Sich selbst hacken: Webapplikationen angreifen

Webapplikationen sind so exponiert wie kaum eine andere Unternehmensanwendung – und doch sind viele von ihnen verwundbar. Zahlreiche Tools helfen dabei, die eigenen Web-Apps zu untersuchen und Schwachstellen zu beheben.

Von Georg Bube

Eine Welt ohne Webapplikationen ist heute undenkbar, sie sind allgegenwärtig. Gegenüber herkömmlichen Desktop-Anwendungen bieten sie einige Vorteile: Benutzer müssen sich nicht um Updates kümmern, Systemanforderungen beschränken sich auf einen aktuellen Webbrowser und eine funktionierende Internetverbindung; außerdem sind Webapplikationen günstiger in Entwicklung und Instandhaltung.

Also wo ist der Haken? Neben regulatorischen Fragen, die beispielsweise mit dem Speichern von Daten auf externen Servern aufkommen, hat sich über die Jahre gezeigt, dass Webanwendungen böswilligen Akteuren im Netz reichlich Angriffsfläche bieten [1]. Webapplikationen können von vielen unterschiedlichen Schwachstellen betroffen sein. Cross-Site Scripting (XSS), SQL Injections, Man-in-the-Middle- und Denial-of-Service-Angriffe sind nur einige davon. Solche Schwachstellen können den Crash des betroffenen Dienstes, das Abfließen vertraulicher Informationen oder gar das (unbemerkte) Übernehmen der zugrunde liegenden IT-Infrastruktur durch Kriminelle zur Folge haben. Auch Web-APIs rücken zunehmend in den Fokus von Angreifern [2, 3].

Was man tun kann

Um diese Bedrohungen zu minimieren, führen viele Unternehmen heutzutage Sicherheitsüberprüfungen ihrer Webapplikationen durch. Im Rahmen unserer Web Application Penetration Tests treffen meine Kollegen und ich bei vielen Projekten auf einige immer wiederkehrende Risiken. Diese könnten Unternehmen oft mit relativ wenig Aufwand auf eigene Faust identifizieren, um sie im Anschluss angemessen zu behandeln. Beispiele für solche Risiken sind das Preisgeben verwendeter Softwareversionen, das Verwenden veralteter Software, öffentlich zugängliche Admin-Oberflächen – schlimmstenfalls ohne Zugriffsschutz oder mit Standardpasswörtern – sowie fehlende sicherheitsrelevante HTTP-Header oder Cookie-Flags, die die Systeme und Applikationen auf einfache Weise besser härten würden.

Dieser Artikel stellt einige frei verfügbare Werkzeuge vor, mit deren Hilfe selbst entwickelte oder eingekaufte Webapplikationen größtenteils automatisiert auf solche Einfallstore hin untersucht werden können. Einige der vorgestellten Methoden sollten nur gegen lokale Testapplikationen oder mit expliziter Erlaubnis der verantwortlichen Betreiber oder Personen durchgeführt werden. Das betrifft insbesondere Methoden, die nicht nur passiv die mitgeschnittene Kommunikation einer Webapplikation mit dem Webserver auswerten, sondern aktiv nach potenziellen Sicherheitsrisiken in Anwendungen suchen.

Starten mit OSINT

Als Erstes kann man sich einen Überblick darüber verschaffen, welche Software – gegebenenfalls inklusive deren Version – und verwendeten Technologien für einen Angreifer leicht sichtbar sind. Das Sammeln solcher öffentlich verfügbaren Informationen wird als OSINT (Open

iX-TRACT

- Für Webapplikationen existiert eine große Bandbreite an möglichen Schwachstellen, die Angreifer mit teils gravierenden Auswirkungen ausnutzen können.
- Mithilfe automatisierter Tools kann man relativ leicht einige „Quick Wins“ ausmachen. Damit sind insbesondere Sicherheitsrisiken gemeint, die sich durch kleine Anpassungen der Konfiguration einfach beheben lassen.
- Mit den Werkzeugen BuiltWith, Wappalyzer, Burp Suite, OWASP Zed Attack Proxy (ZAP) und feroxbuster kann man eigene Webapplikationen selbst überprüfen.

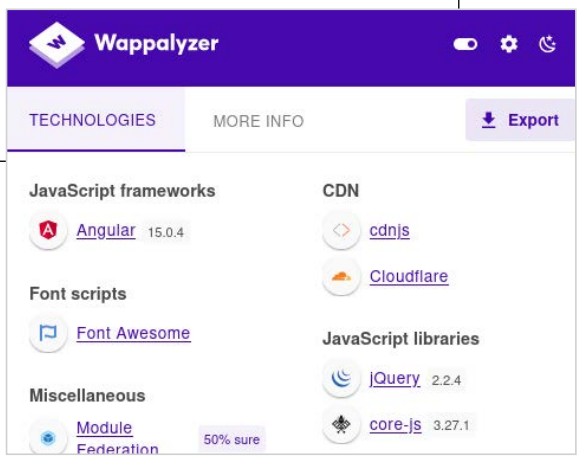
Listing 1: Installieren und Ausführen von Docker und Juice Shop

```
$ sudo apt update
$ sudo apt install -y docker.io docker-compose
$ sudo systemctl enable docker --now
$ echo "127.0.0.1 juiceshop.owasp" | sudo tee -a /etc/hosts
$ sudo docker pull bkimminich/juice-shop
$ sudo docker run --rm -p 127.0.0.1:3000:3000 bkimminich/juice-shop
```

Source Intelligence) bezeichnet und ist für Kriminelle wie Sicherheitstester der erste Schritt. Diese Informationen können einem Angreifer dabei helfen, gezielt nach Schwachstellen zu suchen. Im schlimmsten Fall sind öffentlich bekannte Exploits verfügbar, die mit geringem Aufwand eine komplette Kompromittierung des betroffenen Systems zur Folge haben.

