



BBOT: Angriffsflächen automatisch erkennen und Risiken reduzieren

Das Open-Source-Tool BBOT deckt Schatten-IT auf, identifiziert IT-Assets präzise und erleichtert damit das Attack Surface Management.

Von Danny Scherer

■ Um IT-Infrastruktur wirksam vor Angriffen zu schützen, muss zunächst klar sein, aus welchen Komponenten sie besteht. Nur dann kann man wissen, was überhaupt geschützt werden soll. Jedes erreichbare System und jeder erreichbare Dienst kann potenziell als Einstiegspunkt für Angreifer dienen, um sich Zugriff zum Netzwerk zu verschaffen oder sich darin fortzubewegen. Die Gesamtheit dieser potenziellen Einstiegspunkte wird als Attack Surface bezeichnet.

Ohne detaillierte Kenntnisse über alle Komponenten bleibt die Sicherheit der gesamten IT-Landschaft unüberschaubar. Es besteht die Gefahr, dass eine sogenannte Schatten-IT aus nicht gewarteten oder verwaisten Diensten entsteht, in denen sich im Laufe der Zeit kritische Sicherheitslücken anhäufen können.

Sichtbarkeit schaffen

Das Aufspüren, Dokumentieren und Aktualisieren von Assets innerhalb der zu verwaltenden IT-Infrastruktur wird als Attack Surface Management (ASM) be-

zeichnet. Ziel ist, eine vollständige und aktuelle Übersicht aller potenziellen Angriffspunkte zu gewährleisten.

Die Identifikation von Assets kann manuell erfolgen, beispielsweise durch den Einsatz von Portscannern zur Erkennung erreichbarer Dienste, durch DNS-Aggregatoren zur Aufdeckung vorhandener Subdomains oder durch die Analyse von Konfigurationen. Die dabei gewonnenen Informationen werden händisch systematisch erfasst und dokumentiert.

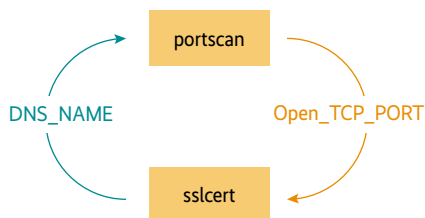
XX-TRACT

- Ohne klare Übersicht über alle IT-Assets wächst das Risiko durch unentdeckte Schatten-IT.
- ASM-Tools wie BBOT automatisieren die Erkennung und reduzieren so effektiv die Angriffsfläche.
- BBOT punktet als Open-Source-Tool mit Tiefe und Flexibilität, verlangt aber Einarbeitung.

In größeren IT-Umgebungen wäre ein rein manuelles ASM mit enormem Zeit- und Organisationsaufwand verbunden und damit kaum praktikabel, insbesondere wenn die Daten laufend aktuell gehalten werden müssen. In solchen Fällen empfiehlt sich der Einsatz spezialisierter ASM-Software, um automatisiert eine möglichst umfassende und kontinuierliche Sichtbarkeit aller IT-Assets zu gewährleisten.

Eine leistungsfähige ASM-Software sollte in der Lage sein, sämtliche IT-Assets zuverlässig über unterschiedliche Zugriffswege zu identifizieren, etwa über IP-Adressen, Ports, Domainnamen oder Subdomains. Idealerweise werden die Ergebnisse dieser einzelnen Identifikationsprozesse dynamisch miteinander verknüpft: Erkennt das System beispielsweise eine aktive IP-Adresse, sollte es automatisch prüfen, ob sich daraus über DNS-Lookups weitere Informationen wie zugehörige Domainnamen ableiten lassen.

Die ermittelten Informationen sollten sich zudem in maschinenlesbaren For-



Die rekursive Ereignisverarbeitung von BBOT führt zu einem sich selbst erweiternden Erkennungsprozess (Abb. 1).

maten exportieren lassen, um sie in nachgelagerte Sicherheitstools wie Schwachstellenscanner integrieren zu können. Auf diese Weise entsteht ein automatisierter, effizienter Workflow, der zur kontinuierlichen Reduktion der Angriffsfläche beiträgt.

