ban
4. robots.txt + Crawl-Delay anpassen

Full screen Share Clone Edit

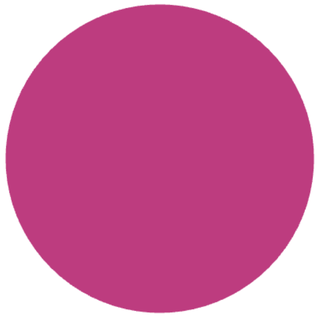
Filters 1 KQL Last 15 minutes Show dates Refresh

useragent: Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko); compatible; ChatGPT-User/1.0; +https://openai.com/bot + Add filter

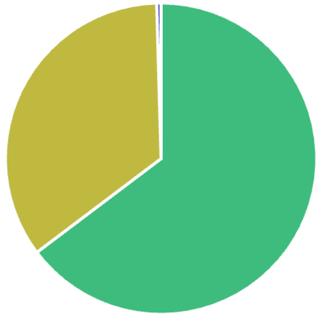
Status Code over time



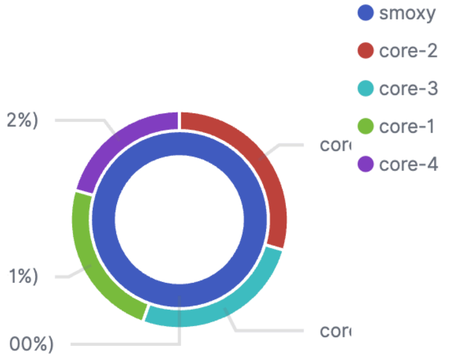
SC_Customer



Host



Sender



Hit Counter Sum

139,582
Hits

Percentiles total_time

0.003
75th percentile of time

2.002
90th percentile of time

Top IPs

ip: Descending	Count

Device

Content Encoding

Status Code

WARUM KI-BOTS ANDERS SIND

Klassische Bots

Einfache Skripte, vorhersehbare Muster
Statische User-Agents, kein JS-Rendering
Leicht durch Fingerprinting erkennbar

KI-gesteuerte Bots

Adaptives Verhalten, lernen aus Blocking
Simulieren echtes Browsing (Headless Chrome)
Rotieren IPs, User-Agents, TLS-Fingerprints

Neue Bedrohungskategorien

LLM-Crawler (GPTBot, ChatGPT-User, Claude-Web, Bytespider)
Trainieren ihre Modelle auf euren Content ohne Erlaubnis
AI-Scraper die Produktdaten für Konkurrenz-Analysen extrahieren
Automated Account Creation mit KI-generierten Profildaten
Credential Stuffing mit KI-optimierten Timing-Patterns

02

THREAT INTELLIGENCE

CROWDSEC: COMMUNITY-DRIVEN THREAT INTELLIGENCE

25M+	12M	190+	85%
Malicious IPs	Signale/Tag	Länder	Precision Rate

Community Blocklist

Crowdsourced IP-Reputation aus 70.000+ Installationen
Echtzeit-Updates wenn neue Angreifer erkannt werden
Integration via Bouncer (nginx, iptables, Cloudflare)
Szenarien: Brute-Force, Scan, DDoS, Credential Stuffing
Open Source, kostenlose Basisfunktion

Threat Forecast (AI/ML)

ML-basierte Vorhersage neuer Angriffsquellen
Predictive Blocking bevor der Angriff startet
Korrelation von Mustern über Installationen hinweg
Blocklist-as-a-Service für Enterprise
API-Integration in bestehende SIEM/SOAR Pipelines

BUNNYCDN BOT-SHIELD & EDGE RULES

Bot-Shield

Automatische Bot-Erkennung an der Edge
JS-Challenges für verdächtige Clients
Reputationsbasiertes Scoring
Aktivierung per Toggle im Dashboard

Edge Rules

Benutzerdefinierte Regeln auf CDN-Ebene
Trigger: User-Agent, IP, Country, URL-Pattern
Actions: Block, Redirect, Header-Modify
Kein Origin-Request nötig bei Block

Rate Limiting

Requests pro Zeitfenster pro IP limitieren
Konfiguriurbar pro Pull Zone
Burst-Toleranz einstellbar
Schont Origin-Server effektiv

BunnyCDN eignet sich besonders als vorgelagerte Schicht für Shops (Shopware, Magento). Standardmäßig Managed von smoxy/ ScaleComnmerce Kunden, die CDN Funktion aktiviert haben.