Um sich einen Überblick über die eingesetzte Technologie zu verschaffen, kann man frei verfügbare Webseiten-Profiler wie BuiltWith und Wappalyzer nutzen (siehe ix.de/z3sj). Beide bieten weitreichende Funktionen zum Webseiten-Profiling, für eine Marktanalyse, zum Erstellen von Berichten mit spezifischen Informationen – zum Beispiel Webseiten mit bestimmter Technologie und Kontaktdaten – sowie einiges mehr. Hierfür ist ein kostenloses Konto notwendig, das eine begrenzte Anzahl an Funktionen umfasst. Der BuiltWith-Account erlaubt beispielsweise die kostenlose Abfrage von bis zu zehn detaillierten Profilen pro Tag. Diese enthalten Informationen zur ersten und letzten Nutzung der aufgeführten Technologien, darunter verwendete Webserver, Frameworks, Content-Management-Systeme (CMS), JavaScript-Bibliotheken, Content-Delivery-Netzwerke (CDN), Analytics und Tracking.

Beide Webseiten-Profiler bieten außerdem Browsererweiterungen für Firefox und Chrome, die in Form eines Aufklappmenüs eine Übersicht über die erkannten Technologien anbieten. Abbildung 1 zeigt dies beispielhaft für Wappalyzer. Sobald eingesetzte Software und Versionen identifiziert sind, kann man überprüfen, ob sie in der aktuellen Version vorliegen oder eines Updates bedürfen. Aus Sicht des Webanwendungsbetreibers ist es sinnvoll, die verwendete Software zu inventarisieren [4] und regelmäßig auf Updates zu überprüfen, um die Angriffsfläche möglichst gering zu halten.



Dank der Browsererweiterungen für Firefox und Chrome lassen sich die Webseiten-Profiler komfortabel nutzen (Abb. 1).

Installieren des OWASP Juice Shop

Wie oben angesprochen ist es nicht legal, gewisse Werkzeuge ohne explizites Einverständnis auf beliebige Webapplikationen anzuwenden. Daher wird zur Illustrierung der Vorgehensweise im weiteren Verlauf dieses Artikels eine absichtlich verwundbare Webapplikation als Ziel fungieren. Davon gibt es mittlerweile eine Reihe, beispielsweise den OWASP Juice Shop oder das Totally Insecure Web Application Project (TIWAP). Weitere absichtlich verwundbare Webapplikationen finden sich zum Üben im Netz (siehe ix.de/z3sj).

Für die ersten Versuche bietet sich der OWASP Juice Shop an, installiert in einer virtuellen Maschine mit Kali Linux. Das Einrichten von Kali Linux wurde bereits in im ersten Teil der Sich-selbst-hacken-Reihe behandelt (siehe Seite 40). Der OWASP Juice Shop ist eine unsichere Webapplikation, die für Sicherheitstrainings, Awareness-Schulungen und als Ziel für Sicherheitstools dienen soll. Sie verwendet moderne Technologien wie Node.js für die REST-Schnittstelle und Angular für die Single-Page-Anwendung (SPA). Zur Verfügung gestellt und gepflegt wird der sprichwörtliche Saftladen von OWASP, dem Open Web Application Security Project (siehe ix.de/z3sj), einer Non-Profit-Organisation, die sich bereits mit zahlreichen Werkzeugen, Projekten und Veranstaltungen um die Sicherheit von Webanwendungen verdient gemacht hat.

Die verwundbare Applikation installiert man mit einem Docker-Image durch das Ausführen der Befehle aus Listing 1 in der vorbereiteten Kali-Linux-VM. Die hosts-Datei wird dabei außerdem um einen Domänennamen für Localhost ergänzt, unter dem der Tester den Juice Shop dann aufrufen kann.

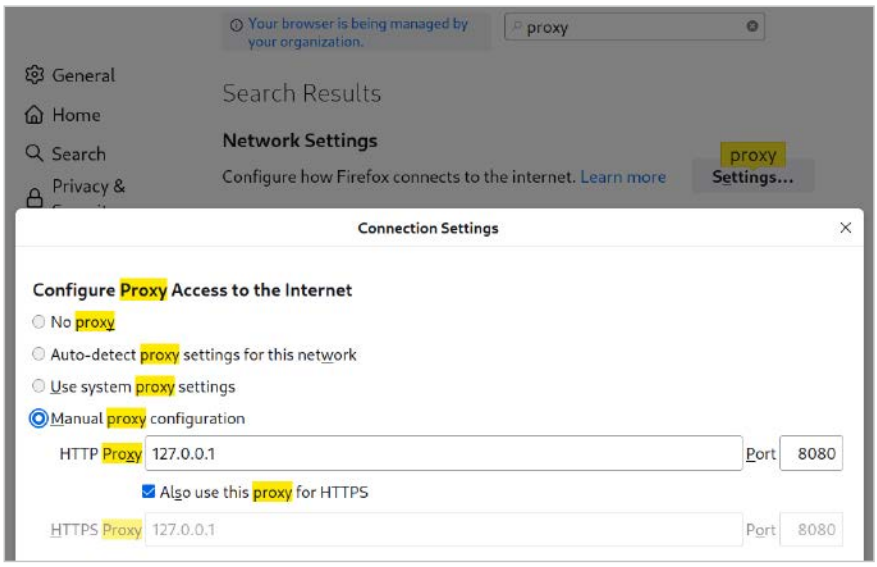
Der letzte Befehl aus Listing 1 startet den Docker-Container, der auf Localhost unter Port 3000 den Juice Shop exponiert. Dieser kann nun durch Aufruf von http://juiceshop.owasp:3000 besucht werden. Der verwundbare Webshop enthält alle Schwachstellen aus den OWASP Top Ten [1], einer von der OWASP veröffentlichten Liste der zehn kritischsten Schwachstellen für Webanwendungen. Auch die meisten der OWASP API Security Top Ten [2], der Entsprechung für Webschnittstellen, sind mit an Bord.