Kommerzielle Anbieter wie CrowdStrike, Palo Alto Networks oder SentinelOne bieten leistungsfähige Software für das Attack Surface Management (ASM) an. Eine kosteneffizientere und oft flexiblere Alternative stellt jedoch der Einsatz passender Open-Source-Software dar. Besonders geeignet sind Automatisierungswerkzeuge aus dem Bereich der Open Source Intelligence (OSINT), da sie öffentlich zugängliche Informationen über extern erreichbare Systeme aus verschiedenen Quellen zusammentragen. Auf diese Weise lässt sich die IT-Landschaft sehr gut aus der Perspektive eines potenziellen Angreifers analysieren.

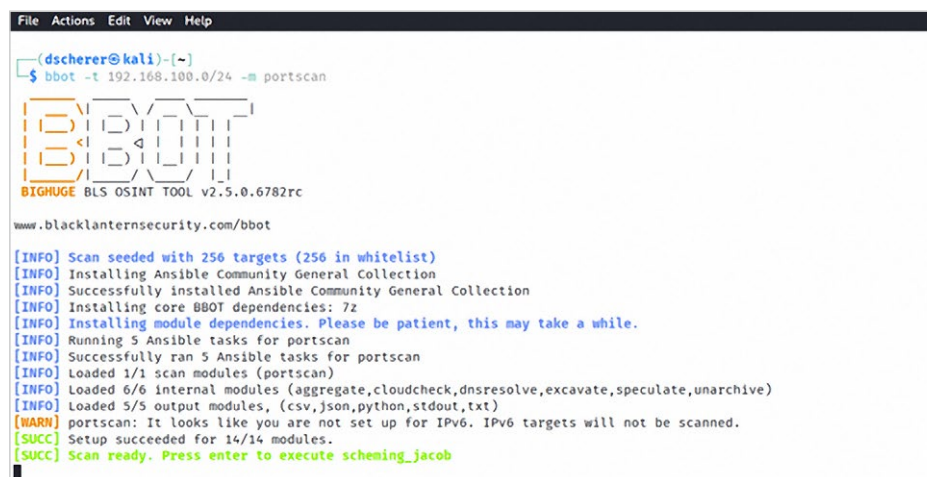
Ein Beispiel für eine solche Software ist das quelloffene BBOT (Bighuge BLS OSINT Tool). Es erfüllt die zentralen Anforderungen an eine effektive ASM-Software und überzeugt durch hohe Leistungsfähigkeit. BBOT orchestriert eine Vielzahl bewährter Open-Source-Tools wie Portscanner, Subdomain-Enumeratoren und Web-Fuzzer und kombiniert sie mit den öffentlich zugänglichen APIs von Diensten wie Shodan oder DNS-Dumpster. BBOT fügt deren Ergebnisse zu einem oder vielen Systemen in einem integrierten Workflow zusammen (rekursive Verknüpfung). Das ermöglicht eine umfassende Erkennung und strukturierte Dokumentation aller relevanten IT-Assets.

BBOT bietet eine deutlich bessere Scanleistung und -tiefe im Vergleich zu anderen Open-Source-Tools wie SpiderFoot oder Amass. Aufgrund der Architektur von BBOT findet das Werkzeug nicht nur umfangreichere Ergebnisse, sondern führt Scans auch spürbar schneller durch.

Die Verarbeitung der Scanergebnisse in BBOT erfolgt ereignisbasiert: Jedes Modul konsumiert bestimmte Ereignistypen und erzeugt daraus neue. So verarbeitet das Modul portscan Ereignisse des Typs DNS_NAME und erzeugt daraus OPEN_



Die Hilfeseite von BBOT lässt sich über die Kommandozeile aufrufen und bietet eine ausführliche Übersicht über die Einstellungsmöglichkeiten (Abb. 2).



BBOT informiert über das Set-up und ermöglicht das manuelle Starten des Scans (Abb. 3).

TCP_PORT-Ereignisse, sobald ein offener Port gefunden wurde. Das Modul sslcert hingegen konsumiert OPEN_TCP_PORT und generiert wiederum neue DNS_NAME-Ereignisse, sofern der übergebene Port zu einem HTTP-Server gehört, der TLS-Zertifikate (Transport Layer Security) ausliefert.

Führt das System beide Module gleichzeitig aus, entsteht eine rekursive Verknüpfung (siehe Abbildung 1), die eine besonders gründliche Erfassung der IT-Assets ermöglicht.

