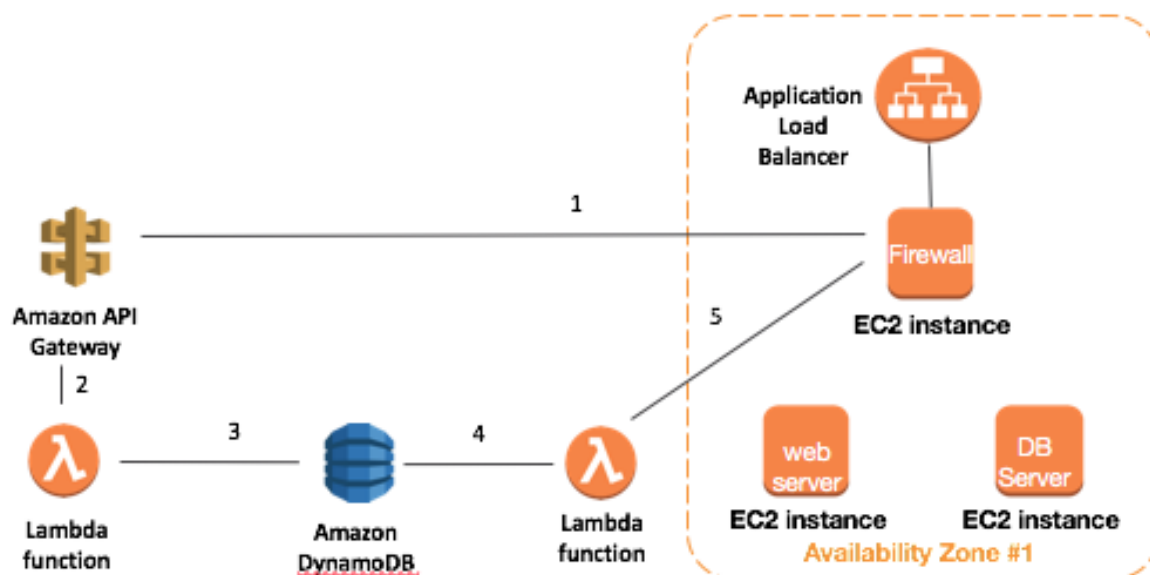


Overview

The components shown below are deployed in this demo

XFF UserID Demo Topology



Introduction

The demo performs the following actions.

- 1) In this demo we use the DVWA <http://www.dvwa.co.uk> to generate detectable threats between a browser and the web server. In this case we use a SQL Injection attack for demonstration purposes
- 2) The firewall will detect the SQL Injection threat and forward the log data to the AWS API Gateway via an HTTP log forwarding action.
- 3) The AWS API Gateway invokes a lambda function that queries the firewall for additional information based on the sessionId of the detected threat. The additional information includes the IP address in the X-Forwarded-For HTTP header. The additional data is written to a DynamoDB database.
- 4) A second Lambda function that is triggered from a Cloudwatch timer triggered event reads the current entries in the DynamoDB database and forwards them to the UserID database in the firewall.
- 5) The IP address from the X-Forwarded-For header is added to the "bad user" group and is then subject to a blocking policy in the firewall.

Initial Setup

1) Load The Application Load Balancer Certificate

The application load balancer will require a certificate. Access the AWS Certificate Manager and select "Load Certificate".

Import a certificate

Step 1: Import certificate

Step 2: Review and import

You can use AWS Certificate Manager certificates with other AWS Services.