Darüber hinaus enthält der Juice Shop viele weitere Lücken, die in Form von Herausforderungen in der versteckten Fortschrittsübersicht aufgelistet sind – eine der ersten Herausforderungen besteht darin, das verborgene Scoreboard zu finden.

Interception-Proxy – der Angreifer in der Mitte

Zur automatisierten und manuellen Untersuchung von Webapplikationen auf Sicherheitsrisiken wird üblicherweise ein sogenannter Interception-Proxy verwendet. Mit diesem wird ein Man-in-the-Middle-Angriff auf die Kommunikation zwischen Webbrowser und Webserver durchgeführt und der gesamte Datenverkehr mitgeschnitten. Die Anfragen können im Interception-Proxy im Detail betrachtet und auf mögliche Angriffspunk-

Tutorialinhalt



In den Firefox-Einstellungen kann man den Proxy manuell für unverschlüsselte wie verschlüsselte Verbindungen konfigurieren (Abb. 2).

te hin analysiert werden. Außerdem erlaubt er es, Anfragen an den Webserver abzufangen, Parameter zu manipulieren und die manipulierte Anfrage weiter an den Webserver zu schicken oder aber bereits gesendete Anfragen erneut zu versenden – mit oder ohne Änderungen.

In der aktuellen Version leitet Firefox Anfragen nach 127.0.0.1 nie über einen Proxy, selbst wenn einer konfiguriert ist. Diese Einstellung kann man durch Aufrufen von `about:config` über die Adressleiste, Suche nach `network.proxy_allow_hijacking_localhost` und Ändern des Eintrags auf `true` deaktivieren.

Der wohl bekannteste Interception-Proxy ist die kommerzielle Burp Suite von PortSwigger (siehe ix.de/z3sj). Dabei handelt es sich um eine Java-Applikation, die unter den meisten Betriebssystemen läuft. Eine Lizenz kostet unter 500 Euro und ist damit relativ preiswert. Es gibt außerdem eine kostenlose Community Edition der Burp Suite, die über einen eingeschränkten Funktionsumfang verfügt. Im Netz gibt es zahlreiche Anleitungen, Videotutorials und Kurse zur Burp Suite. Zudem erleichtert ein umfangreiches Ökosystem nützlicher Erweiterungen die Arbeit mit Burp. Diese könnte man prinzipiell auch selbst entwickeln.

Eine kostenlose Alternative ist der quelloffene OWASP Zed Attack Proxy (ZAP; siehe ix.de/z3sj). ZAP bietet ähnliche Funktionen wie Burp, allerdings findet sich im Netz deutlich weniger Dokumentation zur Software. Es gibt ebenfalls ein Ökosystem an Erweiterungen. ZAP ist in den Paketquellen von Kali Linux enthalten und lässt sich mit `sudo apt install zaproxy` einfach installieren. Der Kali-VM sollten mindestens 6 GByte Arbeitsspeicher zur Verfügung stehen, denn ZAP ist genauso wie Burp sehr ressourcenhungrig. Beim ersten Öffnen von ZAP über das Anwendungsmenü von Kali erscheint möglicherweise der Add-on-Manager, der ein Update einfordert.

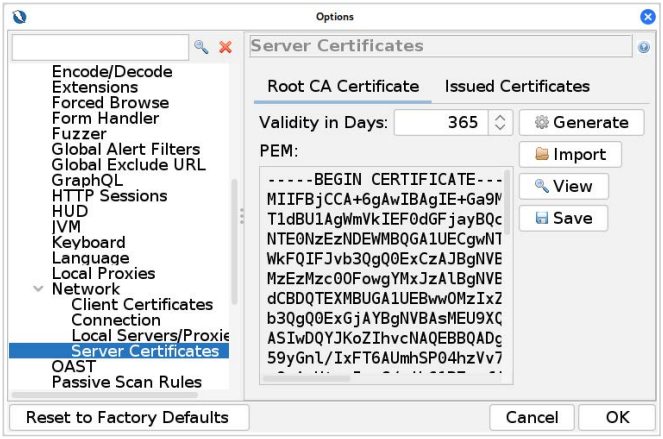
Speichern für später

Beim Start kann man entscheiden, ob ZAP die Sitzung speichern soll oder nicht. Eine gespeicherte Sitzung lässt sich zu einem späteren Zeitpunkt wieder öffnen, sodass die Resultate daraus weiterhin verfügbar sind. Insbesondere für mehrtägige Tests oder im Fall von Abstürzen kann dies hilfreich sein. ZAP läuft standardmäßig unter 127.0.0.1 auf Port 8080.