BBOT installieren

Die von BBOT generierte Inventarisierung der IT-Landschaft wird standardmäßig im Textformat, als JSON und als CSV gespeichert. Dadurch lassen sich die gewonnenen Daten unkompliziert wei-

terverarbeiten oder in andere Sicherheitstools integrieren.

BBOT wird vollständig in Python entwickelt und lässt sich unter Linux mithilfe von pipx, einem Hilfsprogramm zur Installation von Python-Modulen mit ausführbaren Programmen, in einer isolierten Python-Umgebung installieren:

```
pipx install bbot
```

Der Befehl bbot führt das Tool nach der Installation in der Kommandozeile aus. bbot --help zeigt beispielsweise die Hilfeseite an (siehe Abbildung 2).

Für Windows oder macOS steht ein Docker-Image zur Verfügung. Der folgende Befehl lädt das Image beim ersten Aufruf automatisch herunter und zeigt anschließend die Hilfeseite an:

```
docker run -it blacklanternsecurity/bbot:stable --help
```

Nach der Installation lässt sich ein erster einfacher Scan mit BBOT initiieren. Um beispielsweise einen Portscan aller Hosts im lokalen 192.168.100.0/24-Netz durchzuführen, genügt folgender Befehl:

```
bbot -t 192.168.100.0/24 -m portscan
```

Portscans mit BBOT

Der Parameter `-t` gibt das Scanziel und damit den Scope an. BBOT akzeptiert dabei verschiedene Zieltypen, darunter

Domains, IP-Adressen, IP-Ranges oder URLs. Um mehrere Ziele gleichzeitig zu scannen, werden sie in einer Textdatei aufgelistet und die Datei wird anschließend über `-t` übergeben.

Der Parameter `-m` definiert die zu verwendenden Module. Diese Module übernehmen die eigentliche Scan- und Analysearbeit. Neben dem Scan von Hosts, Ports und Subdomains bietet BBOT eine Vielzahl weiterer Module, unter anderem zur aktiven Suche nach Fehlkonfigura-

tionen, Sicherheitslücken oder anderen OSINT-relevanten Informationen. Damit geht BBOT über den klassischen Einsatz als ASM-Werkzeug hinaus und eignet sich auch für tiefer greifende Sicherheitsanalysen.

Bei der ersten Ausführung eines BBOT-Moduls installiert die Orchestrierungssoftware Ansible automatisch alle erforderlichen Tools auf dem Host, bevor der Scan startet. BBOT informiert die Nutzerinnen und Nutzer über das Set-up (siehe Abbildung 3). Nach Abschluss dieser Vorbereitung beginnt der Scan mit dem Drücken der Enter-Taste.

Während des Scans gibt BBOT fortlaufend den aktuellen Status sowie neu erkannte Ereignisse in der Konsole aus (siehe Abbildung 4). Nach Abschluss erscheint eine Zusammenfassung der Ergebnisse, inklusive der Pfade zu den gespeicherten Dateien (siehe Abbildung 5).

Die Scanergebnisse liegen standardmäßig in einem Unterordner im Verzeichnis `.bbot` im Homeverzeichnis des ausführenden Benutzers. Der Name des Unterordners entspricht dem jeweiligen zufällig von BBOT gewählten Scannamen, im Beispiel `scheming_jacob`.

Je nach Einsatzzweck lassen sich die verschiedenen Ausgabeformate aus dem Ergebnisordner flexibel weiterverarbeiten. Die CSV-Datei eignet sich beispielsweise zur Inventarisierung: Der Import in eine Tabellenkalkulationssoftware ermöglicht das Sortieren oder Filtern der Daten nach Eventtypen (zum Beispiel `IP_ADDRESS`, `OPEN_TCP_PORT`, `DNS_NAME`) oder IP-Adressen (siehe Abbildung 6).

