Neue Spielzeuge für den Bluthund

Um die Angriffsmöglichkeiten auf ein Active Directory auszuloten, ist BloodHound eines der populärsten und nützlichsten Werkzeuge für Verteidiger – und für Angreifer. Rund um das Tool hat sich ein ganzes Ökosystem mit ergänzenden Funktionen gebildet.

Von Philipp Löw

Seit seiner Veröffentlichung 2016 ist BloodHound eines der beliebtesten Werkzeuge für White-Hat- und Black-Hat-Hacker (und alle Nuancen dazwischen), die ein Active Directory (AD) näher untersuchen wollen. Die Darstellung der Angriffspfade als Graphen war seinerzeit ein Novum und ist heute noch das wichtigste Anwendungsmerkmal – auch wenn inzwischen andere Open-Source-Projekte wie Adalanche Ähnliches leisten (dieses und alle nachfolgend besprochenen Tools, Projekte und Quellen sind über ix.de/z539 zu finden). Früher mussten sich Angreifer mühsam durch Listen klicken und dort nach Verbindungen zwischen Objekten und Systemen suchen. Heute reicht meist ein Blick, um einen lohnenden Angriffsvektor zu identifizieren und im zweiten Schritt auszunutzen.

Seit August 2023 gibt es neben der originalen Version von BloodHound, inzwischen als BloodHound Legacy bezeichnet, auch die BloodHound Community Edition (CE). Sie unterscheidet sich in Betrieb und Funktionsumfang erheblich von der Originalversion. Auch sind nicht alle in diesem Artikel vorgestellten Erweiterungen kompatibel mit der Community Edition.

Grundsätzlich besteht BloodHound aus drei Komponenten: einem Datensammler (dem Collector SharpHound), der in der jeweiligen Domäne ausgeführt wird und relevante Daten über Windows-API- und LDAP-Funktionen sammelt, einer Graphdatenbank (Neo4j), in die diese Daten in der Folge geladen werden, und schließlich der BloodHound-Benutzeroberfläche, in der sie betrachtet werden. Einige frühere iX-Artikel [1, 2] erklären das Installieren, die Begrifflichkeiten und die Basisfunktionen der Komponenten. Der Blogartikel „The Ultimate Guide for BloodHound Community Edition (BHCE)“ von Chris Haller beschreibt zudem die BloodHound Community Edition in aller Tiefe (siehe ix.de/z539).

iX-TRACT

- BloodHound, das Werkzeug zur Angriffspfadanalyse, ist für jedes Red Team unverzichtbar geworden. Obwohl das Tool schon sehr leistungsfähig ist, erreicht es sein volles Potenzial erst durch die Integration einer Vielzahl von Erweiterungen.
- Verteidiger und Angreifer profitieren von den Add-ons, die eine aktive Gemeinschaft entwickelt. Dadurch kann der Bluthund fast jeden Anwendungsfall abdecken.
- Je nach Erweiterung werden die Daten in einem Netzwerk auf unterschiedliche Weise gesammelt, bereits vorhandene Daten neu aufbereitet oder durch weitere Informationen einer Domäne angereichert, beispielsweise um Passwort-Hashes.

Viele Helfer füllen die Lücken

Dennoch arbeitet BloodHound nicht immer so wie von vielen Nutzern gewünscht und lässt trotz seines Umfangs noch sinnvolle Funktionen vermissen. Glücklicherweise hat sich um den Spürhund mittlerweile eine große Züchtergemeinschaft entwickelt, die viel Zeit und Energie investiert, um BloodHound neue Tricks beizubringen oder Fehlverhalten zu beseitigen.

Die Erweiterungen lassen sich grob entsprechend ihrem Einsatzzweck unterteilen. Einige, wie Certipy, bloodhound-python, SilentHound oder ADExplorerSnapshot.py, bauen Funktionen zum Sammeln von Daten im untersuchten Active Directory aus. Andere, wie PlumHound, bloodhound-quickwin, AD-Miner oder GoodHound, konzentrieren sich auf die Auswertung der gesammelten Daten.

Sowohl das Sammeln als auch das Auswerten mithilfe der Erweiterungen kann neues Licht auf das Netzwerk werfen und bislang unentdeckte Pfade aufzeigen, die sich für Angreifer und Verteidiger lohnen. Für Betreiber mehrerer BloodHound-Projekte empfiehlt sich das Werkzeug bloodhoundcli; mit der Containerverwaltung podman verwaltet es

mehrere Neo4j-Datenbanken auf demselben System. Zudem installiert die bloodhoundcli BloodHound gleich dazu. In neueren Versionen bringt sie auch BHCE und die zusätzlich benötigte PostgreSQL mit; für BloodHound Legacy genügt Version 0.1.3.

Der stete Strom an veröffentlichten Erweiterungen ist auch ein Zeugnis für die schnelllebige Welt der IT. Besonders in der Informationssicherheit vergehen nach Bekanntwerden einer Schwachstelle oft nur wenige Tage, bis sie ausgenutzt wird. Deswegen sollte Verteidigern wie auch Pentestern stets daran gelegen sein, relevante Erweiterungen von BloodHound in ihren Arbeitsalltag einzubauen.