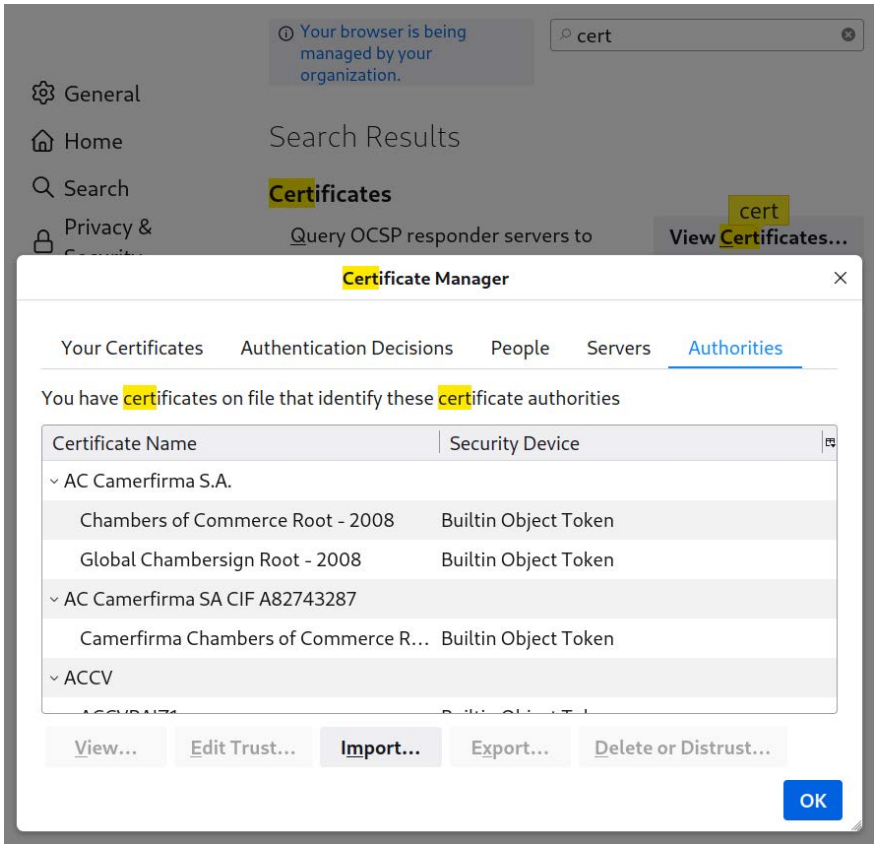
Für unverschlüsselte Verbindungen genügt es, im Webbrowser ZAP als Proxy zu konfigurieren. Dies ist für Firefox unter Aufruf von Settings im Hamburger-Menü möglich. In den Einstellungen sucht man nach proxy und klickt auf die Settings-Schaltfläche der „Network Settings“. In den sich öffnenden „Connection Settings“ wählt man „Manual proxy configuration“ aus und trägt als HTTP-Proxy 127.0.0.1 mit Port 8080 ein (Abbildung 2). Falls die Standardkonfiguration von ZAP verändert wurde, muss man den geänderten Port hier konfigurieren. Hinzu kommt ein Häkchen für „Also use this proxy for HTTPS“, damit Firefox ZAP ebenfalls als Proxy für mit TLS verschlüsselte HTTP-Verbindungen verwendet.

Außerdem muss noch das Zertifikat, das ZAP für den Man-in-the-Middle-Angriff auf die HTTPS-Verbindung verwendet, im Browser importiert werden. Andernfalls kennzeichnet der Browser die Website als unsicher und eine Verbindung ist gegebenenfalls überhaupt nicht möglich. Das benötigte Zertifikat lässt sich aus den Einstellungen von ZAP exportieren. Diese können im Menüpunkt Tools ganz unten mit „Options ...“ oder alternativ über das Zahnradsymbol in der Zeile unter der Menüleiste geöffnet werden. In den Einstellungen findet sich das von ZAP verwendete Zertifikat im Unterpunkt „Server Certificates“ des Eintrags Network, wie in Abbildung 3 gezeigt.

Das Zertifikat wird mit Save am Speicherort der Wahl abgelegt und in den Einstellungen von Firefox (Hamburger-Menü, Settings, Suche nach „View Certificates“) über die Import-Schaltfläche in Firefox installiert. Nun können Tester HTTPS-Verbindungen zu Webservern über den Interception-Proxy ohne Warnung oder Fehlermeldung besuchen (Abbildung 4). Außerdem sollte man „ZAP HUD“ über die Symbolleiste unterhalb der Menüleiste deaktivieren, damit unverschlüsselte Verbindungen wie erwartet im Browser auch als unverschlüsselt erkennbar sind (weit rechts; links vom Firefox-Symbol).



Der Export des ZAP-Zertifikats ist für das Belauschen verschlüsselter Verbindungen erforderlich (Abb. 3).



Nach Einbinden des Zertifikats in Firefox kann man sich ohne Fehlermeldungen in HTTPS-Verbindungen einklinken (Abb. 4).

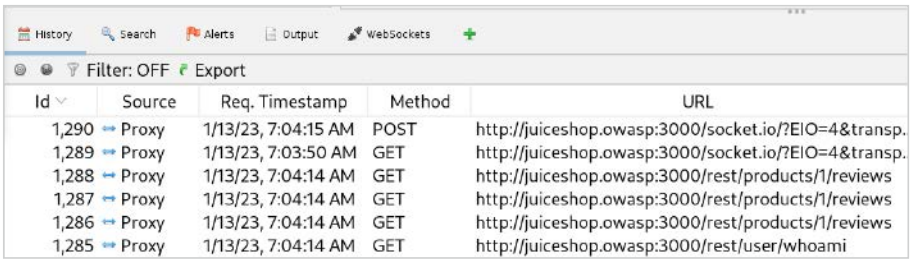
Im unteren Bereich von ZAP finden sich in der Registerkarte History die Anfragen, die durch den Interception-Proxy geleitet wurden (siehe Abbildung 5). Sie lassen sich durch Klick auf die ID-Spalte chronologisch sortieren.

Arbeiten mit ZAP

Durch Klicken auf einzelne Einträge in der Historie zeigt ZAP die Details der Anfragen und Serverantworten weiter oben an. In der Symbolzeile unter der Menüzeile kann man die Ansicht so konfigurieren, dass Anfrage und Serverantwort nebeneinander dargestellt werden, und die Breite von beiden nach Belieben anpassen (Abbildung 6).

