Grundpraktikum Netz- und Datensicherheit

Thema:

SIEM
Forensische Log-Analyse mit Splunk

Lehrstuhl für Netz- und Datensicherheit
Ruhr-Universität Bochum

Versuchsdurchführung: Raum ID 2/168

Betreuung: Dominik Noß
Zusammengestellt von: Louis Jannett
Stand: 4. März 2020
Version: 2.0

Inhaltsverzeichnis

1 Motivation

Bei der Suche nach Jobangeboten im Bereich IT-Security stößt man immer wieder auf die Jobtitel *Threat Detection Engineer* und *Incident Responder*. Zu den Aufgaben dieser Berufsbilder gehören unter anderem die Detektion und Mitigation von Angriffen in Echtzeit sowie die nachfolgende detaillierte Analyse und Auswertung von erfolgreichen Angriffen auf Unternehmensinfrastrukturen.

Ein ausgeklügeltes SIEM ist existenziell für die tägliche Arbeit im Security Operations Center eines Unternehmens. Daher wird dieser Versuch grundlegende Kenntnisse im Umgang mit dem SIEM System Splunk vermitteln.

2 Vorbereitung

Dieser Versuch erweitert den Versuch zum Thema *Web-Service-Sicherheit* um eine abschließende Analyse der mitgeschnittenen Logs. Der gesamte Webtraffic (mit Ausnahme von HTTPS) wurde im WS 18/19 aufgezeichnet und soll in diesem Versuch von Ihnen analysiert und auf die durchgeführten Webangriffe untersucht werden.

Bitte lesen Sie sich die Grundlagen im Abschnitt 3 aufmerksam durch. Die Tabellen 1 im Abschnitt 4.2 und 2 im Abschnitt 4.3 geben eine Übersicht über die zur Verfügung stehenden Informationen in den Logs und können bei der Bearbeitung des Versuches sehr nützlich sein.

Eine Übersicht und kurze Erklärung aller in diesem Versuch benötigten Befehle der Search Processing Language befindet sich im Abschnitt 4.1. Detaillierte Informationen zur Splunk Plattform und zu allen Befehlen findet man in der offiziellen Dokumentation [4]. Als Vorbereitung kann ebenfalls der offizielle Splunk Quick Reference Guide [5] nützlich sein, der einen Überblick über die gesamte Splunk Plattform gibt.

Lesen Sie abschließend die Kontrollfragen im Abschnitt 5 und notieren Sie sich eine kurze Antwort zu jeder Frage. Die Kontrollfragen werden vor dem Versuch von Ihrem Betreuer abgefragt.

3 Grundlagen

Dieser Abschnitt vermittelt grundlegende Kenntnisse im Bereich der Log-Analyse und SIEMs. Außerdem werden die wichtigsten Grundlagen im Umgang mit Splunk vorgestellt.

3.1 Einführung: SIEM

3.1.1 Logs

Eine Logdatei ist eine strukturierte Sammlung von Ereignissen in einem bestimmten System. Die Logdatei ist häufig zeitlich sortiert und beinhaltet pro Ereignis einen Timestamp. Jedes Ereignis wird in einer Logdatei durch einen Delimiter getrennt, sodass automatische Parser die einzelnen Events identifizieren können. Häufig trennen einfache Zeilenumbrüche Ereignisse, wobei jede Zeile dann genau ein Ereignis repräsentiert. Folgende Beispiele illustrieren praktische Logtypen:

Webserver-Log Ein Webserver protokolliert Informationen über die Anfragen und Antworten (Requests & Responses) auf der Anwendungsschicht. Für jede angefragte Datei generiert der Webserver einen eigenständigen Eintrag in der Logdatei. Das bedeutet, dass beim Aufruf einer Webseite meistens mehrere Logeinträge generiert werden (z.B. HTML, JavaScript, CSS und Bilder). Der

Betreiber einer Webseite nutzt die Logs, um das Verhalten seiner Kunden zu analysieren und potentielle Angriffe durch schädliche Payloads zu erkennen (z.B. XSS, SQLi, DoS).

Proxy-Log Ein Proxy nimmt Anfragen entgegen und leitet diese dann an das ursprüngliche Ziel weiter. Als Vermittler kann der Proxy den gesamten Traffic zwischen Client und Server aufzeichnen. Der Proxy-Log speichert somit alle Requests des Clients, der mit diesem Proxy verbunden ist. Proxies werden häufig in Unternehmen eingesetzt, in denen sich Mitarbeiter über den Proxy mit dem Internet verbinden. Dadurch können potentiell unsichere Seiten geblockt werden (Proxy Blacklist) oder in Echtzeit auf Abnormalitäten untersucht werden.

System-Log Der System-Log (syslog) zeichnet alle Ereignisse eines Betriebssystems auf und gibt Auskunft über Systemprozesse und diverse Treiber. Der Syslog informiert über potentielle Fehler und Warnungen im Zusammenhang mit dem Betriebssystem. Dadurch können auch potentielle Bedrohungen im Bereich von maliziösen Prozessen erkannt werden, die bestimmte Schwachpunkte im Betriebssystem auszunutzen um eine Code Execution zu erlangen.

3.1.2 Apache Webserver

Der Apache Webserver ist der am meist verbreitetste Webserver weltweit und wurde im Versuch zum Thema *Web-Service-Sicherheit* eingesetzt, um den Hackazon Webshop bereitzustellen und die Requests zu loggen. Im Apache Webserver gibt es vordefinierte Logformate, die sich leicht anpassen lassen. Das bekannteste Logformat ist das *Common Log File Format*, welches im folgenden Listing 1 aufgeführt ist.

Listing 1: Apache Common Log File Format

```
1 127.0.0.1 - - [01/01/2019:12:00:00 +0100] "GET /index.html?q=foo HTTP
    ↪/1.1" 200 2000
```

In diesem Event werden folgende Informationen gespeichert:

1. IP-Adresse/Hostname des Clients
2. Identität (RFC 1413)
3. Benutzername aus HTTP Authentifizierung
4. Timestamp mit Zeitzone
5. Methode, Pfad & Query, Protokoll
6. HTTP-Statuscode
7. Größe der Response (in Bytes)

Ein Bindestrich markiert, dass der entsprechende Eintrag nicht vorhanden ist oder nicht ermittelt werden konnte. Das folgende Listing 2 zeigt den Eintrag in der Apache Konfigurationsdatei, der das `common` Format spezifiziert (1. Zeile) und eine Datei für das `common` Format festlegt (2. Zeile). Mit der Direktive `CustomLog` werden Logdateien spezifiziert, in denen mit dem am Ende der Zeile angegebenen Format geloggt werden soll. Die Direktive `LogFormat` definiert das Logformat mit Platzhaltern, die wie in der Funktion `printf()` aus der Programmiersprache C dynamisch durch den jeweiligen Inhalt der Variable ersetzt werden.