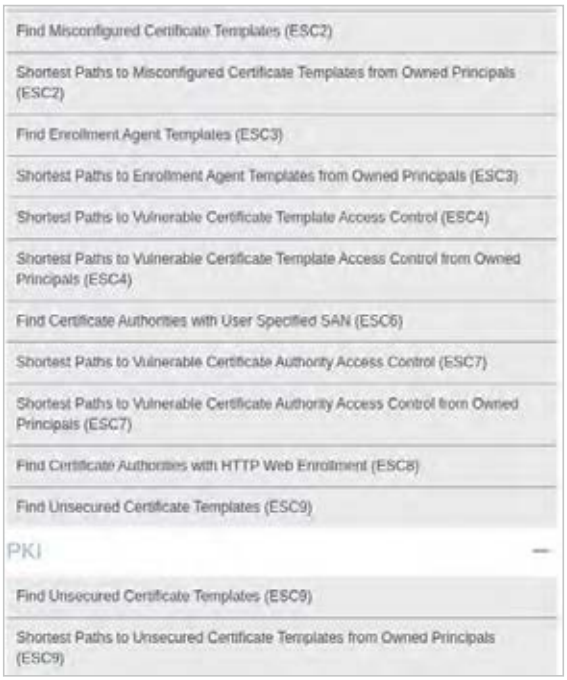
Analyse der Active-Directory-Zertifikatsdienste

Certipy ist ein Angriffswerkzeug, das sich auf die Active-Directory-Zertifikatsdienste (ADCS) spezialisiert hat. Es ist ein Ableger des originalen ADCS-Angriffswerkzeugs Certify und ermöglicht ADCS-Angriffe aus einer Linux-Umgebung heraus. Seit Version 2.0 ist es in BloodHound integrierbar.

Wie im 2021 veröffentlichten Whitepaper „Certified Pre-Owned“ von Will Schroeder und Lee Christensen beschrieben (siehe ix.de/z539), entfalten Angriffe auf die Public-Key-Infrastruktur eines AD in vielen Fällen eine verheerende Wirkung [3]. Das liegt daran, dass sich die Schwachstellen vergleichsweise einfach ausnutzen lassen, sobald die Angreifer Zugang zum internen Netzwerk haben; sie können ihre Privilegien meist bis auf jene eines Domänen- oder Organisationsadministrators erhöhen. Das kommt einer Kompromittierung des gesamten Forests gleich.

Dass der Bluthund solche wichtigen Angriffsvektoren enthalten muss, erkannten die zuständigen Entwickler schnell und statteten Certipy mit Parametern aus, die die Ergebnisse wahlweise für eine modifizierte BloodHound-Version oder BloodHound Legacy aufbereiten.

Mit benutzerdefinierten Abfragen kann man verschiedene Angriffsvarianten auf die Zertifikatsdienste überprüfen (Abb. 1).



Wer Certipy direkt ausführbar in einer eigenen virtuellen Python-Umgebung nutzen will, installiert es mit pipx:

```
$ pipx install certipy-ad
```

Zu beachten ist der Paketname `certipy-ad`; es gibt nämlich noch ein damit nicht verwandtes Python-Paket `certipy`, ein einfaches Werkzeug zum Erzeugen von Zertifikaten.

Die Befehle werden ähnlich ausgeführt wie im originalen Certify. Einen Überblick über die verwendeten Zertifikate, Zertifizierungsstellen und zusätzliche Konfigurationen fragt man wie folgt ab:

```
$ certipy find -bloodhound -dc-ip ↵
  192.168.56.10 -u cersei.lannister@↵
  sevenkingdoms.local -p 'il0vejaime'
```

Durch benutzerdefinierte Abfragen (Custom Queries) lässt sich BloodHound Legacy zudem so erweitern, dass es ADCS-Angriffspfade berücksichtigt. Dazu können Abfragen aus dem Certipy-Projekt verwendet und unter /$HOME/.config/bloodhound als JSON-Datei abgelegt werden. Nach einem Neustart von BloodHound sind die ADCS-Abfragen verfügbar, wie in Abbildung 1 dargestellt.

Wie man ESC1 überprüft

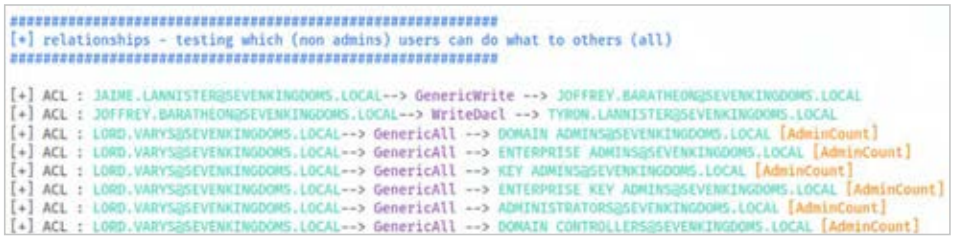
Als Beispiel dient die Rechteausweitung ESC1 (Escalation Attack 1). Das Whitepaper von Schroeder und Christensen definiert folgende Voraussetzungen, damit ESC1 funktioniert: Die Enterprise CA muss Benutzern mit niedrigen Privilegien Anmelderechte und das Recht zur Registrierung eines Zertifikats gewähren. Des Weiteren muss die Genehmigung durch einen Manager deaktiviert und autorisierte Signaturen dürfen nicht erforderlich sein. Die Zertifikatsvorlage muss außerdem `Extended Key Usages` (EKUs) definieren, die eine Authentifizierung ermöglichen. Schließlich müssen Antragsteller einen `subjectAltName` angeben dürfen. Die Attribute der Zertifikatsvorlagen sind zum einen in deren Knoten in BloodHound zu finden. Zum andern kann man vordefinierte oder eigene Abfragen verwenden, um verwundbare Vorlagen zu identifizieren.

Im Gegensatz zu den Datensammlern der ursprünglichen BloodHound-Version erweitern die Datensammler von BloodHound CE die Funktion zum Abrufen von Daten der Active Directory Certificate Services von Haus aus. Aktuelle Varianten des Collectors SharpHound sammeln die benötigten Informationen – die Community Edition von BloodHound zeigt sie als eigene Knoten an; dafür wurde der neue Typ CertTemplate definiert. Auch darüber findet man die Fehlkonfigurationen in den Details des Knotenobjektes.

In der Community Edition sind die mitgelieferten Abfragen etwas versteckt erst nach Klick auf die Registerkarte Cypher und anschließend auf das Ordnersymbol abrufbar. Im Bereich „Pre-built Searches“ findet die Abfrage „Enrollment rights on published ESC1 certificate templates“ Zertifikatsvorlagen mit der oben genannten Konfiguration.

Jene Knoten, die das Enroll-Recht besitzen, werden nun in der BloodHound-Oberfläche dargestellt (Abbildung 2).