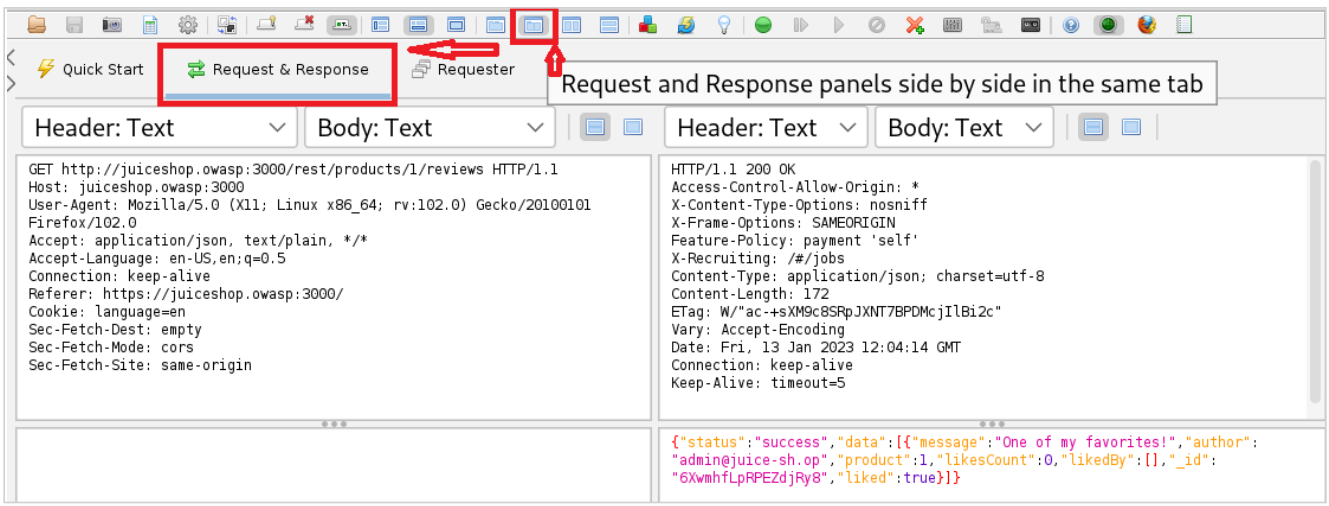
Durch einen Rechtsklick auf die Anfrage und Wählen von „Open in Requester Tab“ geht die Anfrage in den Requester-Tab, von wo sie sich mit oder ohne Änderungen erneut abschicken lässt.

In ZAP kann man einen Kontext („Context“) definieren. Standardmäßig ist er nicht eingeschränkt. Um ihn einzuschränken oder weitere Kontexte zu definieren, nutzt man reguläre Ausdrücke. Diese müssen mit der gesamten URL übereinstimmen. Laut Dokumentation soll ein Kontext einer Webapplikation entsprechen. Beim Testen kann dann immer der Kontext der Webapplikation, die gerade untersucht wird, als aktiv gesetzt werden. Ein Kontext lässt sich über das Rechtsklickmenü einer Domäne in der Sites-Ansicht unter „Include Site in Context“ ergänzen oder dort neu definieren.



Die History-Registerkarte in ZAP bietet eine Übersicht über die umgeleiteten Verbindungen (Abb. 5).

Durch Doppelklick auf einen Kontext in der Sites-Ansicht öffnet sich ebenfalls das Konfigurationsfenster. Dort gibt man durch reguläre Ausdrücke an, was im Kontext eingeschlossen sein soll und was nicht (Unterpunkte „Include in Context“, „Exclude from Context“). Zudem bietet ZAP noch weitere Optionen, den Kontext zu verfeinern: Auf der höchsten Ebene lässt sich das Häkchen „In Scope“ setzen. Die Kontexte und Webseiten im Scope haben in der Sites-Ansicht ein ro-



Konfigurieren der „Request & Response“-Ansicht in ZAP (Abb. 6).

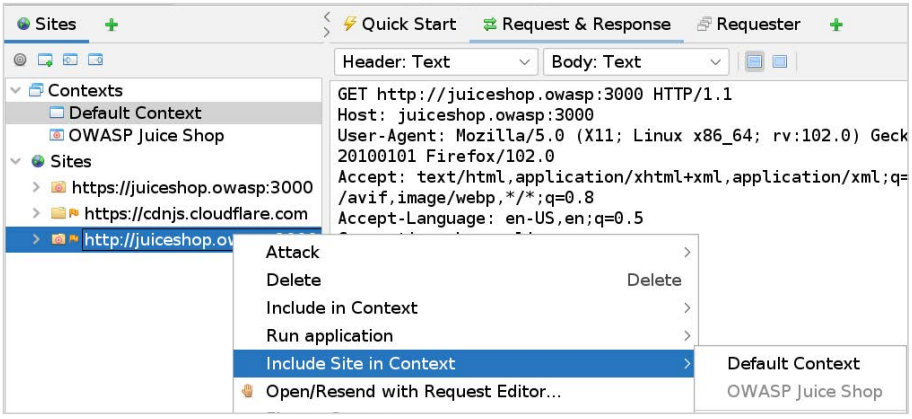
tes Fadenkreuz auf ihrem Symbol (siehe Abbildung 7).

ZAP verfügt über vier Modi, in denen es operieren kann. Der „Safe Mode“ erlaubt überhaupt keine potenziell gefährlichen Aktionen. Der „Protected Mode“ erlaubt diese ausschließlich gegen URLs im Scope. Im „Standard Mode“ gibt es keine Einschränkungen, welche (potenziell) gefährlichen Aktionen ausgeführt werden können. Der „ATTACK Mode“ führt zusätzlich automatisch „Active Scans“ gegen jeden neuen Pfad von Domänen im Scope durch. Um diese Active Scans geht es im Folgenden.

Alerts und Active Scans

Für die restlichen Ausführungen sollte der Juice Shop zum Default-Kontext als einzigem Kontext hinzugefügt werden (Rechtsklick auf http://juiceshop.owasp:3000, Include Site in Context, Default Context). Zudem stellt man den Modus Protected in der Menüleiste unter „Edit – ZAP Mode“ ein. Diese Vorsichtsmaßnahmen gewährleisten, dass ausschließlich der im Scope konfigurierte Juice Shop mit

Angezeigte Meldungen können auf den Scope eingeschränkt werden (Abb. 8).