BBOT findet während des Portscans einige Systeme im Zielnetz (Abb. 4).

```
File Actions Edit View Help
[SUCCESS] Scan ready. Press enter to execute scheming_jacob

[SUCCESS] Starting scan scheming_jacob
[SCAN]
[SCAN] scheming_jacob (SCAN:58d7db607fdefa7a37dd9752444ff1ff57cd060) TARGET
[IP_RANGE] 192.168.100.0/24 TARGET (in-scope, ipv4, private-ip, target)
[DNS_NAME] dev-machine-02.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] web-server-01.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] musings_dhawan.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] db-server-mysql.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] dev-machine-01.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] vpn-gateway.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] mail-server.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] iot-device.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] file-server-samba.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] db-server-postgres.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] monitoring-server.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] printer-server.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] web-server-02.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] dns-server.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[DNS_NAME] ad-controller.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[IP_ADDRESS] 192.168.100.15 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.12 A (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.8 A (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.2 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.13 A (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.16 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.7 A (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.5 A (in-scope, ipv4, private-ip, ptr-record)
[DNS_NAME] backup-server.toy.net PTR (a-record, dns-whitelisted-a, domain, in-scope, private-ip)
[IP_ADDRESS] 192.168.100.13 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.16 A (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.17 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.11 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.3 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.17 A (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.14 speculate (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.4 A (in-scope, ipv4, private-ip, ptr-record)
[IP_ADDRESS] 192.168.100.14 A (in-scope, ipv4, private-ip, ptr-record)

[INFO] Finishing scan
[SCAN]
[INFO] aggregate: + scheming_jacob (SCAN:58d7db607fdefa7a37dd9752444ff1ff57cd060) TARGET
[INFO] aggregate: | Module | Produced | Consumed |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] aggregate: | portscan | 29 (29 OPEN_TCP_PORT) | 273 (16 DNS_NAME, 256 IP_ADDRESS, 1 IP_RANGE) |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] aggregate: | speculate | 17 (17 IP_ADDRESS) | 273 (16 DNS_NAME, 256 IP_ADDRESS, 1 IP_RANGE) |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] aggregate: | PTR | 16 (16 DNS_NAME) | 0 |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] aggregate: | A | 16 (16 IP_ADDRESS) | 0 |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] aggregate: | host | 11 (11 IP_ADDRESS) | 0 |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] aggregate: | dnsresolve | 0 | 345 (16 DNS_NAME, 300 IP_ADDRESS, 29 OPEN_TCP_PORT) |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] aggregate: | cloudcheck | 0 | 345 (16 DNS_NAME, 300 IP_ADDRESS, 29 OPEN_TCP_PORT) |
[INFO] aggregate: +-----+-----+-----+-----+
[INFO] output.csv: Saved CSV output to /root/.bbot/scans/scheming_jacob/output.csv
[INFO] output.json: Saved JSON output to /root/.bbot/scans/scheming_jacob/output.json
[INFO] output.txt: Saved TXT output to /root/.bbot/scans/scheming_jacob/output.txt
[SUCCESS] Scan scheming_jacob completed in 1 minute, 47 seconds with status FINISHED

(dscherer@kali)~$
```

Vom ersten Scan zur ASM-Lösung

Um BBOT für eine größere IT-Infrastruktur als effektive Attack-Surface-Management-Software einzusetzen, sollte der Scope des Scans klar definiert sein. Dabei gilt es, sowohl die Zielsysteme als auch die relevanten Zugriffsebenen genau festzulegen. Je nach Anwendungsfall können als Ziele beispielsweise IP-Adressen oder IP-Bereiche, aber auch Domains oder URLs infrage kommen. Sinnvolle Zugriffsebenen umfassen neben den Ports auch Softwareversionen, Domains, Subdomains und URLs.

Die Zusammenfassung zeigt, dass der Portscan von BBOT 29 offene Ports und 16 aktive Systeme identifizieren konnte (Abb. 5).