Domänenbenutzer mit der Enroll-Kante zu verwundbaren Zertifikatsvorlagen: BloodHound offenbart diesen Angriffsvektor auch in der Community Edition (Abb. 2).



bloodhound-quickwin gibt die Beziehungen aus, die die Benutzer untereinander haben (Abb. 3).

Liegt dieses Recht beispielsweise bei der Gruppe der Domänenbenutzer, kann sie dadurch die zugewiesenen Rechte beliebig erweitern und sich selbst administrative Privilegien in der Domäne geben.

Für die Datensammler der Legacy-Versionen BloodHound 4.2 und 4.3 sowie die BloodHound Community Edition existieren mittlerweile Umsetzungen in Python, bekannt unter dem Namen BloodHound.py. Sie basieren allesamt auf dem „Schweizer Taschenmesser" eines jeden Red Teams: der Impacket-Werkzeugsammlung.

Damit kann ein Angreifer die Daten für BloodHound direkt von einem Linux-System aus dem AD auslesen – gültige Anmeldedaten für die entsprechende Domäne vorausgesetzt. Standardmäßig fragt BloodHound.py die Domänencontroller über LDAP und die erreichbaren Computer der Domäne ab, um Benutzer, Computer, Gruppen, Vertrauensstellungen, Sitzungen und lokale Administratoren aufzuspüren. Jedoch kann man, ähnlich zum klassischen SharpHound, über zusätzliche Parameter andere Eigenschaften abfragen. Lediglich lokale Gruppenmitgliedschaften, die über Gruppenrichtlinien verwaltet werden, findet die Python-Variante nicht.

Diese Art des Datensammelns ist besonders dann interessant, wenn man nur über ein Linux-System mit der Domäne interagieren kann. Die Installation ist über pip beziehungsweise pipx möglich:

```
$ pipx install bloodhound
```

Der folgende Befehl sammelt anschließend Informationen über die Domäne, wie der bereits bekannte Collector SharpHound. Die Daten kann man dann in BloodHound Legacy Version 4.2 oder 4.3 auswerten.

```
$ bloodhound-python -u cersei.↵
   lannister@sevenkingdoms.local -p ↵
  'il0vejaime' -ns 192.168.56.10 -d ↵
 sevenkingdoms.local -c all --dns-tcp
```

Dabei kann entweder das Klartextpasswort oder ein NT-Hash (mit dem Argument `--hashes`) als Anmeldeinformation verwendet werden; `-ns` übergibt den Nameserver der Domäne. Mit `-c` wird festgelegt, welche Daten gesammelt werden. Zuletzt bewirkt `--dns-tcp`, dass für DNS-Abfragen TCP anstelle von UDP verwendet wird. Die Daten werden als JSON-Dateien auf dem Linux-System abgelegt und können anschließend importiert werden.

Zum Sammeln von Daten, die in der BloodHound Community Edition ausgewertet werden sollen, dient der Branch bloodhound-ce des GitHub-Repositorys von BloodHound.py.

Ein unauffälliger BloodHound? Schwierig, aber nicht unmöglich!

Bei einer Angriffssimulation besteht ein Ziel häufig darin, sich möglichst lange unerkannt im Netzwerk zu bewegen. Ein Sammler wie SharpHound eignet sich dafür nicht: Wenn nicht schon das Ausführen von SharpHound auf einem kompromittierten Rechner blockiert oder als bösartig gemeldet wird, fällt spätestens beim Datensammeln auf, dass ungewöhnlich viele Abfragen in vergleichsweise kurzer Zeit stattfinden. Zu diesem Zeitpunkt sollten insbesondere in gehärteten Umgebungen sämtliche Alarmlämpchen blinken.

Um einer Entdeckung zu entgehen, kann man das Werkzeug SilentHound von einem Linux-Rechner aus verwenden. SilentHound liest mit lediglich einer einzigen LDAP-Abfrage die Daten aus, schreibt sie in eine Cachedatei und verarbeitet sie anschließend weiter. Trotz der Namensgebung ist SilentHound ein unabhängiges Tool und ermöglicht keinen Import in BloodHound.

Ein Nachteil ist die beschränkte Ausgabe. SilentHound sammelt nur Informationen zu Domänenbenutzern und -gruppen (`-g`), Organisationseinheiten (`-n`) und Dienstprinzipalen (`--kerberoast`). Zudem existiert nur eine eingeschränkte Suche nach Schlüsselwörtern (`-k`); diese müssen direkt im Quelltext der Anwendung (silenthound.py, Zeile 121) übergeben werden. Dennoch können diese Informationen schon hilfreich sein, um einen ersten unauffälligen Überblick über die Domäne zu bekommen:

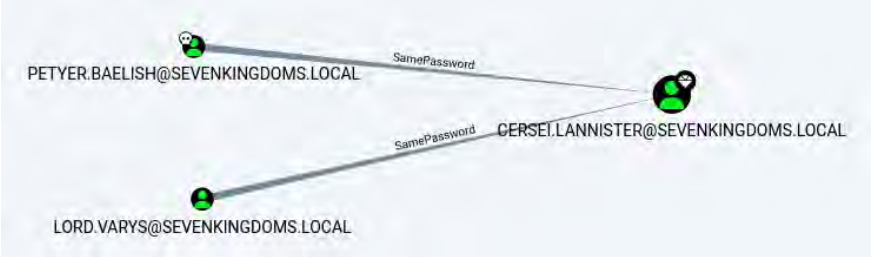
```
$ python3 silenthound.py -u cersei.↵
   lannister@sevenkingdoms.local -p ↵
          'il0vejaime' 192.168.56.10↵
          sevenkingdoms.local -g -n ↵
                      -k --kerberoast
```

Einen anderen Ansatz verfolgt ADExplorerSnapshot.py. Hier erstellt man zunächst mit dem von Microsoft signierten Werkzeug Active Directory Explorer aus der Sysinternals-Sammlung auf einem Windows-Rechner einen Schnappschuss der Domäne. Dazu genügen die Rechte eines normalen Benutzers, die viele Informationen in einem AD abfragen können. Der Snapshot wird auf ein System des Angreifers übertragen und mithilfe von ADExplorerSnapshot.py analysiert: Das Werkzeug wandelt die Daten aus dem Snapshot in JSON-Dateien um, die der Bluthund einlesen kann.