Listing 2: Apache Konfiguration

```
1 LogFormat "%h %l %u %t \"%r\" %>s %b" common
2 CustomLog "/var/log/apache2/splunk/combined.log" common
```

3.1.3 Security Information and Event Management

Ein *Security Information and Event Management* (SIEM) beschreibt ein System, welches viele verschiedene Daten eines Unternehmensnetzwerks an einem zentralen Ort sammelt, speichert und effizient durchsuchbar macht. SIEMs werden in der Industrie eingesetzt und bieten eine umfassende Übersicht über alle Sicherheitsvorfälle in einem Unternehmen. Eine zentrale Datensenke hat den Vorteil, dass man alle Daten, die aus vielen unterschiedlichen Bereichen eines Unternehmens stammen, in einer zentralen Plattform betrachten und analysieren kann um somit gleichbleibende Muster, kleinere Abweichungen und große Abnormalitäten im Datenverkehr zu detektieren. SIEMs verarbeiten Ereignisse aus Netzwerken, Intrusion Detection Systemen, Firewalls, E-Mail Gateways, Proxies, Betriebssystemen, Sensoren und vielen weiteren Systemen.

Ein SIEM kann sowohl in Echtzeit als auch auf vergangene Ereignisse operieren. Die Daten werden direkt nach dem Empfang in einer Datenbank, die auch Index genannt wird, indiziert. Das SIEM kann die Daten direkt nach dem Empfang analysieren (Echtzeit) oder erst später, wenn genug Rechenressourcen zur Verfügung stehen und die Server generell weniger ausgelastet sind.

3.1.4 Log-Analyse

Bei der Analyse von Logdateien unterschiedet man zwei Strategien: rule-based & anomaly-based Detektion. [3]

Die rule-based Detektion verwendet statische Regeln, die *einmalig vor* der Analyse der Logs definiert werden. Typische Beispiele für statische Regeln sind reguläre Ausdrücke und Grenzwerte. Die rule-based Detektion kann als negatives oder positives Sicherheitsmodell klassifiziert werden.

Im negativen Sicherheitsmodell wird standardmäßig alles akzeptiert und die eingesetzten Regeln entscheiden über einzelne Verbote (Blacklist Ansatz). Ein Vorteil ist die Simplizität der Implementierung. Der Nachteil ist, dass die Detektion stark von den eingesetzten Regeln abhängt und nicht alle (insbesondere unbekannte) Angriffe erkannt werden.

Im positiven Sicherheitsmodell wird standardmäßig alles verboten und die eingesetzten Regeln entscheiden über erlaubte Aktionen (Whitelist Ansatz). Ein Vorteil ist die verbesserte Sicherheit, da maliziöse oder unbekannte Aktionen standardmäßig verboten sind. Der Nachteil ist die komplizierte und aufwendige Erstellung der Regeln.

Die anomaly-based Detektion verwendet dynamische Regeln die *während* einer Learning Phase erstellt werden. In dieser Learning Phase wird der gesamte Traffic analysiert. Somit können Abweichungen vom normalen Traffic identifiziert werden. Man kann entweder algorithmische Verfahren oder neue Techniken wie Machine Learning verwenden. Letzteres wird vor allem in größeren Unternehmen mit hohem Durchsatz verwendet, da es sich besonders gut auf den individuellen Datenverkehr im Unternehmen anpassen lässt.

3.2 Einführung: Splunk

Das SIEM Splunk konzentriert sich primär auf die Analyse von Maschinendaten und ermöglicht einen Einblick in die IT-Operations und IT-Security einer Unternehmensinfrastruktur. Die Splunk Plattform ist umfangreich und besteht aus mehreren Komponenten. Dieser Versuch konzentriert sich auf die Search Processing Language und die damit verbundene Datenanalyse und Detektion von Cross-Site-Scripting Angriffen.

3.2.1 Splunk Konzepte

Zunächst werden die wichtigsten Konzepte in Splunk kurz zusammengefasst. Diese Grundlagen werden für das weitere Verständnis der SPL vorausgesetzt.

Events Ein Splunk Event repräsentiert genau einen Logeintrag aus einer Logdatei. Das Event durchläuft mehrere Stufen beim Indizieren. Zunächst wird die Rohform des Events (Feld: `_raw`) aus der Logdatei extrahiert und an den Parser übergeben. Ein Event kann durch beliebige Zeichen und Patterns in der Logdatei begrenzt werden (z.B. Zeilenumbrüche). Der Parser extrahiert Felder aus dem Event (mehr dazu später). Jedes Event besitzt einen Timestamp (Feld: `_time`), der die zeitliche Einordnung dieses Events ermöglicht.

Das Event bildet die unterste logische Einheit in Splunk und repräsentiert genau ein Datenelement. Fortführende Funktionen, wie zum Beispiel Transaktionen, werden direkt auf Events angewendet. Somit können gleichartige Vorkommnisse aus mehreren Datenelementen konzeptionell zusammengefasst und logisch gruppiert werden. Zum Beispiel könnte gezielt die Session eines Benutzers herausgefiltert werden, obwohl diese durch die vielen Anfragen der anderen Benutzer dem Analysten nicht direkt ersichtlich ist.

Abbildung 1 zeigt ein Beispielevent aus dem Index `hackazon-splunkpra` in der Listen-Ansicht. Diese Ansicht zeigt im oberen Teil das rohe Event aus dem Webserver-Log an. Im unteren Teil werden alle geparsten Felder mit den Name-Wert Paaren angezeigt. Alternativ zur Listen-Ansicht kann auch die Rohtext-Ansicht oder Tabellen-Ansicht ausgewählt werden.

Abbildung 1: Beispielevent in Splunk

i		Uhrzeit	Ereignis
>	1	13.12.18 16:53:47,000	[13/Dec/2018:16:53:47 +0100],XBKAi8CoARgAAAPdxpAAAAAC,XBKAi8CoARgAAAPdxpAAAAAC,127.0.0.1,200,-,GET,"/swf/coupon_as.swf","",HTTP/1.1,549,4778,1894,2,+

bytes_received = 549 | bytes_send = 4778 | forensic_id = XBKAi8CoARgAAAPdxpAAAAAC
index = hackazon-splunkpra | keepalive = 2,+ | processing_time = 1894 | remote_host = 127.0.0.1
request_id = XBKAi8CoARgAAAPdxpAAAAAC | request_method = GET | request_protocol = HTTP/1.1
request_url = /swf/coupon_as.swf | request_user = - | response_status = 200

Sourcetypes Der Sourcetype eines Events (Feld: `sourcetype`) definiert das Layout bzw. die Struktur des Events. In Splunk gibt es einige bereits vordefinierte Formate, wie zum Beispiel das *Common Log File Format* aus Abschnitt 3.1.2. Es gibt allerdings auch die Möglichkeit eigene Sourcetypes zu erstellen. Diese können zum Beispiel durch Regex-Ausdrücke definiert werden. Jedes Event hat genau einen Sourcetype, da es nur ein Format annehmen kann. Der Sourcetype beeinflusst das Parsen der Events und die Extraktion der Felder.