Select certificate

Paste the PEM-encoded certificate body, private key, and certificate chain below. [Learn more.](#)

Certificate body*

```
-----BEGIN CERTIFICATE-----
MIICQDCCAZACCQCmgXE9NeMypjANBgkqhkiG9w0BAQsFADAWMRQwEgYDVQQDEwtk
ZWZ0ZXN0LmNvbTAeFw0xNzExMTAyMjIyMDhaFw0xODEyMTAyMjIyMDhaMBYx
FDAS
```

Certificate private key*

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA9Kn7Jnsf2y9qBfUpl/uauySROKxOVPrYHJ2Ag8gngQg/iNXQ
VSne0KGWnEuVdqSffhQZ/FIUaY9TOxbEak1hLs5n3ltF+oBjrm/upPLfawSY2PxY
```

Certificate chain

* Required

Paste in the text for the public and private key.

Make a note of the arn of the certificate as it is required by the Cloud Formation Template

The screenshot shows the AWS Certificate Manager console. At the top, there's a status bar with 'defest.com' and 'Issued'. Below that, the 'Status' section shows 'Status: Issued' and 'Detailed status: The cert was imported at 2017-11-14T21:21:42UTC'. A 'Reimport certificate' button is visible. The 'Details' section contains a table with the following information:

Type	Imported
In use?	No
Domain name	defest.com
Number of additional names	0
Identifier	6d5b962d-296a-42f6-acac-574e1099f196
Serial number	a6:81:71:3d:35:a3:18:a6
Imported at	2017-11-14T21:21:42UTC
Not after	2018-11-10T22:22:08UTC
Expires in	361 Days
Public key info	RSA 2048-bit
Signature algorithm	SHA256WITHRSA
ARN	arn:aws:acm:us-west-2:106608901653:certificate/6d5b962d-296a-42f6-acac-574e1099f196

The ARN is highlighted with a red box. Below the details, there's a 'Tags' section with an 'Edit' button and a text input field. At the bottom, there's a status bar with 'defest.com', 'Issued', 'Imported', and 'No'. A pagination link at the bottom right says '< < Viewing 1 to 2 of 2 certificates > >'.

2) Create the Bootstrap Buckets

More documentation required here but we can import from the existing Wordpress Demo.

3) Deploy the Cloud Formation Template

Create stack

[Select Template](#)

Specify Details

[Options](#)

[Review](#)

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name

Parameters

BootstrapBucketName	<input type="text" value="xff-fw-bucket"/>	Bucket name for FW bootstrap configuration
CertificateARN	<input type="text" value="arn:aws:acm::us-west-2:106808901653:cert"/>	Enter arn for the certificate
ELBName	<input type="text" value="public-elb"/>	Enter the name of the External Classic Load Balancer
LambdaBucketName	<input type="text" value="xff-lambda-bucket"/>	Bucket name for FW bootstrap configuration
NumberOfAZs	<input type="text" value="2"/>	Total Number of AZs which will be used in this deployment (Default 2)
ServerKeyName	<input type="text" value="AWS-Key-Pair"/>	Name of an existing EC2 KeyPair to enable SSH access to the FW (Hint: You MUST have its private key)
VpcAzs	<input type="text" value="us-west-2a x us-west-2b x"/>	Enter the list of Availability Zones (Based on Number of AZs above)

Check that the template deployment completes and make a note of the Wordpress URL which is the DNS name of the Application Load Balancer and the firewall management IP XFF-Demo-Stack

Stack name: XFF-Demo-Stack

Stack ID: arn:aws:cloudformation:us-west-2:106808901653:stack/XFF-Demo-Stack/27949c10-c98a-11e7-9bcb-50a68a2012f2