Dem Vorteil dieses vergleichsweise heimlichen Vorgehens stehen einige Einschränkungen gegenüber: Da es sich um einen Offlineschnappschuss von Daten aus einem Domänencontroller handelt,

Starting Node	Number of Enabled Non-Admins with Path	Percent of Total Enabled Non-Admins with Path	Number of Hops	Exploit Cost	Risk Score	Path
SMALL COUNCIL@SEVENKINGDOMS.LOCAL	5	55.6	1	0	55.6	SMALL COUNCIL@SEVENKINGDOMS.LOCAL - CanRDP -> KINGSLANDING.SEVENKINGDOMS.LOCAL
BARATHEON@SEVENKINGDOMS.LOCAL	3	33.3	1	0	33.3	BARATHEON@SEVENKINGDOMS.LOCAL - CanRDP -> KINGSLANDING.SEVENKINGDOMS.LOCAL

Die Pfadanalyse mit GoodHound enthüllt große Risiken, die in diesem Fall von nicht administrativen Benutzern ausgehen (Abb. 4).



Geteilte Passwörter werden mit hashcathelper als zusätzliche Kanten sichtbar (Abb. 5).

sind etwa keine Informationen zu laufenden Sitzungen oder lokalen Adminrechten verfügbar. Auch Gruppenrichtlinien und Organisationseinheiten fehlen.

Zur Einrichtung von ADExplorerSnapshot klont man das entsprechende GitHub-Repository und installiert wie dort beschrieben die notwendigen Abhängigkeiten, am besten in einer virtuellen Umgebung. Zudem muss in der __init__.py (`adexpsnapshot/__init__.py`) in Zeile 129 `self.preprocessCached()` durch `self.preprocess()` ersetzt werden, um die Kompatibilität zu einer verwendeten Abhängigkeit zu gewährleisten. Der folgende Befehl erstellt die JSON-Dateien für BloodHound aus einem AD-ExplorerSnapshot:

```
$ python3 ADExplorerSnapshot.py
snapshot.dat
```

Das Tool gibt zudem JSON-Dateien aus, die, falls vorhanden, ein ADCS abbilden. Diese Dateien können in die BloodHound-Legacy-Version eingelesen werden. Die unterschiedlichen Dateiformate der Datensammler in den BloodHound-Versionen verhindern jedoch die Analyse der Daten in der Community Edition.

Bevor man mit der Analyse in BloodHound beginnen kann, müssen die gesammelten Daten in die Graphdatenbank hochgeladen werden. Die JSON-Dateien werden eingelesen und in CREATE-Statements übersetzt, die schließlich die Knoten und Kanten in der Neo4j-Datenbank anlegen. Das wird von den meisten Anwendern zunächst einfach zur Kenntnis genommen. Der Ansatz funktioniert so lange, bis man vor den Daten eines AD sitzt, das mehrere Zehntausend Objekte enthält. Bei so großen Datensätzen stoßen die Datenbank und BloodHound schnell an ihre Grenzen.

Ein Blogartikel von Arris Huijgen (siehe ix.de/z539) erklärt, wie man große Datensätze in BloodHound verarbeitet. Aus seinen eigenen Herausforderungen im Umgang mit ausufernden Datenmengen hat er Skripte entwickelt, die auf GitHub verfügbar sind. Die Skripte chophound.ps1 und chophound.py lösen das Problem, indem sie die JSON-Dateien von SharpHound in kleinere Dateien aufteilen, die die Neo4j-Datenbank einfacher verarbeiten kann. Die Daten werden im Anschluss in der Datenbank wieder zusammengesetzt.

Listing: Benutzerdefinierte Abfrage für das Identifizieren geteilter Passwörter

```
MATCH p=((a:User)-[r:SamePassword*1..2]-(b:User))
WHERE ALL(x in r WHERE STARTNODE(x).objectid > ENDNODE(x).objectid)
AND ANY(c in [a,b] WHERE c.admincount OR c.name =~ '(?i)adm_.*')
RETURN p
```

Nachdem die fehlende Abhängigkeit ijson installiert wurde (mit pip oder via `apt install python3-ijson`), führt man die Python-Implementierung von chophound wie folgt aus:

```
$ python3 chophound.py
20240618082136_users.json -c 1000
```

Der Parameter `-c` gibt an, wie viele Objekte in einer der kleineren Dateien enthalten sein sollen. Je größer der Wert, desto weniger Dateien werden generiert.

Die Dateien können anschließend Stück für Stück in BloodHound Legacy 4.2 und 4.3 eingelesen werden. Man kann auch kleine JSON-Dateien für die Community Edition erstellen. Die Ausgabe ist jedoch zum Teil fehlerhaft: So werden beispielsweise keine Benutzernamen, sondern nur die SIDs (Security Identifier) der Objekte angezeigt.

Aus Graphen werden Listen

Massen an Informationen wirken rasch erschlagend, das bestätigt wahrscheinlich jeder, der schon einmal ein Active Directory mit mehr als fünf Benutzern in BloodHound analysieren durfte. Zwar sind alle Informationen in der Neo4j-Graphdatenbank enthalten, aber es ist mühselig, sie mithilfe einzelner Abfragen herauszuarbeiten.

