EXOCET
SECURITY

Spikedata.co.za Security Assessment Report

Spike Security Assessment Report
Version 1.0
10/05/2018
Sherwyn Moodley
Targets: api-v5.cloudbank.co.za/beta/
35.177.252.164

Document Properties

Title	Version	Author	Assessor	Reviewed by	Date
Spike Security Assessment Report	1.0	Sherwyn Moodley	Sherwyn Moodley	Christian Jeurissen	10/05/2018

1. Executive Summary

Exocet Security was commissioned by Spike to conduct a Security Assessment on their beta banking API which is in development.
The targets given for the assessment were: api-v5.cloudbank.co.za/beta/ and 35.177.252.164
A number of tests were conducted over the period of 10 working days utilizing a number of solutions including vulnerability scanners, penetration tests and hacking methodologies such as cross site scripting, sql injection and brute forcing.
This was done to emulate an exhaustive real world attack and develop a clear picture of Spike's Security Outlook.
Due to Spike's architecture there are a limited amount of entrance points to the servers. This decreases the possibility of vulnerabilities and therefore exploits. The only method of communicating with the api server is through the front web server.

1.1 Methodology

Exocet Security utilised the following methodology in this security assessment:
An initial black box security assessment was conducted on api-v5.spikedata.co.za with no prior knowledge of the architecture or software. This consisted of the following tests:

- Vulnerability scans
  - a collection of industry standard vulnerability scanners were utilised to look for known risks and ranked vulnerabilities against the CVE database.
- Penetration tests
  - based on the outcomes of the vulnerability scans findings above we conducted manual tests to uncover security flaws.

Thereafter we received additional information on the architecture of the application from Spike Data and performed additional vulnerability scans on an internal server 35.177.252.164. We also discussed various security recommendations with Spike Data based on this architecture.
This report details the results of these tests, summarizes findings and suggested remedial action as well as various recommended security best practices

1.2 Project Objectives

Spike requires a full security assessment to display to all stakeholders and interested parties their commitment to ensuring a safe and secure environment with a well developed security posture.

1.3 Timeline

Project Start	30/04/2017

Vulnerability Check	30/04/2017
Prelim Report	02/05/2018
Additional Vulnerability Scans	03/05/2018
Penetration Testing Commences	04/05/2018
Client Report Engagement	07/05/2018
Final Report	10/05/2018

2. Vulnerability Scans

The following vulnerability scanners were utilised to offer as complete a picture as possible:

Scanner	Description
Nexpose	Developed by Nessus, one of the most commonly used scanners
OpenVas	OpenVAS is a free scanner in the Kali suite, also very commonly used.
Nmap	Nmap is a network scanning and host detection tool
Shodan.io	An internet analysis tool which allows for digital footprint scanning
OWASP ZAP	**Zed** Attack Proxy is an open-source web application security scanner.

2.1 Findings for api-v5.spikedata.co.za

Scanner	Summary
Nexpose	No potential vulnerabilities found
OpenVas	15 results all rated CVSS 0.0 except 1 low at 2.6
Nmap	Report showed AWS with 443 open
Shodan.io	5 instances of https://api-v5.spikedata.co.za
OWASP ZAP	Medium: 1 Low: 4

All Vulnerabilities found were labelled low or information. There are no High or Medium risk vulnerabilities detected by any of the scanning software.

The only port open on the website is the HTTPS port 443.

2.2 Findings for 35.177.252.164

Open Ports

22 - SSH and 80 - HTTP (website returns forbidden)