	A	B	C	D	E	F
1	Event type	Event data	IP Address	Source Module	Scope Distance	Event Tags
2	IP_RANGE	192.168.100.0/24		TARGET	0	in-scope,ipv4,private-ip,target
3	SCAN	{'id': 'SCAN:a1568974644b2172b1d11729960c9f5db81ff473', 'name': 'savvy_gabriel', 'target': 'si		TARGET	0	
4	OPEN_TCP_PORT	192.168.100.1:80	192.168.100.1	portscan	0	in-scope
5	OPEN_TCP_PORT	192.168.100.1:3389	192.168.100.1	portscan	0	in-scope
6	OPEN_TCP_PORT	192.168.100.1:22	192.168.100.1	portscan	0	in-scope
7	IP_ADDRESS	192.168.100.1	192.168.100.1	host	0	in-scope,ipv4,private-ip
8	DNS_NAME	web-server-01.toytnet	192.168.100.10 PTR		0	a-record,dns-whitelisted-a,domain,in-scope,private-ip
9	IP_ADDRESS	192.168.100.10	192.168.100.10 speculate		0	in-scope,ipv4,private-ip,ptr-record
10	OPEN_TCP_PORT	192.168.100.10:80	192.168.100.10 portscan		0	in-scope
11	OPEN_TCP_PORT	web-server-01.toytnet:80	192.168.100.10 portscan		0	in-scope
12	DNS_NAME	backup-server.toytnet	192.168.100.11 PTR		0	a-record,dns-whitelisted-a,domain,in-scope,private-ip
13	IP_ADDRESS	192.168.100.11	192.168.100.11 A		0	in-scope,ipv4,private-ip,ptr-record
14	OPEN_TCP_PORT	192.168.100.11:873	192.168.100.11 portscan		0	in-scope
15	OPEN_TCP_PORT	backup-server.toytnet:873	192.168.100.11 portscan		0	in-scope
16	DNS_NAME	dns-server.toytnet	192.168.100.12 PTR		0	a-record,dns-whitelisted-a,domain,in-scope,private-ip
17	IP_ADDRESS	192.168.100.12	192.168.100.12 speculate		0	in-scope,ipv4,private-ip,ptr-record
18	OPEN_TCP_PORT	192.168.100.12:10000	192.168.100.12 portscan		0	in-scope
19	OPEN_TCP_PORT	192.168.100.12:53	192.168.100.12 portscan		0	in-scope
20	OPEN_TCP_PORT	dns-server.toytnet:10000	192.168.100.12 portscan		0	in-scope
21	OPEN_TCP_PORT	dns-server.toytnet:53	192.168.100.12 portscan		0	in-scope
22	DNS_NAME	db-server-postgres.toytnet	192.168.100.13 PTR		0	a-record,dns-whitelisted-a,domain,in-scope,private-ip
23	IP_ADDRESS	192.168.100.13	192.168.100.13 A		0	in-scope,ipv4,private-ip,ptr-record
24	OPEN_TCP_PORT	192.168.100.13:5432	192.168.100.13 portscan		0	in-scope
25	OPEN_TCP_PORT	db-server-postgres.toytnet:5432	192.168.100.13 portscan		0	in-scope
26	DNS_NAME	ad-controller.toytnet	192.168.100.14 PTR		0	a-record,dns-whitelisted-a,domain,in-scope,private-ip
27	IP_ADDRESS	192.168.100.14	192.168.100.14 A		0	in-scope,ipv4,private-ip,ptr-record
28	OPEN_TCP_PORT	192.168.100.14:389	192.168.100.14 portscan		0	in-scope
29	OPEN_TCP_PORT	ad-controller.toytnet:389	192.168.100.14 portscan		0	in-scope
30	DNS_NAME	db-server-mysql.toytnet	192.168.100.15 PTR		0	a-record,dns-whitelisted-a,domain,in-scope,private-ip
31	IP_ADDRESS	192.168.100.15	192.168.100.15 speculate		0	in-scope,ipv4,private-ip,ptr-record
32	OPEN_TCP_PORT	192.168.100.15:3306	192.168.100.15 portscan		0	in-scope

Die Auswertung der CSV schafft eine Übersicht der Ergebnisse und bietet Filtermöglichkeiten (Abb. 6).

Die Auswahl der eingesetzten Module richtet sich an diesen Definitionen aus und bestimmt, welche Funktionen BBOT bei regelmäßigen Scans ausführt, um eine umfassende und zielgerichtete Erfassung der IT-Assets sicherzustellen. Für die Erfassung von Ports und Softwareversionen eignen sich insbesondere die Module portscan und fingerprintx.

Subdomains lassen sich entweder passiv über öffentlich verfügbare APIs oder aktiv durch Brute-Forcing mittels Wörterlisten identifizieren. BBOT unterstützt beide Methoden und bietet hierfür zahlreiche Module, beispielsweise das Modul dnsdumpster, das die API von dnsdumpster.com nutzt, um Subdomains einer Domain passiv zu ermitteln.