Felder Ein Splunk Feld ist ein Schlüssel-Wert Paar und repräsentiert einen fest definierten Wert aus dem indizierten Event. Mit Feldern kann der Anwender unkomplizierte Filterungen anwenden und explizite Suchvorgaben definieren. Das Event in Listing 1 könnte zum Beispiel die Felder `ip=127.0.0.1` und `url=/index.html?q=foo` besitzen.

Direkt ersichtlich ist der Wert des Feldes, der aus dem Inhalt des Events extrahiert wird. Außerdem muss automatisiert erkannt werden, wo sich zum Beispiel die IP-Adresse und URL im rohen Event befinden und ob sie überhaupt existieren. Dieser Prozess wird in Splunk als *Feldextraktion* bezeichnet und entweder zum Zeitpunkt des Indizierens oder Suchprozesses durchgeführt.

In diesem Versuch wurde die dynamische Extraktion von Feldern bereits konfiguriert. Alle zur Verfügung stehenden Felder sind in der Tabelle 2 (Abschnitt 4.3) aufgelistet und direkt in Ihren Search Queries

nutzbar.

Ebenso existieren Standardfelder die automatisch beim Indizieren zu jedem Event hinzugefügt werden. In der folgenden Liste befindet sich eine Auswahl von den wichtigsten Standardfeldern:

_raw Roher Text des gesamten Events.
_time Timestamp des Events. Weitere Felder wie `date_hour`, `date_minute`, `date_month`, `date_wday`, ... stehen Ihnen ebenfalls zur Verfügung.
host IP-Adresse oder Hostname des Gerätes, dass dieses Event generiert hat.
index Name des Indexes in dem das Event gespeichert ist.
source Dateiname, Ordnername oder Name des Streams/Ports, in dem das Event auf dem Host gespeichert ist oder generiert wurde.
sourcetype Format des Events.

3.2.2 Search Processing Language

Splunk unterscheidet sich von anderen SIEMs durch die native Search Processing Language. Diese ermöglicht ein schnelles und effizientes Analysieren von Events. Die SPL implementiert zusätzlich tiefgreifende Analysetechniken, wie zum Beispiel die Datenkorrelation. Sie bildet das Fundament der Splunk Plattform, denn viele Features und Visualisierungen werden intern durch SPL Queries umgesetzt. *Splunk Dashboards* sind zum Beispiel regelmäßig ausgeführte SPL Queries, die Ergebnisse in Graphen visualisieren. *Splunk Alerts* sind gespeicherte SPL Queries, die entweder regelmäßig oder in Echtzeit ausgeführt werden und unter definierten Bedingungen eine Aktion ausführen, wie zum Beispiel das Versenden von Warnungen per E-Mail oder Push-Benachrichtigung. Ein Anwendungsbeispiel ist die direkte Benachrichtigung der zuständigen Incident Responder im Falle eines neuen Sicherheitsvorfalls.

Syntax Eine SPL Query besteht aus Befehlen und Argumenten. Die Ergebnisse aus einer Abfrage können in die Eingabe der nächsten Abfrage übergeben werden. Die Syntax der SPL erinnert an die Bash Kommandozeile mit dem Pipe-Operator:

```
search arguments0 | command1 arguments1 | command2 arguments2 | ...
```

Search Eine SPL Query startet immer mit dem `search` Befehl, der Events aus den Indizes lädt und basierend auf den übergebenen Argumenten filtert. Nachfolgend wird das Event geparsed und die Felder dynamisch extrahiert. Da die SPL Query immer mit dem `search` Befehl startet, kann man diesen am Anfang der Query optional weglassen bzw. er wird nach der Eingabe automatisch entfernt. Dies ist nur am Anfang der SPL Query erlaubt.

Nach dem `search` Befehl können beliebige Argumente übergeben werden. Die SPL Query `search ↪index="hackazon-combined"` selektiert zum Beispiel alle Events des Indexes `hackazon-combined`. Man beachte, dass hier das Standardfeld `index` zur Filterung verwendet wurde.

Die SPL Query `index="hackazon-combined" GET` ermöglicht das direkte Suchen nach Schlüsselwörtern. Allerdings liefert die SPL Query `index="hackazon-combined" request_method="GET"` präzisere Ergebnisse, da man explizit nach dem Feld filtert und nicht nur nach dem Schlüsselwort, dass zum Beispiel auch in anderen Teilen des Events vorkommen kann.

Der `search` Befehl bietet viele weitere Funktionen, wie zum Beispiel das logische Verknüpfungen von Bedingungen, Wildcards und Vergleichsoperatoren.

Ausführung einer SPL Query Bei der Ausführung des ersten `search` Befehls werden die gewünschten Events aus dem Index in eine dynamisch erstellte Tabelle geladen. Jede Zeile repräsentiert ein Event und die Spalten repräsentieren die einzelnen Felder. Die SPL Query operiert auf Basis dieser Tabelle und filtert zum Beispiel Events (löscht Zeilen) oder extrahiert neue Felder (fügt Spalten hinzu). Das Ergebnis wird abschließend als Tabelle oder Graph angezeigt. In der Search Pipeline wird die Tabelle an den jeweils nachfolgenden Befehl übergeben, der die Tabelle weiter modifizieren kann.

Besondere Befehle Die SPL besitzt viele unterschiedliche Befehle, von denen einige im Abschnitt 4.1 dokumentiert sind. Folgende Befehle beinhalten eigenständige Funktionen.

Der `eval` Befehl berechnet beliebige Ausdrücke und speichert das Ergebnis in einem neuen Feld. Mit einer großen Vielfalt an `eval` Funktionen können mathematische oder logische Operationen auf Zahlen oder Strings angewendet werden.

Die `stats`, `chart` und `timechart` Befehle gehören zur Gruppe der non-streaming (postpone) Befehle und besitzen ebenfalls eigene Funktionen. Diese werden verwendet um statistische Aussagen über alle selektierten Events der SPL Query zu treffen, wie zum Beispiel der Durchschnittswert oder meist vorkommende Wert eines Feldes.

Eine Besonderheit der non-streaming Befehle ist, dass die Ausgabetabelle nicht verändert wird, da sie die Events nicht direkt filtern oder modifizieren. Daher generieren die Befehle eine zusätzliche Ausgabe, die unter dem Reiter *Statistik* (tabellarische Ausgabe) oder *Visualisierung* (graphische Ausgabe) zu finden ist.

Die non-streaming Befehle werden meistens am Ende der SPL Query positioniert (postpone), da zunächst der Datensatz gefiltert wird und erst abschließend eine Statistik generiert wird. Die Ausgabe kann nicht weiter in der Search Pipeline übergeben bzw. gestreamt werden (non-streaming).