No Vulnerabilities Found

3. Penetration Tests

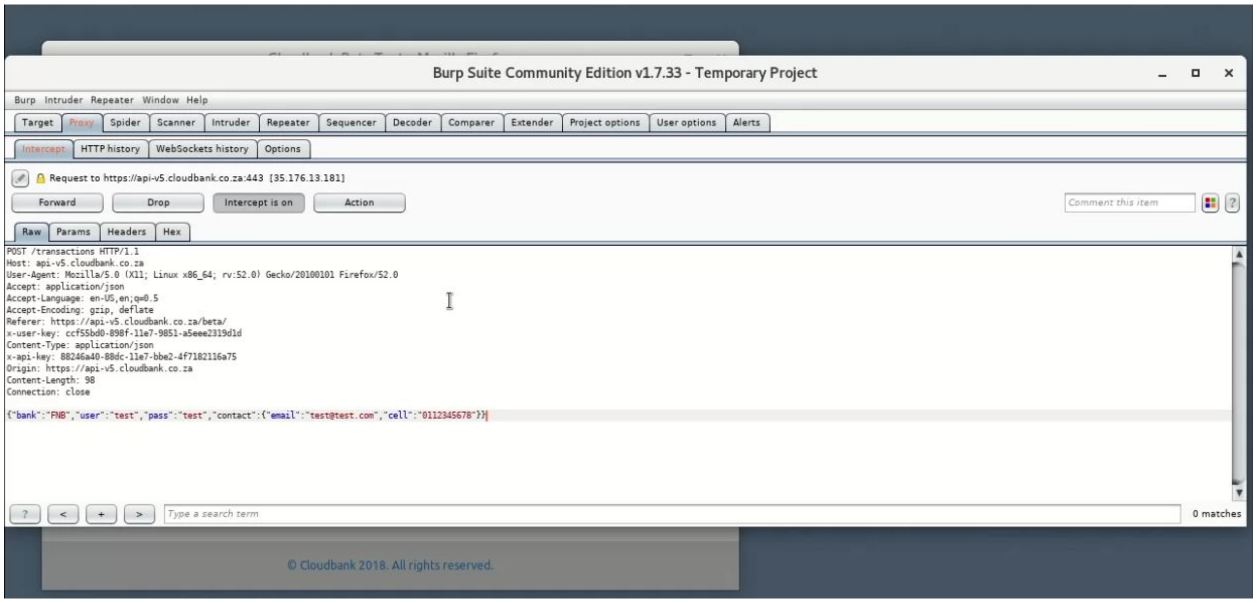
The penetration tests focused on https://api-v5.spikedata.co.za/beta/ – an html5/javascript app which allows a user to input their internet banking login credentials and then retrieve their 90-day transaction history from the Spike API (also running on https://api-v5.spikedata.co.za).

3.1 Fuzzing Communication

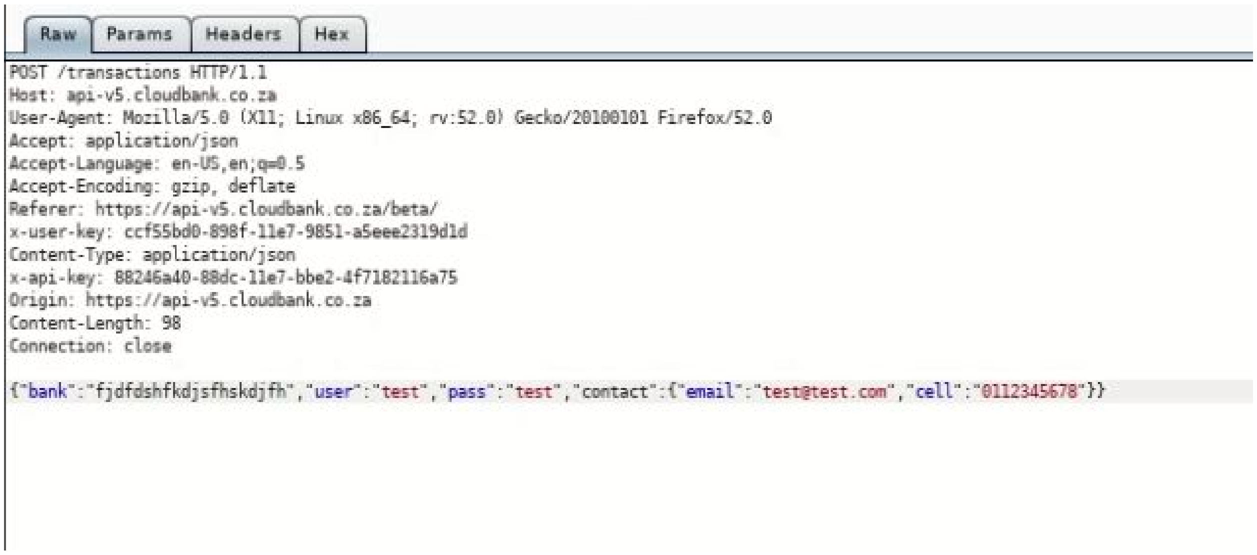
As shown in the images below, communication between the client web browser and the webserver is captured and modified. The string entry for bank is changed, first to a random string and then to a proposed java code injection.