Dieses Problem versuchen gleich zwei Erweiterungen anzugehen: PlumHound und bloodhound-quickwin. Deren Name ist Programm: Mit vordefinierten Regeln erstellen die beiden Werkzeuge umfassende Berichte mit den wichtigsten Erkenntnissen und Daten des AD. Auch diese Werkzeuge sollten in einer virtuellen Python-Umgebung betrieben werden, damit es zu keinen Problemen mit den Abhängigkeiten kommt. Sie nutzen die Neo4j-Datenbank, in der die Domäneninformationen gespeichert sind. Nachdem über pip die notwendigen Abhängigkeiten installiert worden sind, wird bloodhound-quickwin mit folgendem Befehl ausgeführt:

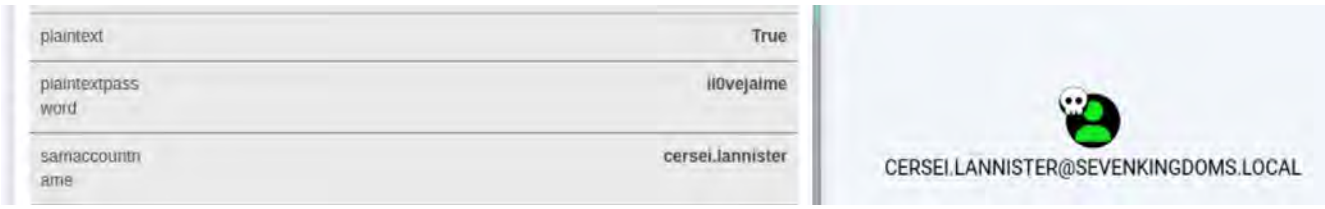
```
$ ./bhqc.py --heavy -u neo4j -p ↵
                                bloodhound
```

Die Ausgabe (siehe Abbildung 3) ähnelt der von SilentHound, ist aber umfangreicher, da bloodhound-quickwin Zugriff auf alle Daten hat, die SharpHound zuvor gesammelt hat. Besonders der verwendete Parameter `--heavy` ermöglicht die Darstellung verschiedener Beziehungen und Rechte, die über Access Control Lists (ACLs) [5] vergeben werden.

PlumHound wird mit folgendem Befehl ausgeführt:

```
$ python3 PlumHound.py -x tasks/↵
 default.tasks -u neo4j -p bloodhound
```

Die „Tasks“ in PlumHound bestehen aus einzelnen Abfragen in der Neo4j-Sprache Cypher. Die Aufgaben können je nach Verwendungszweck gewählt oder angepasst werden. PlumHound erstellt standardmäßig verschiedene CSV- und HTML-Dateien. Letztere kann man im Browser öffnen. So verschafft PlumHound sowohl Verteidigern als auch Angreifern schnell einen Überblick über die Domäne und hilft, interessante An-



In BloodHound angezeigte Klartextpasswörter visualisieren die Gefahr, die davon ausgeht (Abb. 6).

griffspfade zu identifizieren. Die Standardaufgaben zeigen zum Beispiel alle Benutzer und Computer mit der Zeichenkette „Pass“ oder „PW“ in ihrer Beschreibung – ein Hinweis auf ein Passwort. Die Abfragen sind natürlich beliebig erweiterbar.

Es geht nicht darum, einen komplexen Angriffspfad über viele verschiedene Knoten und falsch konfigurierte Berechtigungen zu finden, sondern darum, dass Angreifer beziehungsweise Verteidiger schnell verstehen, wie es um das Active Directory bestellt ist. Dieser Ansatz lohnt sich für Red Teams, da nach wie vor viele Umgebungen eklatante Sicherheitslücken aufweisen, die PlumHound oder bloodhound-quickwin schnell aufdecken. Solche Listen sind auch für Verteidiger interessant, weil sie einen schnellen Überblick über mögliche Schwachstellen im eigenen Netzwerk geben.

Wegen der unterschiedlichen Struktur der JSON-Dateien in den BloodHound-Versionen können bloodhound-quickwin und PlumHound nicht ohne manuelle Anpassungen mit der Community Edition verwendet werden.

Von unten oder von oben – warum nicht beides?

Auch AD-Recon – nicht zu verwechseln mit ADRecon, das Informationen direkt aus einer Domäne in CSV-Dateien oder Excel-Tabellen aufbereitet – erstellt Listen mit den Daten aus einer Neo4j-Datenbank. Der Mehrwert ergibt sich aus den beiden Analyseansätzen Top-down und Bottom-up, die das Werkzeug in sich vereint. AD-Recon funktioniert ebenfalls nicht mit der BloodHound Community Edition.

Nach dem Klonen des GitHub-Repos und optional dem Erstellen einer virtuellen Python-Umgebung installiert man die Abhängigkeiten wieder über pip:

```
$ pip install -r requirements.txt
$ pip install toml
```

Im Standardaufruf erstellt AD-Recon Listen, die denen von PlumHound ähneln.

```
$ python ad_recon.py -u neo4j -p ↵
                                bloodhound↵
```

Fügt man den Parameter `--transitive` an, nutzt die Anwendung den Bottom-up-Ansatz und analysiert die ausgehenden Kontrollbeziehungen einzelner Benutzer oder Computer. So findet man schnell Konten niedrig privilegierter Benutzer, die zu viele Berechtigungen haben und somit ein Risiko darstellen: Beispielsweise lassen sich darüber Benutzerkonten entdecken, die sich zu allen Rechnern per Remote Desktop verbinden können – einschließlich des Laptops des Geschäftsführers und der Domänenadmins. Bei größeren Umgebungen kann die Ausführung mit dem `transitive`-Parameter laut Entwickler bis zu fünf Stunden dauern.