Status: **CREATE_COMPLETE**

Status reason:

Termination protection: Disabled

IAM Role:

Description: Install Web Server, Firewall and Application LB to Demonstrate XFF Header IP extraction for use in policy

▼ Outputs

Key	Value	Description
WordpressURL	https://XffALB-272684332.us-west-2.elb.amazonaws.com/dvwa/setup.php	Web server
FirewallManagementURL	https://34.208.95.62	VM-Series management interface URL

► Resources

▼ Events

2017-11-14	Status	Type	Logical ID	Status reason
22:26:44 UTC+0000	CREATE_COMPLETE	AWS::CloudFormation::Stack	XFF-Demo-Stack	

4) Create the API Gateway
Go to the AWS console and select API gateway

Create new API

In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ New API
 ☐ Import from Swagger
 ☐ Example API

Settings

Choose a friendly name and description for your API.

API name*

Description

Endpoint Type

* Required

Select Actions and create a POST method

[Amazon API Gateway](#)
[APIs](#) > [XFF API \(bw301q3p0i\)](#) > [Resources](#) > [/\(ckx6sh1sva\)](#) > [POST](#)

Resources

Actions

POST

/ - POST - Setup

Choose the integration point for your new method.

Integration type
☒ Lambda Function ⓘ
 ☐ HTTP ⓘ
 ☐ Mock ⓘ
 ☐ AWS Service ⓘ

Use Lambda Proxy integration ☐ ⓘ

Lambda Region

Lambda Function ⓘ

Add the lambda region and then the function.

The Lambda function will have the name {Stack Name}-GetXFFHeaderLambda-*

Next Deploy the gateway to a stage

After deployment copy the URL from the summary screen

Select Actions and then Deploy

Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage

[New Stage]

Stage name*

LATEST

Stage description

Deployment description

Cancel

Deploy

> Stages > LATEST

LATEST Stage Editor

Invoke URL: <https://37uhcyum4f.execute-api.us-west-2.amazonaws.com/LATEST>

Settings

Stage Variables

SDK Generation

Export

Deployment History

Documentation History

Configure the metering and caching settings for the **LATEST** stage.

Cache Settings

Enable API cache

CloudWatch Settings

Enable CloudWatch Logs

Log level

INFO

Log full requests/responses data

Login to the firewall with credentials Password for the firewall is admin/Pal0Alt0

Extract the DNS name from the URL and modify the HTTP log forwarding values

HTTP Server Profile

Name

☐ Tag Registration
The server(s) should have User-ID agent running in order for tag registration to work

Servers Payload Format

Name	Address	Protocol	Port	HTTP Method	Username	Password
<input checked="" type="checkbox"/> AWS	dmjhoy2vhk.... api.us-west- 2.amazonaw...	HTTPS	443	POST		

Test Server Connection

Modify the “URI Format” under “Payload Format”

HTTP Server Profile

Name

☐ Tag Registration
The server(s) should have User-ID agent running in order for tag registration to work

Servers Payload Format

Log Type	Format
Config	Default
System	Default
Threat	send_threat_session_id
Traffic	Default
URL	Default
Data	Default
WildFire	Default
Tunnel	Default
Authentication	Default
User-ID	Default
HIP Match	Default

Pre-defined Formats

Name

send_threat_session_id

URI Format

/LATEST

HTTP Headers

Headers	Value
content-type	application/json

+

 Add

−

 Delete

Parameters

Params	Value
--------	-------

+

 Add

−

 Delete

Payload