Subsearches Analog zur Bash Kommandozeile kann die SPL auch Subsearches verarbeiten. Diese führen eine unabhängige, eigenständige Suche durch und geben die finale Ausgabetabelle an den übergeordneten Befehl zurück. Die Subsearch wird in eckigen Klammern verpackt:

```
index="hackazon-combined" | search [ search index="hackazon-combined" | top 1
↪  request_method | fields request_method ]
```

Die SPL Query selektiert alle Events aus dem Index `hackazon-combined` mit der meist vorkommenden Request Methode in diesem Index. Die Subsearch gibt als Ergebnis `request_method="GET"` zurück, welches dann direkt mit dem `search` Befehl der übergeordneten SPL Query alle Events mit der Request Methode `GET` selektiert.

4 Referenz

4.1 Befehle

In diesem Abschnitt finden Sie Informationen zu den Splunk Befehlen, die Sie in diesem Versuch verwenden werden. Die Syntax der Befehle ist wie folgt strukturiert: Zunächst kommt der Befehlsname, gefolgt von einer Liste aus Parametern, wobei Parameter in eckigen Klammern optional sind.

4.1.1 search

Syntax: `search [<field><operator><value>]...`

Selektiert Events aus den Indizes, filtert die Ergebnisse eines vorherigen Befehls in der Search Pipeline und sucht nach bestimmten Werten in Feldern.

Als `<operator>` stehen `=, !=, <, <=, >, >=` bereit. Es können mehrere `<field><operator><value>` Paare durch `AND, OR, NOT` logisch verknüpft und durch Klammerung verschachtelt werden. Außerdem werden Wildcards, wie zum Beispiel `*`, unterstützt.

Beispiel: `search (index="hackazon*" OR index="proxy*")`

Das Beispiel selektiert Events aus *allen* Indizes, deren Namen mit „hackazon“ oder „proxy“ beginnen. Das Feld `index` ist ein Standardfeld. Die Klammerung ist hier optional, kann aber bei verschachtelten Konstruktionen mit `AND`, `OR` und `NOT` Sinn ergeben.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/Search`

4.1.2 head

Syntax: `head [<int>]`

Selektiert die `<int>` aktuellsten Events aus der Search Pipeline. Wird der Parameter `<int>` nicht angegeben, werden standardmäßig die 10 aktuellsten Events selektiert.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/head`

4.1.3 stats

Syntax: `stats <stats-func>(<field>) [AS <name>] [BY <field>]`

Berechnet Statistiken, wie zum Beispiel den Durchschnittswert, die Summe oder die Anzahl der Events und stellt unter anderem die folgenden Funktionen zur Verfügung:

avg(<field>) Berechnet den Durchschnittswert des Feldes `<field>`.

count(<field>) Berechnet die Anzahl der Vorkommnisse des Feldes `<field>`. Um einen spezifischen Feldwert festzulegen, muss `<field>` als `eval(<field>="value")` formatiert werden. Wird `count` ohne Parameter `(<field>)` angegeben, wird die Anzahl der an den `stats` Befehl übergebenen Events in der Search Pipeline berechnet.

dc(<field>) Berechnet die Anzahl der Events mit unterschiedlichen Werten des Feldes `<field>`.

mode(<field>) Berechnet den meist vorkommenden Wert des Feldes `<field>`.

sum(<field>) Berechnet die Summe der Werte des Feldes `<field>`.

Mit dem `AS` Parameter kann man die Ausgabe der `stats` Funktion in `<name>` umbenennen, die dann in der Tabellenspalte `<name>` angezeigt wird. Wenn der Befehl `stats` ohne `BY` Parameter verwendet wird, wird nur eine Zeile zurückgegeben. Dies ist die Aggregation über die gesamte eingehende Ergebnismenge. Wird der `BY` Parameter verwendet, wird für jeden unterschiedlichen Wert des im `BY` Parameter angegebenen Feldes `<field>` eine Zeile zurückgegeben.

Beispiel: `... | stats count(eval(status="200")) AS status200, count(eval(status="400")) AS status400`

Im Beispiel werden zwei `stats` Funktionen gleichzeitig verwendet. Es wird die Anzahl der Events mit dem Feldwert `status="200"` in der Spalte `status200`, sowie die Anzahl der Events mit dem Feldwert `status="400"` in der Spalte `status400` gegenübergestellt.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/stats`

4.1.4 top

Syntax: `top [<int>] <field>`

Berechnet die `<int>` meist vorkommenden Werte des Feldes `<field>`. Falls kein `<int>` Parameter übergeben wird, werden standardmäßig die 10 meist vorkommenden Werte des Feldes ermittelt.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/Top`

4.1.5 eval

Syntax: `eval <field>=<eval-func>(<param>[, <param>]...)[, <field>=<eval-func>(<param>[, <param>]...)]...`

Der `eval` Befehl berechnet eine „mathematical, string, oder boolean expression“ und speichert das Ergebnis im Feld `<field>` des entsprechenden Events. Falls das Feld `<field>` bereits existiert, wird das Feld überschrieben, andernfalls wird es neu erstellt.

Folgende `eval` Ausdrücke können verwendet werden:

round(<field>)	Gibt den gerundeten Wert des Feldes `<field>` zurück.
if(X,Y,Z)	Falls X zu `TRUE` evaluiert wird, wird der Wert Y in das Feld `<field>` geschrieben, andernfalls der Wert Z.
match(<field>, "<regex>")	Gibt `TRUE` zurück, falls `<field>` dem regulären Ausdruck `<regex>` entspricht. Kann zum Beispiel in Kombination mit `if(match(<field>, "<regex>"), "True", "False")` verwendet werden, wobei dem Rückgabewert dann entweder der String „True“ oder „False“ zugewiesen wird.
strftime(_time, "<format>")	Gibt das gerenderte Datum des Feldes `_time` im Format `<format>` zurück. Folgende Platzhalter stehen zur Verfügung: `%d` (Tag des Monats, 01-31), `%m` (Monat, 01-12) und `%Y` (Jahr, 2018).

Beispiel: `... | eval error = if(status == 200, "OK", "Problem")`

Erstellt in jedem Event ein neues Feld `error`. Falls das Feld `status` den Wert 200 besitzt, wird der Wert des Feldes `error` auf „OK“ gesetzt. Andernfalls wird der Wert auf „Problem“ gesetzt.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/eval`

Referenz Platzhalter: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/Commontimeformatvariables`

4.1.6 fields

Syntax: `fields <field>[, <field>]...`

Entfernt alle Felder der Events aus der Eingabemenge *mit Ausnahme* der explizit angegebenen Felder. Wird unter anderem zur Beschleunigung von Suchanfragen und zur Rückgabe eines Wertes in einer Subsearch verwendet.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/Fields`

4.1.7 timechart

Syntax: `timechart [span=<int><timescale>] <stats-func>(<field>) [BY <field>] [AS <name>]`

Erstellt ein Zeitdiagramm, dass eine statistische Aggregation auf ein Feld anwendet und abschließend ein Diagramm erzeugt, welches auf der X-Achse die Zeit darstellt und auf der Y-Achse die Aggregation.

Der `span` Parameter definiert das Zeitintervall auf der X-Achse. Als `<timescale>` stehen `s` (second), `m` (minute), `h` (hour), `d` (day) und `mon` (month) zur Verfügung.