Überblick über die Konfiguration mit Presets

Für einen möglichst umfassenden Subdomain-Scan empfiehlt es sich, mehrere

dieser Module zu kombinieren. BBOT stellt hierfür sogenannte Presets zur Verfügung. Das sind YAML-Dateien, in denen sämtliche Scaneinstellungen gespeichert sind. Dadurch entfällt die Notwendigkeit, Module und Optionen manuell über die Kommandozeile anzugeben. BBOT kommt mit vordefinierten Presets, eigene lassen sich zusätzlich erstellen. Ein Scan unter Verwendung eines Presets erfolgt über den folgenden Befehl:

```
bbot -p <Name des Presets>
```

Zusätzlich lassen sich wie gewohnt weitere Optionen an den Befehl anhängen, etwa die Scanziele mit -t <Ziele>. BBOT kombiniert die Einstellungen aus dem Preset mit den manuell übergebenen Parametern.

Für Subdomain-Scans eignet sich besonders das Preset subdomain-enum, das standardmäßig sowohl passive als auch aktive Module verwendet. Steht ausschließlich das passive Scannen im Fokus,

etwa um keine direkte Verbindung zum Ziel aufzubauen, lässt sich der Scan entsprechend einschränken. Mit dem Parameter -rf passive (Require Flag) werden nur passive Module ausgewählt und im Scan verwendet:

```
bbot -p subdomain-enum -rf passive -t heise.de
```

Dieser Befehl identifiziert Subdomains der Domain heise.de über öffentlich zugängliche Datenquellen (siehe Abbildung 7), ohne dabei aktiv mit den Servern der Zieldomain zu kommunizieren. Eine vollständige Übersicht über Flags und Filtermöglichkeiten bietet die offizielle BBOT-Dokumentation (alle Links zum Artikel unter ix.de/z7j4).

Schon der passive Subdomain-Scan liefert beeindruckende Ergebnisse: Das Beispiel zeigt 220 identifizierte Subdomains der Domain heise.de. Dabei kommt BBOT ganz ohne die aktive Kommunikation mit dem Zielsystem aus.

```
File Actions Edit View Help
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | urlscan | 6 (6 DNS_NAME) | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | cloud_amazon | 2 (2 STORAGE_BUCKET) | 0 |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | azure_tenant | 1 (1 AZURE_TENANT) | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | dnscaa | 1 (1 EMAIL_ADDRESS) | 220 (220 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | dnsresolve | 0 | 7,847 (3,370 DNS_NAME, 14 EMAIL_ADDRESS, 11 FINDING, 4,138 IP_ADDRESS, 203 OPEN_TCP_PORT, 2 STORAGE_BUCKET, 107 TECHNOLOGY, 2 URL_UNVERIFIED) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | cloudcheck | 0 | 4,162 (1,126 DNS_NAME, 2,126 DNS_NAME_UNRESOLVED, 1 EMAIL_ADDRESS, 11 FINDING, 584 IP_ADDRESS, 203 OPEN_TCP_PORT, 2 STORAGE_BUCKET, 107 TECHNOLOGY, 2 URL_UNVERIFIED) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | azure_realm | 0 | 220 (220 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | dnsbimi | 0 | 21 (21 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | dnstlsrpt | 0 | 21 (21 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | anubisdb | 0 | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | digitorus | 0 | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | dnsdumpster | 0 | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | leakix | 0 | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | myssl | 0 | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | subdomaincenter | 0 | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | wayback | 0 | 1 (1 DNS_NAME) |
[INFO] aggregate: +-----+-----+
[INFO] aggregate: | social | 0 | 2 (2 URL_UNVERIFIED) |
[INFO] aggregate: +-----+-----+
[INFO] output.csv: Saved CSV output to /home/dscherer/.bbot/scans/ingenious_terra/output.csv
[INFO] output.json: Saved JSON output to /home/dscherer/.bbot/scans/ingenious_terra/output.json
[INFO] output.subdomains: Saved 220 subdomains to /home/dscherer/.bbot/scans/ingenious_terra/subdomains.txt
[INFO] output.txt: Saved TXT output to /home/dscherer/.bbot/scans/ingenious_terra/output.txt
[SUCC] Scan ingenious_terra completed in 2 minutes, 7 seconds with status FINISHED
```

Der Subdomain-Scan hat 213 Subdomains von heise.de identifiziert (Abb. 7).

Um BBOT optimal für einen Scan im Rahmen des Attack Surface Managements zu konfigurieren, empfiehlt es sich, ein eigenes Preset zu erstellen. Das Listing zeigt, wie das aussehen könnte.