Hinzufügen des Juice Shop zum Kontext (Abb. 7).

potenziell gefährlichen Aktionen angegriffen wird. Wer auf Nummer sicher gehen möchte, kann zudem die Internetverbindung der virtuellen Maschine kappen.

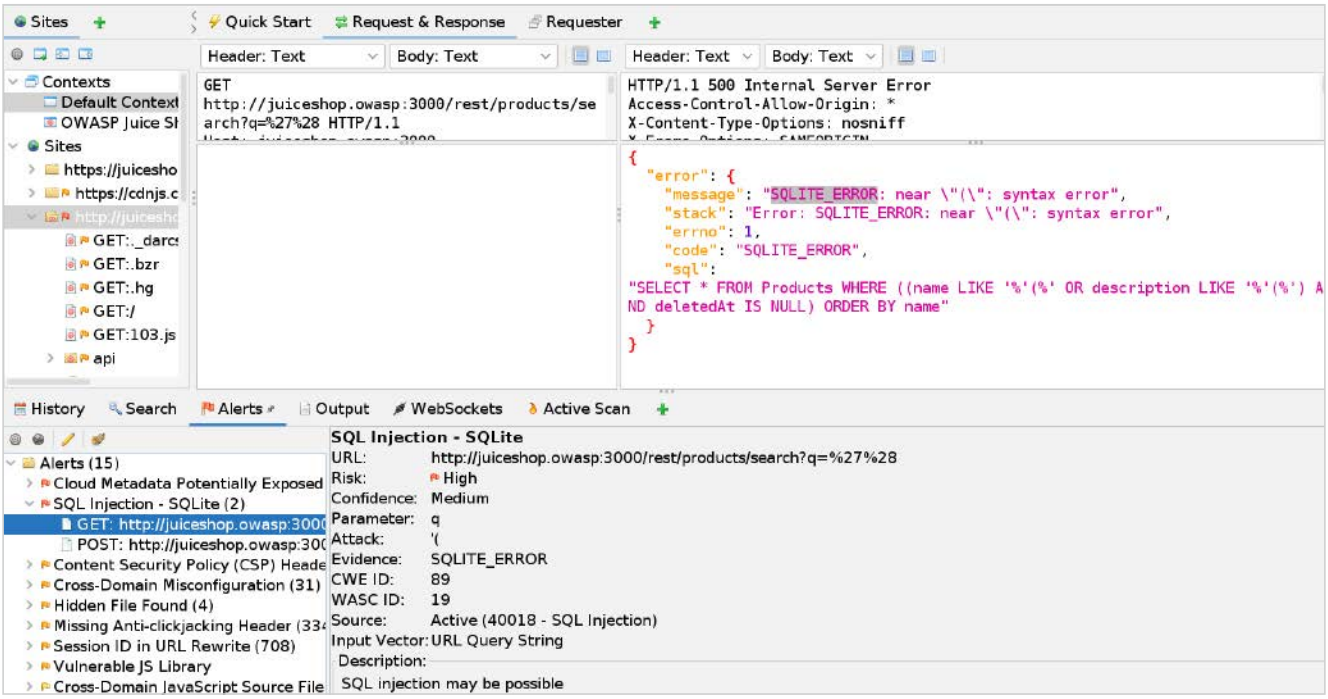
Im ZAP-Fenster findet sich unten neben der Registerkarte History das Register Alerts. Sobald eine Webanwendung durch ZAP geleitet wird, analysiert dieser die Anfragen passiv im Hintergrund und meldet in der Alerts-Registerkarte potenzielle Sicherheitsrisiken. Die angezeigten Alarme kann der Tester auf URLs im Scope einschränken (Zielscheibensymbol oberhalb der Alerts-Auflistung links, Abbildung 8).

Auslöser für Alarme können bei normaler Nutzung einer Webanwendung beispielsweise veraltete Software, fehlende sicherheitsrelevante HTTP-Header und Cookie-Flags oder heikle Daten wie Sitzungstoken in der URL sein. Ein Klick auf einen Alarm zeigt detaillierte Informationen an. Betroffene Anfragen sind in der Baumstruktur versteckt und können bei Bedarf in der „Request & Response“-Registerkarte angeschaut werden (Abbildung 9). Mithilfe der Informationen und der betroffenen Anfragen sollte man das Sicherheitsrisiko lokalisieren, um entsprechende Stellen im Quellcode zu überprüfen. Die Lösungsvorschläge geben zudem Hinweise darauf, wie die Schwachstelle behoben werden kann.

Angreifen mit den üblichen Verdächtigen

ZAP erlaubt neben dieser passiven Analyse auch die aktive Suche nach potenziellen Lücken durch die genannten Active Scans. Sie führen automatisiert bekannte Angriffe gegen die Webanwendung durch, darunter Path Traversal, Remote File Inclusion, Cross-Site Scripting und SQL Injection.

Die Konfiguration eines Active Scan lässt sich per Rechtsklick auf den Juice Shop in der Sites-Ansicht und Wählen von „Attack – Active Scan“ oder auch aus der History-Registerkarte heraus öffnen.



Lokalisierung einer Schwachstelle mithilfe der Alerts-Registerkarte in ZAP (Abb. 9).

Active Scans lassen sich wie viele andere Funktionen in ZAP an die eigenen Bedürfnisse anpassen. Die Standardeinstellungen genügen für diesen Artikel. Die Option Recurse stößt einen rekursiven Scan der gesamten Webapplikation mit den Berechtigungen des aktuellen Benutzers an. Das Ausführen des Active Scan braucht etwas Zeit; den Fortschritt kann man in der zugehörigen Registerkarte verfolgen, die sich automatisch öffnet.