Ähnlich wie beim `stats` Befehl kann man mit dem `AS` Parameter das Ergebnis der `<stats-func>` umbenennen, welches in diesem Fall die Beschriftung der Y-Achse ist.

Die Timechart wird gruppiert nach dem Wert des Feldes `<field>` des `BY` Parameters. Das bedeutet, dass für jeden unterschiedlichen Wert des Feldes ein Eintrag im Graph erzeugt wird (z.B. eine Linie in einem Liniendiagramm).

Beispiel: `... | timechart span=1m avg(CPU) AS "CPU Usage" BY host`

Hier wird der Durchschnittswert des Feldes `CPU` gruppiert nach dem Host berechnet. Auf der X-Achse wird ein minütliches Intervall gewählt und die Y-Achse ist mit „CPU Usage“ beschriftet.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/timechart`

4.1.8 regex

Syntax: `regex <field>=<regex-expression>`

Entfernt alle Events, in denen der Wert des Feldes `<field>` nicht dem regulären Ausdruck `<regex-expression>` entspricht. Splunk verwendet die standardisierte Pearl Compatible Regular Expression (PCRE). Ein vorgestelltes `(?i)` definiert einen case-insensitive regulären Ausdruck.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/Regex`

Interaktiver PCRE Tester: `https://regexr.com`

4.1.9 anomalousvalue

Syntax: `anomalousvalue [action=summary|filter] [pthresh=<threshold>] [<field>[, <field>]...]`

Generiert einen *anomaly score* für die angegebenen Felder eines Events relativ zu den Werten der Felder aller anderen Events. Dieser Wert indiziert inwiefern sich das Event von dem normalen Verhalten unterscheidet. Werden keine Felder angegeben, wird der anomaly score standardmäßig über alle Felder generiert.

Man unterscheidet zwischen einer numerischen und kategorischen Analyse. Die Anomalie von numerischen Feldern wird anhand der Standardabweichung vom Mittelwert bestimmt. Kategorische Felder werden mittels der Häufigkeit des Auftretens der entsprechenden Kategorie analysiert.

Die Aktion `summary` erzeugt eine allgemeine Statistik aller Anomalien in Form einer Tabelle. Jedes Feld wird individuell in einer Zeile der Tabelle repräsentiert und es stehen unter anderem folgende Informationen für jedes Feld zur Verfügung:

isNum Dieses Feld hat einen numerischen Wert

useCat Für dieses Feld wurde eine kategorische Anomalie Detektion durchgeführt

useNum Für dieses Feld wurde eine numerische Anomalie Detektion durchgeführt

catAnoFreq Die Auftrittshäufigkeit eines anormalen, kategorischen Wertes in Bezug auf die Gesamtanzahl aller kategorischen Werte des Feldes und den im Parameter `pthresh` angegebenen `<threshold>`.

numAnoFreq Die Auftrittshäufigkeit eines anormalen, numerischen Wertes in Bezug auf die Gesamtanzahl aller numerischen Werte des Feldes und den im Parameter `pthresh` angegebenen

`<threshold>`.

Die Aktion `filter` gibt ausschließlich die anormalen Ereignisse zurück, basierend auf den im Parameter `pthresh` angegebenen `<threshold>`.

Beispiel: `... | anomalousvalue action=filter pthresh=0.01`

Das Beispiel führt eine Anomalie Detektion auf allen Feldern der eingegebenen Events durch und gibt ausschließlich die anormalen Ereignisse zurück, die den angegebenen `<threshold>` erfüllen.

Referenz: `https://docs.splunk.com/Documentation/Splunk/7.2.1/SearchReference/Anomalousvalue`

4.2 Indizes

Tabelle 1 stellt die verfügbaren Indizes vor. Von den insgesamt 7 Indizes stammen 5 aus dem Hackazon-Webserver und 2 aus dem Proxy. Jeder Index enthält nur Events mit einem Sourcetype, sodass das Format aller Events im selben Index einheitlich ist.

Tabelle 1: Indizes

Index	Instanz	Sourcetype
hackazon-combined	Hackazon-Webserver	hackazon-combined
hackazon-splunkpra	Hackazon-Webserver	hackazon-splunkpra
hackazon-post	Hackazon-Webserver	hackazon-post
hackazon-forensic	Hackazon-Webserver	hackazon-forensic
hackazon-error	Hackazon-Webserver	hackazon-error
proxy-combined	Proxy	proxy-combined
proxy-squid	Proxy	proxy-squid

4.3 Formate der Sourcetypes

Tabelle 2 definiert die Formate der Sourcetypes. Die unten aufgeführten Felder wurden bereits extrahiert. Zusätzlich stehen Ihnen auch die Standardfelder zur Verfügung, die in dieser Tabelle nicht extra aufgeführt sind.

Tabelle 2: Formate der Sourcetypes

Sourcetype	Beispielereignis	Felder	
hackazon-combined	`127.0.0.1 - - [13/Dec/2018:16:53:47 +0100] "GET /swf/coupon_as.swf HTTP/1.1" 200 4427 " http://hackazon.netzlabor/cart/view" " Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0"`	• remote_host • remote_logname • remote_user • request_method • request_url	• request_protocol • response_status • response_size • referer • user_agent
hackazon-splunkpra	`[13/Dec/2018:16:53:47 +0100], XBKAi8CoARgAAAPdxpAAAAAC, XBKAi8CoARgAAAPdxpAAAAAC,127.0.0.1,200,-, GET,"/swf/coupon_as.swf","q=foo",HTTP/1.1, 549,4778,1894,2,+`	• request_id • forensic_id • remote_host • response_status • request_user • request_method • request_url	• request_query • request_protocol • bytes_received • bytes_send • processing_time • keepalive
hackazon-post	`[Thu Dec 13 16:53:43 2018] 127.0.0.1 "POST /cart/add HTTP/1.1" product_id=16&qty=3`	• remote_host • request_method • request_url	• request_protocol • request_data

hackazon-forensic	+ ↪XBKAi8CoARgAAAPdxpAAAAAC|GET ↪/swf/coupon_as.swf HTTP/1.1|User-Agent: ↪Mozilla/5.0 (X11; Ubuntu; Linux ↪x86_64; rv%3a60.0) Gecko/20100101 ↪Firefox/60.0|Accept:text/html, ↪application/xhtml+xml,application/xml ↪;q=0.9,*/*;q=0.8|Accept-Language:en- ↪GB,en;q=0.5|Accept-Encoding:gzip, ↪deflate|Referer:http%3a//hackazon. ↪netzlabor/cart/view|Cookie:PHPSESSID ↪=74aj1q6cjqshke1561354273h6; ↪visited_products=%252C16%252C18%252 ↪C101%252C81%252C|DNT:1|Host:hackazon. ↪netzlabor|Via:1.1 localhost (squid ↪/3.3.8)|X-Forwarded-For:127.0.0.1| ↪Cache-Control:max-age=259200| ↪Connection:keep-alive - ↪XBKAi8CoARgAAAPdxpAAAAAC	• forensic_id • request_method	• request_url • request_protocol
hackazon-error	[Thu Dec 13 ↪16:53:47.998172 2018] [↪XBKAi8CoARgAAAPdxpAAAAAC] [IVisUUmBjEw] [-] ↪ [debug] [989] [-] [mod_dumpost.c(165)] [↪-] [End brigade for request: GET /swf/coupon_as.swf HTTP/1.1, buffer: 0 bytes]	• request_id • connection_id • module_name • log_level • process_id	• thread_id • source_filename • source_line • error_code • error_message