```
{ "operation": "register", "sessionid": $sessionid, "natsrcport": $natsport, "rxtime": "$receive_time" }
```

Send Test Log

OK

Cancel

Once configured send a test log message

Pre-defined Formats

Name

send_threat_session_id

URI Format

/LATEST/xffdbtable

HTTP Headers

Headers	Value
---------	-------

+

 Add

−

 Delete

Parameters

Params	Value
--------	-------

+

 Add

−

 Delete

Payload

```
{ "operation": "register", "sessionid": $sessionid, "natsrcport": $natsport, "rxtime": "$receive_time" }
```

Test Results

Successfully sent: { "operation": "register", "sessionid": 0, "natsrcport": 12345, "rxtime": "2017/11/14 14:54:53" }

Close

Send Test Log

OK

Cancel

Test the deployment

Open a browser and browse to the URL from the output of the cloud formation template
Setup the DVWA database

XFF-Demo-Stack

Stack name: XFF-Demo-Stack
Stack ID: arn:aws:cloudformation:us-west-2:106808901653:stack/XFF-Demo-Stack/27949c10-c98a-11e7-9bcb-50a68a2012f2
Status: CREATE_COMPLETE
Status reason:
Termination protection: Disabled
IAM Role:
Description Install Web Server, Firewall and Application LB to Demonstrate XFF Header IP extraction for use in policy

▼ Outputs

Key	Value	Description
WordpressURL	https://XffALB-272684332.us-west-2.elb.amazonaws.com/dvwa/setup.php	Web server
FirewallManagementURL	https://34.208.95.62	VM-Series management interface URL

Setup Check

Operating system: ***nix**
Backend database: **MySQL**
PHP version: **7.0.22-0ubuntu0.16.04.1**

Web Server SERVER_NAME: **xffalb-272684332.us-west-2.elb.amazonaws.com**

PHP function display_errors: **Disabled**
PHP function safe_mode: **Disabled**
PHP function allow_url_include: **Enabled**
PHP function allow_url_fopen: **Enabled**
PHP function magic_quotes_gpc: **Disabled**
PHP module gd: **Installed**
PHP module mysql: **Installed**
PHP module pdo_mysql: **Installed**

MySQL username: **root**
MySQL password: *********
MySQL database: **dvwa**
MySQL host: **127.0.0.1**

reCAPTCHA key: **Missing**

[User: root] Writable folder /var/www/html/dvwa/hackable/uploads/: **Yes**
[User: root] Writable file /var/www/html/dvwa/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: **Yes**

Status in red, indicate there will be an issue when trying to complete some modules.

Create / Reset Database

Login to the application using admin / password credentials

Lower the security level of the application to "Low"

Welcome to Damn Vulnerable Web App

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is goal is to be an aid for security professionals to test their skills and tools in a legal e developers better understand the processes of securing web applications and to aid learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities, difficulty**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every mod selecting any module and working up to reach the highest level they can before mo is not a fixed object to complete a module; however users should feel that they have system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with t intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be ena increase the difficulty. This will demonstrate how adding another layer of security m actions. Note, there are also various public methods at bypassing these protections extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & There are also additional links for further background reading, which relates to that :

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your h html folder or any Internet facing servers**, as they will be compromised. It is reco machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. In can downloading and install [XAMPP](#) for the web server and database.

Next generate Threats using the SQL Injection tab

Details of SQL injection attacks that can be performed can be found here

https://www.computersecuritystudent.com/SECURITY_TOOLS/DVWA/DVWAv107/lesson6/index.html

For simplicity use `%'` or `'0'='0`

The firewall should block the attack and the browser will return a 502 bad Gateway error.