Die Ausgaben lassen sich mit Linux-Befehlen sortieren, zum Beispiel nach den Benutzern mit den meisten transitiven ausgehenden Rechten in absteigender Reihenfolge:

```
$ cat users_outbound_trans_rights.↵
                    txt | sort -nr -k 7,7
```

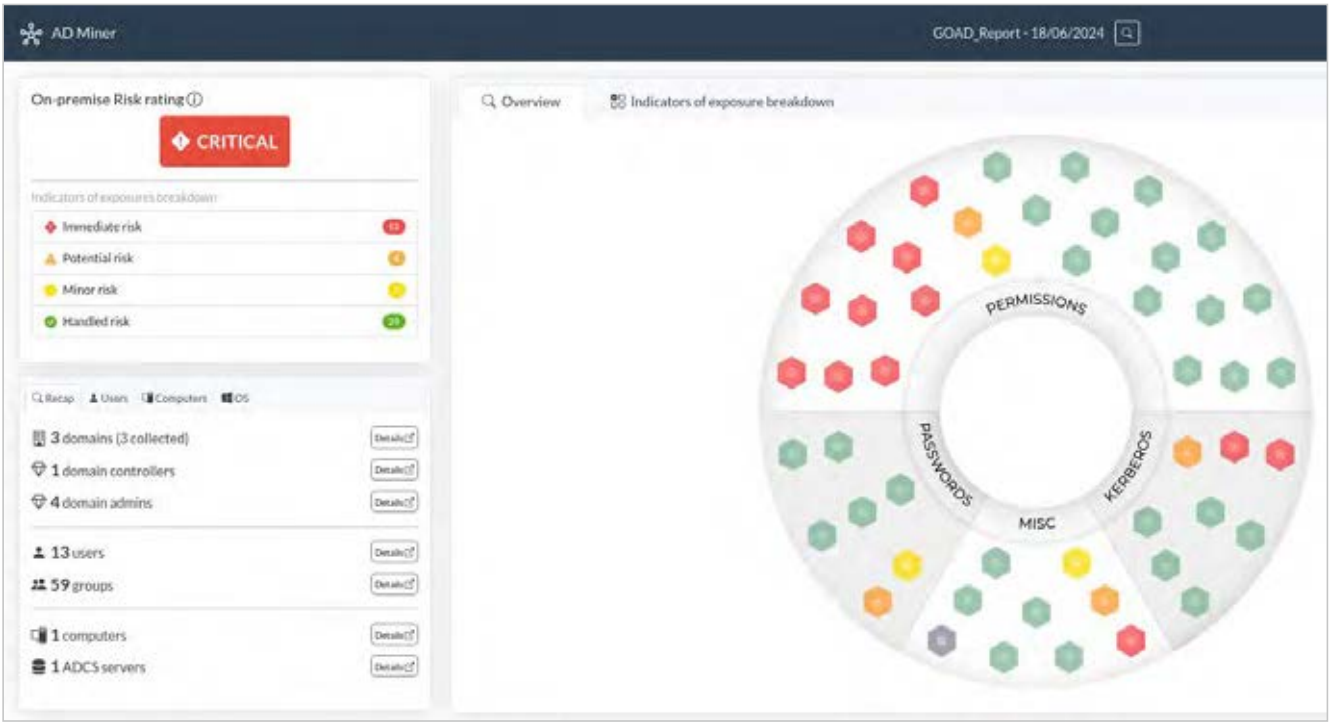
Zusätzlich erzeugt der Parameter `--pathing` weitere Ergebnisse sowohl für den Top-down-Ansatz (eingehende Berechtigungen auf hoch privilegierte Objekte) als auch für Bottom-up (ausgehende Kontrollrechte von regulären Konten). Will man mehr über die Analysen wissen, sieht man mit `--dump` die verwendeten Abfragen in der Cypher-Sprache. Zudem beschreibt AD-Recon über den Parameter `--moreHelp` in Textform, was die Abfragen bewirken.

Die Erweiterung GoodHound erstellt aus den gesammelten Daten ebenfalls eine Liste nach dem Bottom-up-Ansatz. Dabei generiert das Werkzeug eine Auflistung jener Pfade, die die größte Anzahl an Benutzern betreffen, beziehungsweise jener Schwachstellen, die am häufigsten vorkommen.

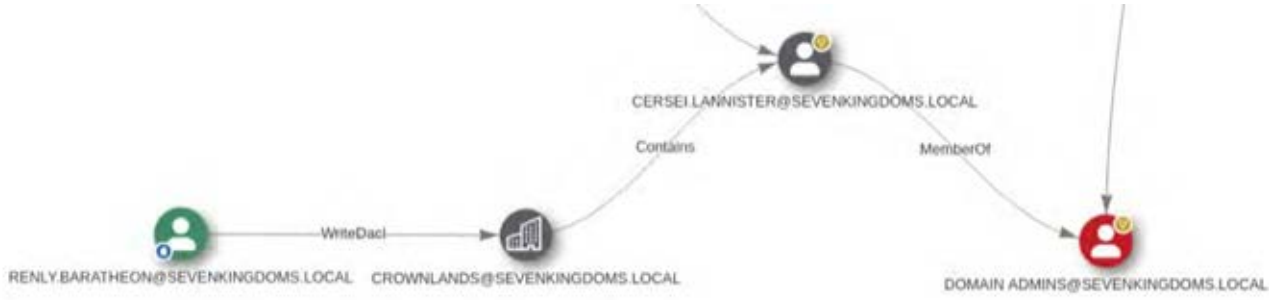
Das Installieren gelingt wegen inzwischen veralteter Abhängigkeiten nicht mehr direkt. Stattdessen muss man zunächst das GitHub-Repo klonen und die requirements.txt-Datei mit den folgenden Werten anpassen:

```
pandas>=1.3.5
py2neo>=2021.2.3
```

Anschließend kann man die Abhängigkeiten installieren und GoodHound mit



Das Dashboard des AD_Miner listet die Befunde auf und bewertet sie nach der Unmittelbarkeit des Risikos (Abb. 7).



Der visualisierte Pfad zur Gruppe der Domänenadministratoren in AD_Miner belegt die Nähe des Tools zu BloodHound (Abb. 8).

Python ausführen. Das Werkzeug stellt dafür mit dem Standardbenutzer neo4j und dem übergebenen Passwort eine Verbindung zur Neo4j-Datenbank her. Zum Befüllen der Datenbank muss BloodHound Legacy in Version 4.2 oder 4.3 verwendet werden.

```
$ pip install -r requirements.txt
$ python -m goodhound -p "bloodhound"
```

GoodHound erstellt mehrere CSV- und HTML-Dateien. In ihnen ist wie in Abbildung 4 zu erkennen, dass der Knoten „Small Council“ die meisten nicht administrativen Benutzer enthält, die einen Angriffspfad zu „High Value Targets“ haben.