CLOUDFLARE BOT MANAGEMENT

ML-basierte Bot Scores

Score 1-99 pro Request (1 = definitiv Bot)
5 ML-Modelle parallel: Fingerprint, Verhaltensanalyse, Anomalie-Detection, IP-Reputation, Challenge-Ergebnisse
Trainiert auf 60M+ Requests/Sekunde Netzwerkdaten

Super Bot Fight Mode

Automatische Bot-Kategorisierung
Verified Bots (Google, Bing) werden durchgelassen
Konfigurierbare Actions: Block, Challenge, Log
JavaScript-Detection und Turnstile CAPTCHA

Praxis-Empfehlung

```
Firewall Rule: Bot Score < 30 AND NOT cf.bot_management.verified_bot
```

Action: Managed Challenge. Logging aktivieren für Tuning.
Tipp: Erst im Log-Modus testen, dann schrittweise verschärfen. False Positives bei APIs und Monitoring prüfen.

03

ANOMALIE-ERKENNUNG

WELCHER USER-AGENT IST FAKE?

Beide senden den identischen User-Agent String:

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 Chrome/120.0.0.0 Safari/537.36
```

Client A

`TLS: JA3 = a0e9f5d64349...`
`Accept-Language: de-DE,de;q=0.9`
HTTP/2 mit korrektem SETTINGS Frame
JavaScript: ausgeführt, Canvas Fingerprint konsistent
Session: 12 Seiten in 4 Minuten, organische Navigation
Cookies: akzeptiert und zurückgesendet

Client B

`TLS: JA3 = 72a589fd5...`
`Accept-Language: fehlt komplett`
HTTP/1.1, kein SETTINGS Frame
JavaScript: nicht ausgeführt, kein Canvas
Session: 200 Seiten in 30 Sek, sequentiell
Cookies: ignoriert, keine zurückgesendet

AUFLÖSUNG: CLIENT B IST DER BOT

Der User-Agent allein reicht ***nicht*** zur Erkennung. Diese Signale entlarven Bots:

TLS Fingerprint (JA3/JA4)

Jeder TLS-Client hat ein einzigartiges Handshake-Profil. Chrome-Bot mit Python-TLS-Stack: sofort erkennbar.

HTTP-Protokoll Verhalten

Echte Browser nutzen HTTP/2 mit korrekten SETTINGS Frames. Bots fallen auf HTTP/1.1 oder fehlerhafte H2 zurück.

Accept-Language Header

Echte Browser senden immer Accept-Language. Fehlt er komplett: starkes Bot-Signal.

JavaScript Execution

Canvas-Fingerprint, WebGL-Renderer, Font-Enumeration. Headless Chrome ohne Patches fällt hier durch.

Session-Verhalten

Request-Timing, Navigation-Muster, Mouse-Events, Scroll-Depth. 200 Seiten in 30 Sek = kein Mensch.

Cookie-Handling

Bots ignorieren oft Set-Cookie oder senden keine zurück. Session-Cookies sind ein einfacher Litmus-Test.

04

LLM-TOOLS ZUR LOG-ANALYSE

CLAUDE: API-BASIERTE LOG-ANALYSE

Warum Claude für Log-Analyse?

200k Token Context Window

Kann tausende Log-Zeilen in einem Request analysieren
Erkennt Anomalien, gruppiert Angriffsmuster
Generiert Firewall-Rules aus erkannten Mustern
Strukturierte Outputs (JSON) für Automation

Kosten (Stand 2026)

Sonnet: ~$3/1M Input Token
~10.000 Log-Zeilen = ~15k Tokens
Kosten pro Analyse: ~$0.05
Für Ad-hoc-Analyse: sehr kosteneffizient

Beispiel-Prompt für Log-Analyse