Der Active Scan auf http://juiceshop.owasp:3000 liefert zwei mögliche SQL Injections, eine davon in der Suchfunktion (siehe Abbildung 9). In diesem Fall verrät die Serverantwort die genaue SQL-Abfrage. Durch Rechtsklick auf die Anfrage und unter „Open in Requester Tab“ schickt man die betroffene Anfrage an den Requester, ein weiteres ZAP-Modul.

Mit dem Wissen um die Struktur des SQL-Statements kann man im Requester eine veränderte Anfrage an den Server schicken, dessen Antwort nun alle gelöschten Produkte enthält (siehe Abbildung 10). Eine Suche nach der „Christmas Super-Surprise-Box (2014 Edition)“ in der Weboberfläche des Juice Shop selbst liefert erwartungsgemäß keine Ergebnisse. Wer vorsichtig ist, führt die Active Scans ausschließlich gegen Testinstanzen der eigenen Webapplikation durch.

Dirbusting und Standardpasswörter

In unseren Projekten finden wir auch regelmäßig auf Webanwendungen verfügbare – gar administrative – Endpunkte, von denen die Unternehmen selbst überhaupt nicht wissen, dass von außen auf sie zugegriffen werden kann. Es gibt verschiedene Dirbusting-Werkzeuge, die versuchen, solche Endpunkte automatisiert zu finden. Das funktioniert durch simples Ausprobieren möglicher bekannter Endpunkte. Die standardmäßig verwendeten Listen können üblicherweise durch eigene Inhalte ergänzt oder ersetzt werden. Falls solche Endpunkte bekannt sind, kann man natürlich direkt prüfen, ob sie von außen erreichbar sind oder nicht.

Ein in den Paketquellen von Kali Linux enthaltenes aktuelles Dirbusting-Werkzeug ist feroxbuster. Es wird per

```
sudo apt install feroxbuster
```

installiert. Der folgende Befehl führt einen Angriff gegen den Juice Shop durch, wobei die Optionen `--no-recursion` und `--rate-limit` verhindern, dass der Juice Shop einen „JavaScript heap out of memory“-Fehler erleidet.

```
feroxbuster --url http://juiceshop.↵
          owasp:3000 --no-recursion ↵
                       --rate-limit 50
```

Bei seinem Durchlauf hat feroxbuster unter anderem den Endpunkt „http://juiceshop.owasp:3000/ftp“ entdeckt, der beim Aufruf vertrauliche Daten preisgibt (Abbildung 11).

feroxbuster verfügt über weitere Parameter, mit denen das Verhalten an die eigenen Bedürfnisse angepasst werden kann. Man kann sie auf der Man-Page von feroxbuster einsehen. Erwähnt sei hier der Parameter `--wordlist`, mit dem eine spezielle Wörterliste übergeben werden kann. In Kali Linux gibt es von Haus aus eine Reihe von allgemeinen und softwarespezifischen Wörterlisten in folgenden Ordnern:

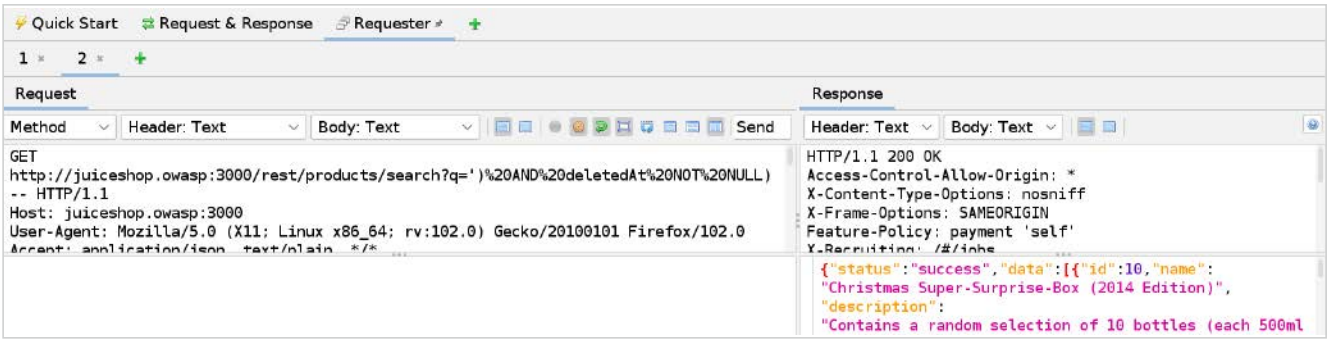
```
/usr/share/wordlists/dirb
/usr/share/wordlists/dirbuster
/usr/share/seclists/Discovery/↵
                          Web-Content
```

Falls Kali Linux für die eingesetzte Software keine geeignete Wörterliste enthält oder die vorhandenen ergänzt werden sollen, bietet die Seite Assetnote Wordlists (siehe ix.de/z3sj) weitere Listen. Auch eine Suche im Netz kann hilfreich sein.

Falls feroxbuster eine administrative Oberfläche findet, wäre es sinnvoll zu überprüfen, ob es für sie öffentlich bekannte Zugangsdaten gibt. Auch wenn mittlerweile Konsens besteht, dass Benutzer besser während der Installation sichere Passwörter für administrative Konten setzen, finden sich häufig in älteren Anwendungen Standardzugangsdaten. Das liegt nicht zuletzt daran, dass der Endbenutzer das Installieren nicht immer selbst durchführt.

Nach Standardpasswörtern kann man im Netz suchen. Alternativ nutzt man Datenbanken, die solche Standardzugangsdaten sammeln. Ein Beispiel dafür ist das „Default Credentials Cheat Sheet“ (siehe ix.de/z3sj), ebenfalls für Kali Linux verfügbar. Mit ihm kann man das geklonte Repository mithilfe von Python in ein CLI-Tool verwandeln und damit nach bestimmten Produkten suchen (siehe Listing 2).