proxy-combined	127.0.0.1 - - [13/Dec/2018:16:56:15 +0100] "CONNECT fonts.googleapis.com:443 HTTP/1.1" 200 3628 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" TCP_MISS:HIER_DIRECT	• remote_host • remote_user_ident • remote_user • request_method • request_url • request_protocol	• response_status • response_bytes • referer • user_agent • request_status • hierarchy_status
proxy-squid	1544716575.846 115150 127.0.0.1 TCP_MISS/200 3628 CONNECT fonts.googleapis.com:443 - HIER_DIRECT/2a00:1450:4001:81c::200a -	• timestamp • response_time • remote_host • request_status • response_status • response_bytes • request_method	• request_url • request_user • hierarchy_status • server_ip • mime_type • request_bytes (automatically calculated)

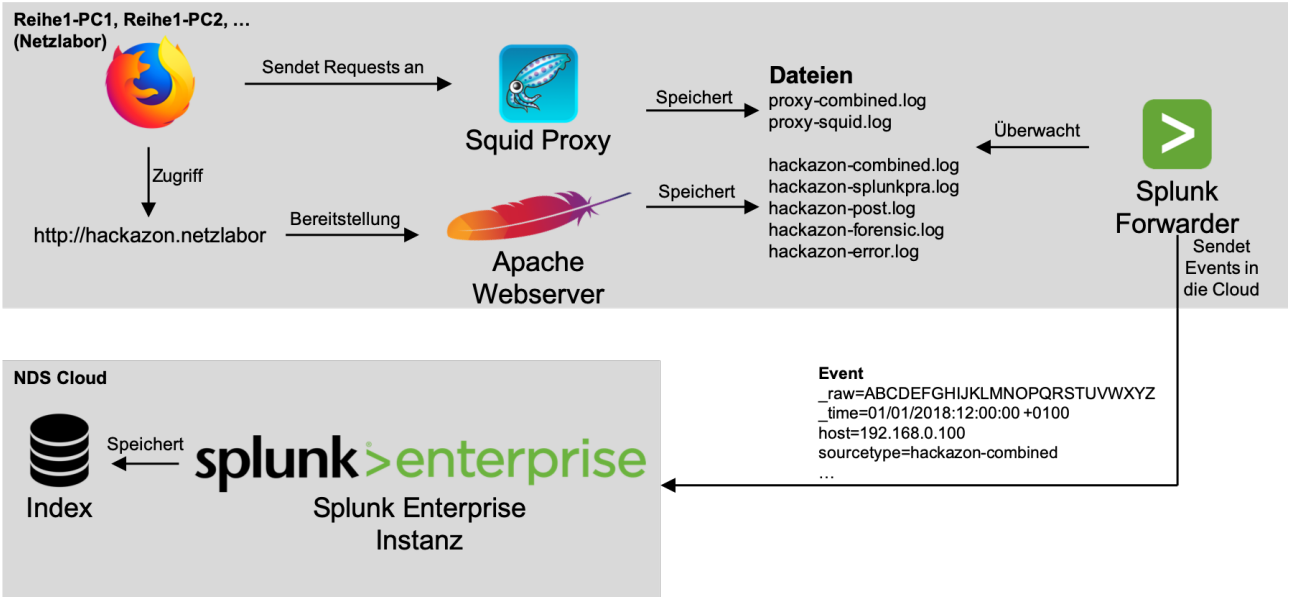
5 Kontrollfragen

1. Was ist das Ziel eines Security Incident and Event Managements und wie wird es erreicht?
2. Nennen Sie mindestens 5 Beispiele für Datenquellen, die indizierbare Events generieren.
3. Was ist der entscheidende Nachteil einer rule-based Detektion?
4. Was sind Splunk Alerts?
5. Was sind die Unterschiede zwischen einem *Host*, *Source* und *Sourcetype* in Splunk? Ordnen Sie die Beispiele `TCP:8080`, `linux_syslog` und `10.0.1.10` den korrekten Bezeichnungen zu.
6. Wie kann die Ausführung einer SPL Query in Splunk allgemein optimiert werden?
7. Worin unterscheiden sich non-streaming Befehle von normalen Befehlen in der SPL?
8. Mit welcher `eval` Funktion lässt sich aus dem `_time` Feld eines Events ein beliebig formatiertes Datum erstellen? Geben Sie einen Format String an, der das Datum im Format `yyyy-mm-dd` extrahiert.
9. Nennen Sie zwei Möglichkeiten wie man in Splunk ein Feld auf einen regulären Ausdruck testen kann. Tipp: Schauen Sie sich die Befehlsreferenz in Abschnitt 4.1 genauer an.
10. Geben Sie einen möglichst einfachen regulären Ausdruck (PCRE) an, der beliebige hexadezimale Zahlen mit einem führenden `0x` selektiert.

6 Versuchsaufbau

Während der Durchführung des Versuchs zum Thema *Web-Service-Sicherheit* im WS 18/19 wurden alle Events des Webservers (`http://hackazon.netzlabor`) und Proxies durch einen Splunk Forwarder an die zentrale Splunk Instanz in der NDS Cloud gesendet. Die Webserver und Proxies haben mehrere Logdateien mit unterschiedlichen Formaten generiert. Die Events wurden in den entsprechenden Indizes, welche in Tabelle 1 im Abschnitt 4.2 aufgelistet sind, gespeichert. Die Konfiguration des Versuchsaufbaus ist in der folgenden Grafik 2 veranschaulicht.

Abbildung 2: Konfiguration des Versuchsaufbaus



Öffnen Sie die URL `https://vanadium.cloud.nds.rub.de:8000/de-DE/` im Firefox. Es kann vorkommen, dass Firefox Sie vor einer unsicheren Verbindung warnt, da der Issuer nicht mit dem Zertifikat übereinstimmt. Klicken Sie auf den Button `Erweitert` und `Ausnahme hinzufügen`. Durch einen Klick auf `Ansehen` können Sie den SHA-256 Fingerabdruck mit dem Wert in Listing 10 vergleichen.

Falls beide Fingerabdrücke übereinstimmen, können Sie das Popup `schließen` und abschließend die „Sicherheits-Ausnahmeregel bestätigen“.

Listing 10: Zertifikat Fingerabdruck

```
1 1B:A5:B7:14:27:C4:D7:DA:2B:D8:A0:8C:41:EB:A5:28:EB:88:D6:A3:FA:20:70:
    ↪BE:57:2E:4D:B3:43:70:14:08
```

Sie können sich nun mit den Ihnen zugeteilten Zugangsdaten einloggen.