Unendliche Möglichkeiten zur Erweiterung

BloodHound hat von Haus aus schon einige Abfragen, mit denen man schnell interessante Angriffspfade aufspürt. Das volle Potenzial entfaltet sich jedoch erst durch den direkten Zugriff auf die Datenbank mit der Sprache Cypher. Insbesondere das Projekt „Maximizing BloodHound with a simple suite of tools“ (kurz Max) erweitert die klassischen BloodHound-Versionen um sinnvolle Abfragen und Befehle, die sich direkt über die Konsole ausführen lassen.

Nach dem Klonen des Repos und Installieren der Abhängigkeiten markiert man beispielsweise über den Parameter `--add-note` kompromittierte Benutzer mit einer zusätzlichen Bemerkung:

```
$ python3 max.py -u neo4j -p ↵
             bloodhound mark-owned ↵
 -f owned_users.txt --add-note "owned↵
          due to guessable passwords"
```

Nach einem Neustart der Neo4j-Datenbank steht diese Zusatzinformation in BloodHound. Auch das Hinzufügen von Informationen zu geteilten Passwörtern über `--add-spw` (für „shared passwords“) kann neue Angriffspfade offenbaren. Wie zuvor wird hierbei eine Liste an Namen übergeben, die in diesem Fall dasselbe Passwort verwenden:

```
$ python3 max.py -u neo4j -p ↵
 bloodhound add-spw -f shared_pws.txt
```

In einem aktiven Red Teaming ist diese Funktion möglicherweise weniger hilfreich. Im Nachhinein kann sie aber dabei helfen, weitere Angriffspunkte zu finden. Mit dem Domain Password Audit Tool (dpa) gibt Max eine Vielzahl an Informationen zu den Passwörtern aus, die in der Domäne eingesetzt werden. Dafür verwendet es einen Hashdump aus der NTDS-Datenbank eines Domänencontrollers [6].

Warum auf Passwörter verzichten?

Auch die Macher des Werkzeugs hashcathelper haben sich gefragt, wie sie das Tool mit Daten in der BloodHound-Oberfläche durch Informationen zu geteilten Passwörtern anreichern können. Dabei arbeitet hashcathelper mit den NT-Hashes oder den gecrackten Klartextpasswörtern. In beiden Fällen muss man die NTDS-Datenbank auslesen:

```
$ impacket-secretsdump sevenking↵
                          doms.local/↵
                 cersei.lannister@192.↵
    168.56.10 -user-status -just-dc-↵
          ntlm -outputfile hashes.txt
```

Anschließend kann man einen Bericht über die Verteilung gleicher Passwörter erstellen:

```
$ hashcathelper analytics -H hashes.↵
      txt -f json -o hashes.json -d 3
```

Falls zusätzlich Klartextpasswörter verwendet werden, kann man hinter dem Parameter den Pfad zur Datei mit den Passwörtern anfügen. Schließlich werden die Domäneninformationen nach einem Neustart der Neo4j-Datenbank durch die Ergebnisse von hashcathelper angereichert:

```
$ hashcathelper bloodhound bolt://↵
    neo4j:bloodhound@localhost:7687 ↵
      sevenkingdoms.local:hashes.json
```

Wie bei Max kann man nun Konten identifizieren, die sich ein Passwort teilen (siehe Abbildung 5). Dafür kann die Abfrage aus dem Listing verwendet werden.

Besonders gut arbeitet hashcathelper in Kombination mit CrackHound. Damit kann man den einzelnen Objekten Klartextpasswörter hinzufügen. Dazu legt man eine Datei an, die Benutzernamen und die zugehörigen Passwörter in der folgenden Form enthält: DOMAIN\USER:NT-HASH:PASSWORD.

CrackHound kann man direkt aus dem Terminal ausführen. Es schreibt Klartextpasswörter direkt in die Neo4j-Datenbank:

```
$ python3 crackhound.py -f ↵
     compromised_users.txt -u neo4j ↵
      -p bloodhound -addpw -plaintext
```

Nach einem Datenbankneustart werden zum einen die Klartextpasswörter direkt in BloodHound angezeigt (Abbildung 6). Zum andern kann man dieses neue Attribut in die Pfadanalyse durch benutzerdefinierte Abfragen einbeziehen. So zeigt etwa folgende Abfrage alle Benutzer, deren Klartextpasswort bekannt ist und die zu den besonders lohnenden Zielen in BloodHound gehören („High Value Targets“ wie Domänenadministratoren oder Domänencontroller).

```
match (u1:User) WHERE u1.↵
   plaintext=True MATCH p=(u1:User)-↵
           [r:MemberOf*1..]->(m:Group
{highvalue:true}) RETURN u1
```

Weitere vordefinierte Abfragen finden sich im GitHub-Projekt von CrackHound.

Das Dashboard von AD_Miner: Gibt's das auch in schön?

Die BloodHound-Benutzerobfläche ist vergleichsweise einfach zu bedienen, doch manche Funktionen oder Pfade sind nicht immer nachvollziehbar oder etwas versteckt. Dieses Problem versucht das Werkzeug AD_Miner zu beheben. Es verbindet sich zur Neo4j-Datenbank und erstellt aus den erhaltenen Daten einen interaktiven Bericht im HTML-Format. Dieser listet die kritischsten Befunde auf einen Blick auf und vergibt auch eine Risikobewertung. Zudem ist eine Pfadanalyse ähnlich der in BloodHound integriert. Besonders interessant für Verteidiger: Der Parameter `--evolution` zeigt, wie sich das Sicherheitsniveau über einen bestimmten Zeitraum entwickelt hat. Dafür erstellt das Programm verschiedene Grafiken.