Neither case yielded any results for the would be attacker. The server does not recognize the random string as a bank and the server does not process the java code execution.

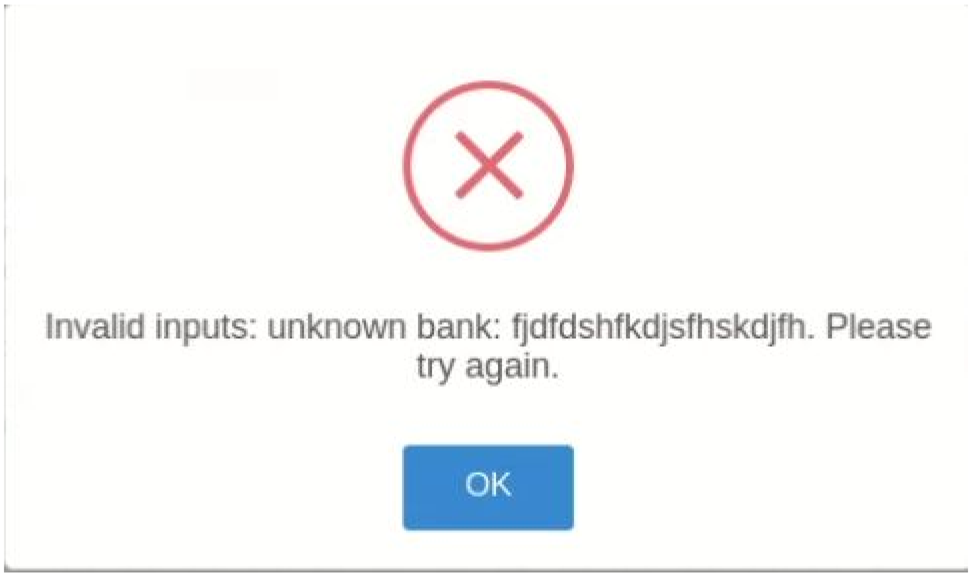
.


Burp Suite Community Edition v1.7.33 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Request to https://api-v5.cloudbank.co.za:443 [35.176.13.181]
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
Type a search term
0 matches
© Cloudbank 2018. All rights reserved.

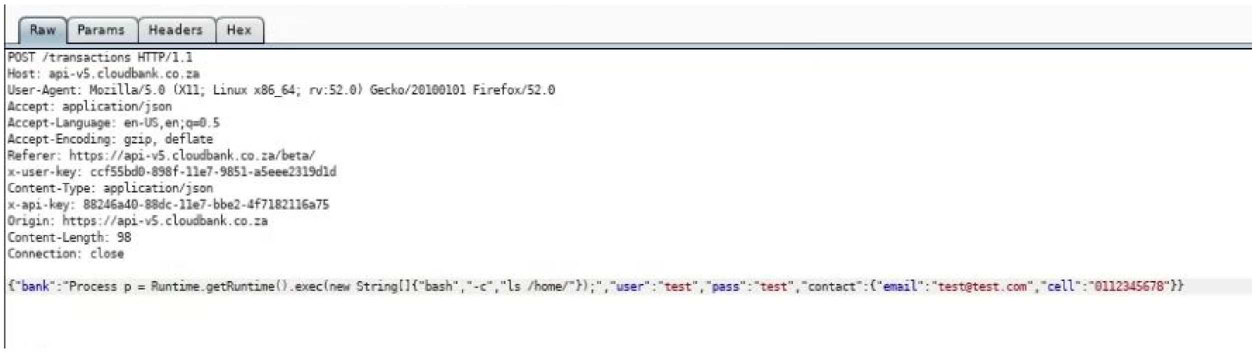
```
POST /transactions HTTP/1.1
Host: api-v5.cloudbank.co.za
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://api-v5.cloudbank.co.za/beta/
x-user-key: ccf55bd0-898f-11e7-9851-a5eee2319d1d
Content-Type: application/json
x-api-key: 88246a40-88dc-11e7-bbe2-4f7182116a75
Origin: https://api-v5.cloudbank.co.za
Content-Length: 98
Connection: close

{"bank":"FNB","user":"test","pass":"test","contact":{"email":"test@test.com","cell":"0112345678"}}
```