Benutzernamen: pc-1, pc-2, ..., pc-24
Passwörter: pc-1, pc-2, ..., pc-24

Nach dem Einloggen werden Sie von einer Einführungs-Tour begrüßt. Klicken Sie auf „Überspringen“. Sie befinden sich nun in der Suchansicht der *Search & Reporting* Applikation. Überprüfen Sie, dass die Zeitauswahl am rechten Rand des Suchfeldes auf **Alle Zeitpunkte** eingestellt ist! Setzen Sie den Modus auf „Ausführlich“, damit Sie auf die Felder der Events zugreifen können. Stellen Sie ebenfalls sicher, dass keine Beispielereignisse abgerufen werden.

Tipps & Tricks

- Zögern Sie nicht, SPL Queries einzugeben und auszuprobieren! Auch die Eingabe von falschen SPL Queries schadet der Plattform nicht und liefert zudem informative Fehlerbeschreibungen. Außerdem können Sie Befehle schrittweise erarbeiten und Teilergebnisse kontrollieren.
- Der Index `hackazon-splunkpra` ist als Standardindex voreingestellt. Wenn Sie keinen Index explizit in Ihrer Suche angeben, wird nur dieser Index durchsucht.
- Sie können jederzeit eine laufende Suchanfrage durch einen Klick auf das Stop-Symbol (Viereck) abbrechen.
- Der Suchverlauf unter dem Eingabefeld der Suche speichert alle von Ihnen durchgeführten SPL Queries. Diese werden automatisch am Ende des Versuches endgültig gelöscht.
- Mit dem folgenden Befehl aus dem Package `listings` können Sie SPL Queries in Latex (inline) einbinden: `\lstinline[basicstyle=\ttfamily,breaklines=true,postbreak={\mbox{$\hookrightarrow$}}]$Ihre SPL Query$`

Wichtiger Hinweis

Denken Sie daran, Ihre SPL Queries in textueller Form zu speichern und geben Sie diese in Ihrem Bericht ab! Sie können Ihre fertigen SPL Queries zum Beispiel in einer Textdatei auf einem USB-Stick speichern und später in Ihren Latex Bericht kopieren. Bitte geben Sie **nicht** für jede SPL Query einen Screenshot in Ihrem Bericht ab, sondern lediglich für die Aufgaben bei denen explizit ein Screenshot verlangt wird.

7 Versuchsdurchführung

7.1 Aufgabe: Grundlagen der Search Processing Language

Diese Aufgabe soll die Grundlagen der SPL vermitteln und dient als Vorbereitung für die Detektion der XSS Angriffe. Bitte beantworten Sie die folgenden Fragen und geben Sie Ihre SPL Queries im Bericht ab.

Starten Sie Ihre erste Splunk Suche indem Sie die SPL Query `index="proxy-combined" | head 10` in das Suchfeld eingeben. Beantworten Sie die folgenden Fragen:

1. Welche Events selektiert die SPL Query? Geben Sie eine allgemeine Antwort.
2. Welches Datum und welche Uhrzeit hat das aktuellste Event aus diesem Index?

3. Wie lautet die IP-Adresse des Hosts, der den Webshop besucht hat? Tipp: Beachten Sie auch die „interessanten Felder“ in der linken Seitenleiste (z.B. `remote_host`)!
4. Die IP-Adresse aus Frage 3 scheint auf den ersten Blick in jedem Event identisch zu sein. Geben Sie eine SPL Query an, die die Anzahl der verschiedenen Werte des Feldes `remote_host` in allen Events im Index `proxy-combined` anzeigt. Verifizieren Sie, dass die IP-Adresse überall gleich ist. Tipp: Verwenden Sie den `stats` Befehl mit einer passenden Funktion.
5. Wie lauten die 5 meist angefragten URL Pfade im Index `hackazon-splunkpra`?
6. Wie viele unterschiedliche URL Pfade wurden im Index `hackazon-splunkpra` angefragt?
7. Erstellen Sie **eine** SPL Query, die die Anzahl der `GET` sowie `POST` Requests im Index `hackazon-splunkpra` evaluiert. Wie viele `GET` sowie `POST` Requests gibt es? Tipp: Nutzen Sie das Beispiel aus Abschnitt 4.1.3.
8. Was ist der häufigste Response-Status im Index `hackazon-splunkpra`?
9. Wie lautet die Gesamtanzahl der gesendeten und empfangenen Bytes im Index `hackazon-splunkpra`? Tipp: Nutzen Sie eine passende `stats` Funktion.
10. Erstellen Sie eine Tabelle, die die Anzahl der Ereignisse im Index `hackazon-splunkpra` pro Tag darstellt. Die Tabelle soll aus den beiden Spalten `Tag` und `Anzahl` bestehen. Jede Zeile repräsentiert einen Tag mit einer nicht-leeren Anzahl von Ereignissen. Der Tag soll im Format `yyyy-mm-dd` extrahiert werden. An welchem Tag wurden wie viele Ereignisse indiziert? Tipp: Extrahieren Sie zunächst den Tag mit Hilfe einer `eval` Funktion und nutzen Sie dann eine `stats` ↪ Funktion mit dem `BY` Parameter. Hinweis: Der Versuch zum Thema *Web-Service-Sicherheit* wurde nur an zwei Tagen im Wintersemester 18/19 durchgeführt!
11. Erstellen Sie eine SPL Query, die den Tag mit den meisten Events im Index `hackazon-splunkpra` selektiert. Die SPL Query soll den Tag dynamisch ermitteln. Erstellen Sie dazu für jedes Event ein neues Feld `day` wie in Frage 10 und ermitteln Sie den am häufigst vorkommenden Feldwert. Fügen Sie `| fields day` am Ende der Query hinzu, um nur das Feld `day` in die Ausgabemenge zu übernehmen.
12. Selektieren Sie alle Events aus dem Index `hackazon-splunkpra`, die an dem Tag aus Frage 11 generiert wurden. Auch hier gilt, dass der Tag dynamisch ermittelt werden soll. Sie erhalten den Tag der zu selektierenden Events, indem Sie Ihre SPL Query aus Frage 11 in einer Subsearch verwenden. Nutzen Sie dann den zurückgelieferten Tag aus der Subsearch in einem `search` Befehl, um die Events zu filtern. Die Anzahl der selektieren Events sollte mit dem Ergebnis aus Frage 10 übereinstimmen. Tipp: Ein Beispiel zu Subsearches befindet sich in Abschnitt 3.2.2.
13. In Frage 12 haben Sie alle Events des Tages selektiert, an dem die meisten Events generiert wurden. Ergänzen Sie die SPL Query aus Frage 12 um die abschließende Generierung einer *Timechart* (`timechart`), die die *Anzahl* (`count`) der Events an diesem Tag in einem 10-Minuten *Intervall* (`span`) darstellt. Geben Sie einen Screenshot der *Line Chart* Visualisierung im Bericht an. Betrachten Sie Ihren Graphen und geben Sie die Uhrzeit des höchsten Peaks an.