Zur Installation von AD_Miner kann pip oder pipx verwendet werden, das automatisch eine virtuelle Python-Umgebung erstellt:

```
$ pipx install git+https://github.↵
        com/Mazars-Tech/AD_Miner.git
```

Die eigentliche Abfrage erfolgt mit dem Befehl:

```
$ AD-miner -c -cf GOAD_Report -u ↵
                  neo4j -p bloodhound
```

Je nach Größe der zu testenden Domäne dauert es einige Zeit, bis die Dateien erstellt sind. Sie sind anschließend in einem angelegten Ordner mit dem Präfix rendered_ zu finden. Öffnet man die darin enthaltene index.html, erhält man Zugriff auf ein Dashboard (siehe Abbildung 7), das einen guten Überblick über die Befunde der Domäne gibt.

Die einzelnen Befunde sind interaktiv gestaltet. Wählt man einen aus, gibt er weitere Informationen preis. Die Befunde werden auch kategorisiert und bewertet, ähnlich wie es das AD-Audit-Werkzeug PingCastle [7] macht.

Dass AD_Miner stark auf BloodHound aufbaut, ist spätestens dann ersichtlich, wenn man einen Befund analysiert, der die Pfade in einer Domäne beschreibt. So lassen sich auch mit AD_Miner Pfade unprivilegierter Benutzer zu hoch privilegierten Objekten gut darstellen, beispielsweise zu der Gruppe der Domänenadmins (Abbildung 8).

AD_Miner funktioniert dabei zuverlässig sowohl mit BloodHound 4.2 und 4.3 als auch mit der Community Edition. Das Passwort zum Verbinden zur neo4j-Datenbank der Community Edition steht in der docker-compose.yml.

Es gibt zu viele Werkzeuge und Geschmacksrichtungen von BloodHound, um sie alle in diesem Artikel zu beschreiben. Auf einige Erweiterungen, die für den einen oder anderen Anwendungsfall interessant sein dürften, sei noch hingewiesen. So konzentriert sich ImproHound auf die Analyse von Durchbrüchen in einem Tiering-Modell [8]. Dazu ordnet der Sicherheitsexperte die Knoten einem bestimmten Tier-Level zu und sucht dann nach Pfaden zwischen den einzelnen Ebenen. ImproHound läuft nur auf Windows.

Der BlueHound bewirbt sich selbst als eine Alles-in-einem-Lösung für Verteidiger. Er sammelt Informationen zu Benutzerinteraktionen, Netzwerkverkehr und ungepatchten Verwundbarkeiten im internen Netzwerk, bereitet sie auf und stellt sie in einem interaktiven Dashboard dar.

Für jede Nische das richtige Werkzeug

Die Erweiterungen pyCobaltHound und BOFHound richten sich an Angreifer, die mit dem Command-and-Control-Framework [9] Cobalt Strike arbeiten. Beide Erweiterungen werden im C2-Framework geladen und dienen dort beispielsweise zur Analyse neuer Angriffspfade auf Basis erbeuteter Anmeldeinformationen aus dem LSASS-Prozess [2]. Auch lassen sich Informationen über entfernte AD-Umgebungen nur anhand der Cobalt-Strike-Logs Stück für Stück in BloodHound nachbilden. Diese Funktion ist praktisch, falls man nur einen Cobalt-Strike-Beacon, aber keinen BloodHound-Collector in der Domäne ausführen kann.

Um die genannten und viele andere Werkzeuge zu testen, ist das GOAD-Projekt von Cyril „Mayfly“ Servieres hilfreich. Das „Game of Active Directory“ (GOAD) stellt eine komplexe und absichtlich verwundbare Active-Directory-Umgebung bereit und kam auch für diesen Artikel zum Einsatz. Mehr Erweiterungen zu BloodHound stellten Esteban Rodriguez und Frank Scarpella in ihrem Vortrag „BloodHound Unleashed“ auf der CactusCon 2023 und einem Blogbeitrag vor (siehe ix.de/z539). Wer auf einen guten Onlineübersetzer zurückgreifen kann, dem sei auch der russische Blogartikel von ardent101 ans Herz gelegt, in dem diverse BloodHound-Erweiterungen unter die Lupe genommen werden: „Ein wenig über den Aufbau von Angriffsvektoren“ (Titel und verlinkter Artikel unter ix.de/z539 ins Deutsche übersetzt).

Die Anzahl der Erweiterungen und Tools ist Legion. Mit ihnen lässt sich das ohnehin schon mächtige Werkzeug BloodHound auf beinahe jeden gewünschten Einsatzzweck anpassen. Dadurch wird die Analyse von Angriffspfaden in einem Active Directory sowohl für Angreifer als auch für Verteidiger nochmals einfacher.

(ur@ix.de)

Quellen

[1] Frank Neugebauer; BloodHound CE: Beutezug durch die Windows-Domäne; iX 3/2024, S. 118
[2] Frank Ully; Fette Beute; Passwörter und Hashes – wie Angreifer die Domäne kompromittieren; iX 11/2020, S. 94
[3] Hans-Joachim Knobloch; Und ewig grüßt das Nilpferd; PetitPotam und weitere Wege, die Kontrolle über das AD zu übernehmen; iX 9/2021, S. 91
[4] Stephan Brandt; Sich selbst hacken: Scannen der eigenen Systeme; iX 8/2023, S. 40
[5] Frank Ully; Frisch geröstet; Roasting, Rechte, Richtlinien: Wie Angreifer sich im Active Directory Zugriff verschaffen; iX 12/2020, S. 92
[6] Sandro Affentranger; Schwierige Wahl; Passwortsicherheit (nicht nur) im Active Directory; iX 1/2022, S. 116
[7] Georg Bube; Interne Netzwerke selbst auditieren; iX 9/2023, S. 138
[8] Hagen Molzer; Active Directory mit dem Schichtenmodell schützen; iX 2/2023, S. 120
[9] Mischa Bachmann; Red Teaming: den Ernstfall proben; iX Special 2024 – Der iX-Notfallguide, S. 138
[10] Sämtliche im Artikel genannten Werkzeuge, Projekte und Quellen sind über ix.de/z539 zu finden.

PHILIPP LÖW

ist Penetrationstester bei der Oneconsult Deutschland AG in München und beschäftigt sich in erster Linie mit der Sicherheit von Unternehmensnetzwerken.