BBOT für ASM konfigurieren

Der Schlüssel includes im Preset dient dazu, bereits vorhandene Presets zu importieren. So verwendet dieses Beispiel das zuvor vorgestellte Preset sudomain-enum. Das erstellte Preset lässt sich unter einem passenden Namen wie asm-scan.yaml speichern.

Der folgende Befehl startet BBOT mit dem erstellten Preset:

```
bbot -p ./asm-scan.yaml
```

Für eine kontinuierliche Überwachung der Attack Surface sollte die ASM-Software die Ergebnisse jedes vorherigen Scans um potenziell neue Daten erweitern. Damit BBOT nicht für jeden ausgeführten Scan einen neuen Ergebnisordner anlegt, braucht das zuvor erstellte Preset folgenden Block:

```
config:
name: asm_scan
```

Ein fester Scannamen sorgt dafür, dass BBOT neue Scanergebnisse an bestehende Daten anhängt. Gleichzeitig nutzt das Tool frühere Ergebnisse, um neue Scans

durchzuführen. Dadurch beschleunigt sich der Scanvorgang deutlich, weil BBOT bereits identifizierte Ereignisse nicht erneut ermittelt.

Automatische Scans einrichten

Um die kontinuierliche Pflege des IT-Asset-Inventars im Rahmen des Attack Surface Managements sicherzustellen, empfiehlt es sich, den eingerichteten BBOT-Scan regelmäßig, beispielsweise einmal täglich, einzuplanen. Auf Linux-Systemen übernimmt Cronjob diese Aufgabe. Mit dem Befehl crontab -e öffnen die Nutzenden den persönlichen Crontab-Editor. Um den BBOT-Scan täglich um 03:00 Uhr auszuführen, braucht es die folgende Zeile Code:

```
0 3 * * * bbot -p <Pfad zur Preset.yaml> -y
```

Der Parameter -y kommt zum Einsatz, um die Bestätigung des Scanstarts durch Drücken der Enter-Taste zu überspringen. Wichtig hierbei: Nutzerinnen und Nutzer sollten den definierten BBOT-Scan vor der automatischen Ausführung mindestens einmal manuell starten. Auf diese Weise lassen sich die für die ausgewählten Module erforderlichen Ansible-Orchestrierungen auf dem Host korrekt ausführen. Zudem benötigen einige Module, wie portscan, Root-Rechte. Das gilt

Listing: ASM Preset

```
description: ASM Scan
target:
- 192.168.0.0/24
- 192.168.10.0/24
- 10.0.1.0/16
- example.com
- foobar.de
includes:
- subdomain-enum
modules:
- portscan
- fingerprintx
```

es bei der Wahl des ausführenden Benutzers zu berücksichtigen.

Stärken und Schwächen

BBOT muss sich neben kommerzieller Attack-Surface-Management-Software keinesfalls verstecken. Es nutzt zahlreiche nicht bewährte quelloffene Tools und orchestriert diese rekursiv, um eine möglichst hohe Scantiefe zu erreichen. Zwar verzichtet BBOT auf eine grafische Benutzeroberfläche und wird vollständig über das Terminal bedient, doch das wird durch die starke Performance und hohe Flexibilität von BBOT ausgeglichen.

Neben der hohen Anpassungsfähigkeit an die Besonderheiten der zu verwaltenen IT-Landschaft überzeugen die Scannergebnisse durch ihren offenen Datenexport, der eine einfache Weiterverarbeitung ermöglicht.

Die Flexibilität von BBOT erfordert jedoch eine intensivere Einarbeitung und sorgfältige Konfiguration. Im Gegensatz zu kommerziellen Lösungen mit benutzerfreundlicheren Oberflächen bereitet BBOT Scannergebnisse nicht automatisch auf und interpretiert sie nicht von selbst. Für die Analyse und Weiterverarbeitung der Daten braucht es ergänzende Tools oder eigene Auswertungsprozesse. Wertvoll ist jedoch die Quelloffenheit, die die Entwicklung eigener Module und die nahtlose Integration individueller Scan-Tools ermöglicht. (mdo@ix.de)

Quellen

Alle Links zum Artikel finden sich unter ix.de/z7j4.

DANNY SCHERER



ist IT-Sicherheitsberater bei der secuvera GmbH und spezialisiert auf Sicherheitsanalysen von Systemen sowie Penetrationstests für Webanwendungen und APIs.