```
# Analysiere diese nginx Access-Logs:
# Finde: Bot-Traffic, Anomalien, verdächtige Patterns
cat access.log | \
  claude -p "Analysiere diese Logs. \
  Gruppiere nach Angriffsmuster. \
  Generiere smoxy-Contional—Rules als Output."
```

LOKALE LLMs: OLLAMA + MISTRAL/LLAMA

Warum lokal?

DSGVO-konform: Daten verlassen den Rechner/Server nicht
Keine API-Kosten, beliebig skalierter
Offline nutzbar, auch in Air-Gapped Environments
Volle Kontrolle über das Modell und die Daten
Latenz: Keine Netzwerk-Roundtrips

Modell-Empfehlung

Mistral 7B: Beste Balance Qualität/Speed
Llama 3 8B: Stärker bei komplexen Patterns
Codestral: Optimiert für Code/Log-Analyse
Hardware: 16GB RAM Minimum, GPU empfohlen
Ollama: 1-Befehl-Setup, REST API inklusive

Ollama Setup + Log-Analyse

```
ollama pull mistral
cat access.log | \
  sed 's/[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}/xxx.xxx.xxx.xxx/g' | \
  ollama run mistral \
  "Finde Bot-Patterns und Anomalien in diesen Logs"
```

05

DATENSCHUTZ UND DSGVO

DSGVO-KONFORM MIT LLMs ARBEITEN

Das Problem

Log-Daten enthalten personenbezogene Daten
IP-Adressen sind PII (Personally Identifiable Information) nach DSGVO Art. 4
User-Agents können Fingerprinting ermöglichen
Cloud-APIs: Daten verlassen die EU Aufbewahrungsfristen müssen eingehalten werden

Die Lösung

Anonymisierung VOR der LLM-Verarbeitung
IP-Adressen maskieren (sed/awk Pipeline)
PII-Detection mit diversen Tools.
Lokale Modelle: keine Datenübertragung
Data Processing Agreement bei Cloud-APIs

Anonymisierungs-Pipeline

`1. Rohdaten > 2. IP-Maskierung (sed) > 3. PII-Scan > 4. LLM-Input`

IPs durch Platzhalter ersetzen, E-Mails entfernen, Session-IDs hashen
Maskierung: Automatische Erkennung von Namen, Adressen, Telefonnummern in Freitext-Feldern
logredact: Spezialisiertes Tool für strukturierte Log-Formate (nginx, Apache, syslog)

CLOUD-API vs. LOKALES LLM

Kriterium	Cloud-API (Claude, GPT)	Lokal (Ollama + Mistral)
DSGVO	DPA erforderlich, Anonymisierung Pflicht	Voll konform, keine Datenübertragung
Qualität	Sehr hoch (Claude Sonnet/Opus)	Gut für Standard-Analysen
Context Window	200k+ Tokens	8k-32k Tokens (modellabhängig)
Kosten	Pay-per-Token (~$0.05/Analyse)	Hardware-Kosten (einmalig)
Latenz	Netzwerk-Roundtrip (1-5s)	Lokal, abhängig von GPU
Offline	Nein	Ja
Setup	API-Key, fertig	Ollama install, Modell pullen

Empfehlung: Lokal für Routine-Analyse, Cloud-API für komplexe Incident-Response.

KEY TAKEAWAYS

01 KI-Bots sind real und aggressiv. 139.582 Hits in 15 Minuten von einem einzelnen LLM-Crawler.

02 Threat Intelligence (CrowdSec, Cloudflare, BunnyCDN) als erste Verteidigungslinie automatisieren.

03 User-Agent allein reicht nicht. TLS-Fingerprint, JS-Execution und Session-Verhalten kombinieren.

04 LLMs beschleunigen Log-Analyse enorm. Claude für komplexe Cases, Ollama für DSGVO-konforme Routine.

05 Daten immer anonymisieren bevor sie in ein LLM fliessen. Lokale Modelle minimieren das Risiko.

FRAGEN?

Danke fürs Zuhören.

SCALE COMMERCE