7.2 Aufgabe: Rule-based XSS Detektion

In dieser Aufgabe werden Sie die XSS Angriffe auf den Hackazon-Webshop detektieren. Für die rule-based Detektion der XSS Payloads werden zunächst statische Regeln betrachtet. Beantworten Sie dazu die nachfolgenden Aufgaben:

14. Gegeben sei das folgende Konstrukt eines regulären Ausdrucks, der XSS Payloads basierend auf speziellen Schlüsselwörtern identifiziert: `(?i)(script|...)`. Um alternative Schreibweisen ebenfalls zu erkennen, ist der Regex case-insensitive. Ergänzen Sie den regulären Ausdruck um weitere Schlüsselwörter.
15. Erstellen Sie eine SPL Query, die XSS Payloads im Feld `request_query` im Index `hackazon-splunkpra` mit Hilfe Ihres regulären Ausdrucks aus Frage 14 detektiert. Wie viele potentielle Events mit

XSS Payloads können Sie detektieren?

16. Nennen Sie ein Beispiel eines Events mit maliziösem XSS Payload, dass von Ihrer SPL Query aus Frage 15 detektiert wurde. Erläutern Sie *kurz* den durchgeführten Angriff.

Um die Detektion zu verbessern, soll nun die Injection von beliebigen HTML Elementen detektiert werden. Dazu sei der folgende reguläre Ausdruck, der HTML Start-Tags *und* End-Tags klassifiziert, gegeben: `((3C)|<)((2F)|\/)?(.)+((3E)|>)`

Erklärung:

`((3C)|<)` Ein Tag wird immer mit `<` geöffnet.

`((2F)|\/)?` Ein End-Tag wird durch `/` geschlossen. Der Quantor `?` stellt sicher, dass der `/` entweder einmal (End-Tag) oder keinmal (Start-Tag) vorkommt.

`(.)+` Im Tag dürfen beliebige Zeichen vorkommen. Der Quantor `+` bewirkt, dass mindestens ein beliebiges Zeichen im Tag vorkommt.

`((3E)|>)` Ein Tag wird immer mit `>` geschlossen.

17. Erstellen Sie eine SPL Query, die XSS Payloads im Feld `request_query` im Index `hackazon-splunkpra` mit Hilfe des obigen regulären Ausdrucks detektiert. Wie viele potentielle Events mit XSS Payloads können Sie detektieren?
18. Erstellen Sie eine Grafik „XSS per Minute“, die die Anzahl der maliziösen (XSS Payload vorhanden) und nicht maliziösen (XSS Payload nicht vorhanden) Events gegenüberstellt. Beachten Sie dabei die folgenden Punkte und geben Sie einen Screenshot der Grafik in Ihrem Bericht ab!
    - Verwenden Sie Ihre SPL Query aus Frage 12 als Basis, um alle Events des Tages, an dem die meisten Events indiziert wurden, zu selektieren.
    - Betrachten Sie den Index `proxy-combined`.
    - Erstellen Sie ein neues Feld `ismalicious` das basierend auf dem Regex aus Frage 17 entweder den Wert „Malicious“ oder „Not Malicious“ besitzt. Falls das Feld `request_url` dem regulären Ausdruck entspricht, soll das Feld `ismalicious` somit den Wert „Malicious“ bekommen. Verwenden Sie dafür den `eval` Befehl mit den `if` und `match` Funktionen.
    - Abschließend soll eine *timechart* (`timechart`) mit einem Intervall (`span`) von 10 Minuten erstellt werden, die die Anzahl (`count`) der Events gruppiert nach (`by`) dem Wert des Feldes `ismalicious` ausgibt.
    - Wählen Sie eine *Line Chart* im *Gitterlayout* mit einer *unabhängigen Skalierung*.

7.3 BONUS: Anomaly-based XSS Detection

Abschließend wird die anomaly-based Detektion betrachtet. Splunk stellt dazu den `anomalousvalue` Befehl zur Verfügung. Aufgrund der beschränkten Datenmenge von drei Versuchsdurchführungen und identischen XSS Mustern derselben Schwachstelle können lediglich offensichtliche XSS Payloads durch die anormalen Payload-Längen identifiziert werden.

19. Erstellen Sie eine SPL Query, die alle Events aus dem Index `proxy-squid` mit den Domänen `hackazon.netzlabor` oder `freefileupload.netzlabor` und dem Protokoll `http://` selektiert. Nutzen Sie dafür folgendes Konstrukt: `index="..." AND (field="..." OR field="...")`. Tipp: Die Wildcard `*` könnte am Ende der Domänen nützlich sein.
20. Ergänzen Sie Ihre SPL Query aus Frage 19 um die abschließende Erzeugung einer tabellarischen Anomalie-Statistik (Zusammenfassung) über *alle* Felder aller zuvor selektierten Events.
21. An welchen Spalten der Anomalie-Statistik können Sie erkennen, ob eine kategorische oder numerische Anomalie-Detektion verwendet wurde und worin unterscheiden sie sich?
22. Welche Spalten informieren über die Anzahl der detektierten anormalen Ereignisse basierend auf den kategorischen und numerischen Feldern?
23. Betrachten Sie das Feld `response_status`. Bietet sich bei diesem Feld eine kategorische oder

numerische Anomalie-Detektion an? Begründen Sie kurz.

24. Welches Feld hat die höchste numerische Anomalie-Rate *und* reflektiert Informationen des Requests? Warum ist dieses Feld für die Anomalie-Detektion von XSS Payloads geeignet? Tipp: Tabellen lassen sich durch einen Klick auf die Spalte sortieren.
25. Modifizieren Sie Ihre SPL Query aus Frage 20 und selektieren Sie ausschließlich anormale Events basierend auf der Anomalie-Rate des von Ihnen gewählten Feldes aus Frage 24. Wählen Sie einen Threshold von `0.05`. Wie viele Events werden insgesamt selektiert?

Literatur

[1] Apache. Apache Access Log Format, 2019. `http://httpd.apache.org/docs/current/mod/mod_log_config.html#formats`.

[2] Apache. Apache Error Log Format, 2019. `https://httpd.apache.org/docs/2.4/en/mod/core.html#errorlogformat`.

[3] Roger Meyer. Detecting Attacks on Web Applications from Log Files, 2008. `https://www.sans.org/reading-room/whitepapers/logging/detecting-attacks-web-applications-log-files-2074`.

[4] Splunk. Splunk Dokumentation, 2019. `https://docs.splunk.com/Documentation`.

[5] Splunk. Splunk Quick Reference Guide, 2019. `https://www.splunk.com/content/dam/splunk2/pdfs/solution-guides/splunk-quick-reference-guide.pdf`.

[6] Wikipedia. Apache Common Log Format, 2019. `https://en.wikipedia.org/wiki/Common_Log_Format`.