Check the firewall for threat logs

Dashboard

ACC



Monitor

Policies

Objects

Network

Device

	Receive Time	Type	Name	From Zone	To Zone	Attacker	Attacker Name	Victim	To Port	Ap
	11/14 15:10:37	vulnerability	HTTP SQL Injection Attempt	UNTRUST	TRUST	10.0.0.62		10.0.0.100	80	we
	11/14 15:10:23	vulnerability	HTTP SQL Injection Attempt	UNTRUST	TRUST	10.0.0.62		10.0.0.100	80	we

Check the cloudwatch logs.

If the threat log has been forwarded and processed logs will appear in cloudwatch under the log group corresponding to the lambda function

CloudWatch

Dashboards

Alarms

Events

Rules

Event Buses

Logs

Metrics

Favorites

CloudWatch > Log Groups > /aws/lambda/XFF-Demo-Stack-GetXFFHeader.Lambda-14AAPMZPV800 > 2017/11/14/[SLATEST]0eca5907bd94cb49a69515e4970eddb8

Expand all Row Text

Filter events

Time (UTC +00:00)	Message
2017-11-14 23:25:49	writing to log
2017-11-14 23:25:49	(Line is: ', 86.143.31.2,1510701949')
2017-11-14 23:25:49	Writing to S3
2017-11-14 23:25:50	An error occurred (AccessDenied) when calling the PutObject operation: Access Denied: ClientError Traceback (most recent call last): File "/var/task/extract_xff.py", line 195, in extract_xff_lambda_handler w
2017-11-14 23:25:50	END RequestId: 20821216-c993-11e7-a774-39941090c580
2017-11-14 23:25:50	REPORT RequestId: 20821216-c993-11e7-a774-39941090c580 Duration: 10205.03 ms Billed Duration: 10300 ms Memory Size: 128 MB Max Memory Used: 53 MB
2017-11-14 23:25:50	START RequestId: 26b63492-c993-11e7-a149-bb8e06fda97 Version: SLATEST
2017-11-14 23:25:50	(Session id is: ', 448)
2017-11-14 23:25:50	(NAT SPORT is: ', 1787)
2017-11-14 23:25:50	(Receive time is: ', u'2017/11/14 15:25:49')
2017-11-14 23:25:50	The command to extract jobid is https://34.208.95.62/api/?type=log&log-type=url&key=LUFRT1KaTRJbGYSUJuTVRaN1M2UzRLZ2NpaHB4S1E9T3FZUXRTQ3FNsnoyenK2S1NuMTdGOEdEVRhVfYxd0x
2017-11-14 23:25:50	(Job id is: ', '21')
2017-11-14 23:25:50	Sleeping for 2 second...
2017-11-14 23:25:53	The command to extract XFF is https://34.208.95.62/api/?type=log&log-type=url&key=LUFRT1KaTRJbGYSUJuTVRaN1M2UzRLZ2NpaHB4S1E9T3FZUXRTQ3FNsnoyenK2S1NuMTdGOEdEVRhVfYxd0x
2017-11-14 23:25:53	The command to extract jobid is https://34.208.95.62/api/?type=log&log-type=url&key=LUFRT1KaTRJbGYSUJuTVRaN1M2UzRLZ2NpaHB4S1E9T3FZUXRTQ3FNsnoyenK2S1NuMTdGOEdEVRhVfYxd0x
2017-11-14 23:25:53	The command to extract XFF is https://34.208.95.62/api/?type=log&log-type=url&key=LUFRT1KaTRJbGYSUJuTVRaN1M2UzRLZ2NpaHB4S1E9T3FZUXRTQ3FNsnoyenK2S1NuMTdGOEdEVRhVfYxd0x
2017-11-14 23:25:53	The command to extract jobid is https://34.208.95.62/api/?type=log&log-type=url&key=LUFRT1KaTRJbGYSUJuTVRaN1M2UzRLZ2NpaHB4S1E9T3FZUXRTQ3FNsnoyenK2S1NuMTdGOEdEVRhVfYxd0x
2017-11-14 23:25:53	(Job id is: ', '22')
2017-11-14 23:25:53	Sleeping for 2 second...
2017-11-14 23:25:56	The command to extract XFF is https://34.208.95.62/api/?type=log&log-type=url&key=LUFRT1KaTRJbGYSUJuTVRaN1M2UzRLZ2NpaHB4S1E9T3FZUXRTQ3FNsnoyenK2S1NuMTdGOEdEVRhVfYxd0x
2017-11-14 23:25:56	The command to extract jobid is https://34.208.95.62/api/?type=log&log-type=url&key=LUFRT1KaTRJbGYSUJuTVRaN1M2UzRLZ2NpaHB4S1E9T3FZUXRTQ3FNsnoyenK2S1NuMTdGOEdEVRhVfYxd0x

Checking DynamoDB should show the IP address in the database

Create table Delete table

Filter by table name

Name

DomainNames

GPCClientIP

xffdbtable

xffdbtable Close

Overview Items Metrics Alarms Capacity Indexes Triggers Access control

Create item Actions

Scan: [Table] xffdbtable: username

Scan [Table] xffdbtable: username

Add filter

Start search

	username	creationdate	expirationdate (TTL)	ipaddress	notes
	86.143.31.2	1510701963	1510788363	86.143.31.2	uidsnt

If the notes column indicates uidsnt then the username IP has been sent to the firewall

```
admin@aws-test-drive-fw> show user ip-user-mapping all
```

IP	Vsys	From	User	IdleTimeout(s)	MaxTimeout(s)
86.143.31.2	vsys1	XMLAPI	86.143.31.2	711	711
Total: 1 users					

```
admin@aws-test-drive-fw> █
```

You should now receive a block page from the original URL