Falls beim Installieren keine Passwörter für administrative Accounts gesetzt werden müssen oder zufällig erzeugt werden, sollte man die Standardzugangsdaten herausfinden und durch sichere,



Mit Kenntnis der Struktur der SQL-Statements kann man der Webanwendung Informationen entlocken, die nicht mehr öffentlich verfügbar sind (Abb. 10).

idealerweise von einem Generator erzeugte Passwörter ersetzen. Für administrative Konten sollten sie eine Mindestlänge von 14 Zeichen haben.

Fazit

Webanwendungen werden wir wohl nicht mehr los – also müssen wir auch in Zukunft mit ihrer großen Angriffsfläche leben. Die Entwicklung neuer Webapplikationen ist oft davon getrieben, ein funktionierendes Produkt, und weniger davon, ein sicheres Produkt zu entwickeln. Gedanken an die Sicherheit kommen daher oft erst später.

In diesem Artikel wurden einige Werkzeuge vorgestellt, mit deren Hilfe sich automatisiert einige Quick Wins identifizieren lassen, die in der Regel leicht durch Anpassung der relevanten Konfiguration zu beheben sind und die Systeme insgesamt besser härten. Weitere Werkzeuge zum Test von Webanwendungen und APIs sind in früheren iX-Artikeln beschrieben [2, 3, 5].

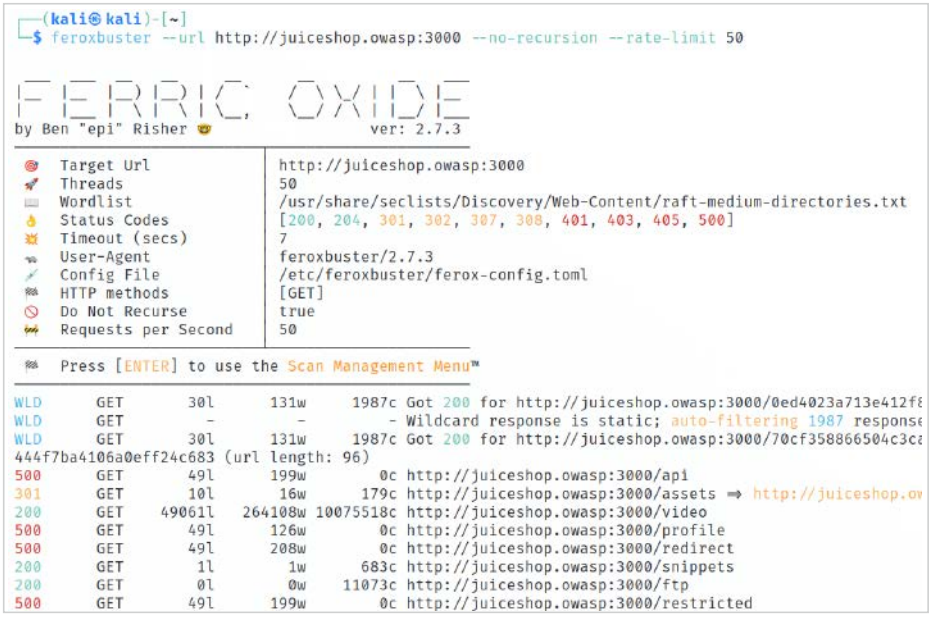
Teilweise lassen sich auch tiefer liegende Sicherheitsprobleme feststellen, wie die SQL Injection am Beispiel des OWASP Juice Shops gezeigt hat. Erkenntnisse aus der Nutzung dieser Tools sollten dabei mit der Zeit in (Entwicklungs-)Prozesse einfließen, sodass immer weniger dieser Quick Wins erst im Nachhinein umgesetzt werden müssen.

In den kommenden Artikeln der Reihe geht es unter anderem darum, durch das Sammeln öffentlich verfügbarer Informationen womöglich unbekannte Server und Webapplikationen der eigenen Organisation zu finden, sowie um das Testen der internen Umgebung. (ur@ix.de)

Listing 2: Installation und Suche mit dem Default Credentials Cheat Sheet

```
$ sudo apt install -y pipx
$ pipx ensurepath && source ~/.profile $ pipx install git+https://github.com/↵
                                                    ihebski/DefaultCreds-cheat-sheet

$ creds search tomcat
[+] Download database...
+---------------------------------+------------+------------+
| Product                         |  username  |  password  |
+---------------------------------+------------+------------+
| apache tomcat host manager (web) |   admin    |   admin    |
| apache tomcat host manager (web) |   ADMIN    |   ADMIN    |
| apache tomcat host manager (web) |   admin    |  <blank>   |
```



Mit speziellen Tools, hier feroxbuster, kann man nach potenziell sensiblen Endpunkten suchen (Abb. 11).

Quellen

[1] Tobias Glemser; Zwo, eins, Risiko ..., OWASP Top 10 in neuer Version; iX 2/2022, S. 86

[2] Frank Ully; APIs sicher entwickeln; iX 7/2022, S. 52

[3] Frank Ully; Schwachstellen in APIs aufspüren; iX 7/2022, S. 64

[4] Udo Schneider; SBOMs – Stücklisten für Software; iX 10/2022, S. 54

[5] Christian Schneider; Automatisch sicher, Web-Security-Tests mit Open-Source-Werkzeugen; iX 7/2019, S. 46

[6] Die im Artikel angesprochenen Werkzeuge, Dokumentationen, Hilfsmittel und Projekte sind über ix.de/z3sj zu finden.

GEORG BUBE

ist Senior Penetration Tester bei der Oneconsult Deutschland AG in München. Im Laufe der Jahre hat er zahlreiche Web Application Penetration Tests in unterschiedlichsten Branchen durchgeführt.