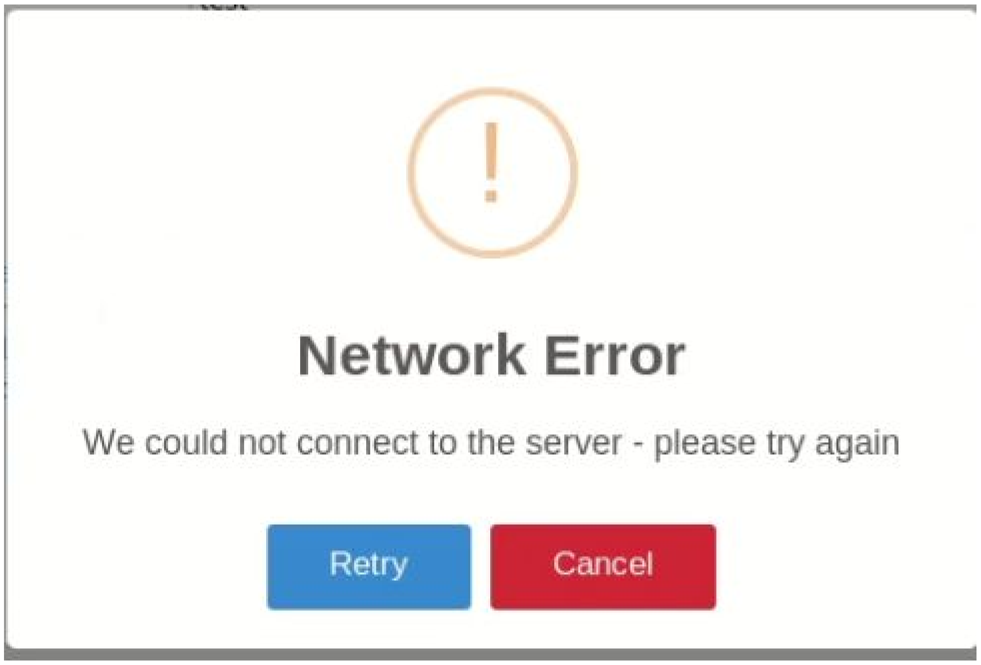
```
POST /transactions HTTP/1.1
Host: api-v5.cloudbank.co.za
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://api-v5.cloudbank.co.za/beta/
x-user-key: ccf55bd0-898f-11e7-9851-a5eee2319d1d
Content-Type: application/json
x-api-key: 88246a40-88dc-11e7-bbe2-4f7182116a75
Origin: https://api-v5.cloudbank.co.za
Content-Length: 98
Connection: close

{"bank":"fjdfdshfkdjsfhskdjfh","user":"test","pass":"test","contact":{"email":"test@test.com","cell":"0112345678"}}
```



Invalid inputs: unknown bank: fjdfdshfkdjsfhskdjfh. Please try again.



```
POST /transactions HTTP/1.1
Host: api-v5.cloudbank.co.za
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://api-v5.cloudbank.co.za/beta/
x-user-key: ccf55bd0-898f-11e7-9851-a5eee2319d1d
Content-Type: application/json
x-api-key: 88246a40-88dc-11e7-bbe2-4f7182116a75
Origin: https://api-v5.cloudbank.co.za
Content-Length: 98
Connection: close

{"bank":"Process p = Runtime.getRuntime().exec(new String[]{"bash","-c","ls /home/"});","user":"test","pass":"test","contact":{"email":"test@test.com","cell":"0112345678"}}
```



Network Error

We could not connect to the server - please try again

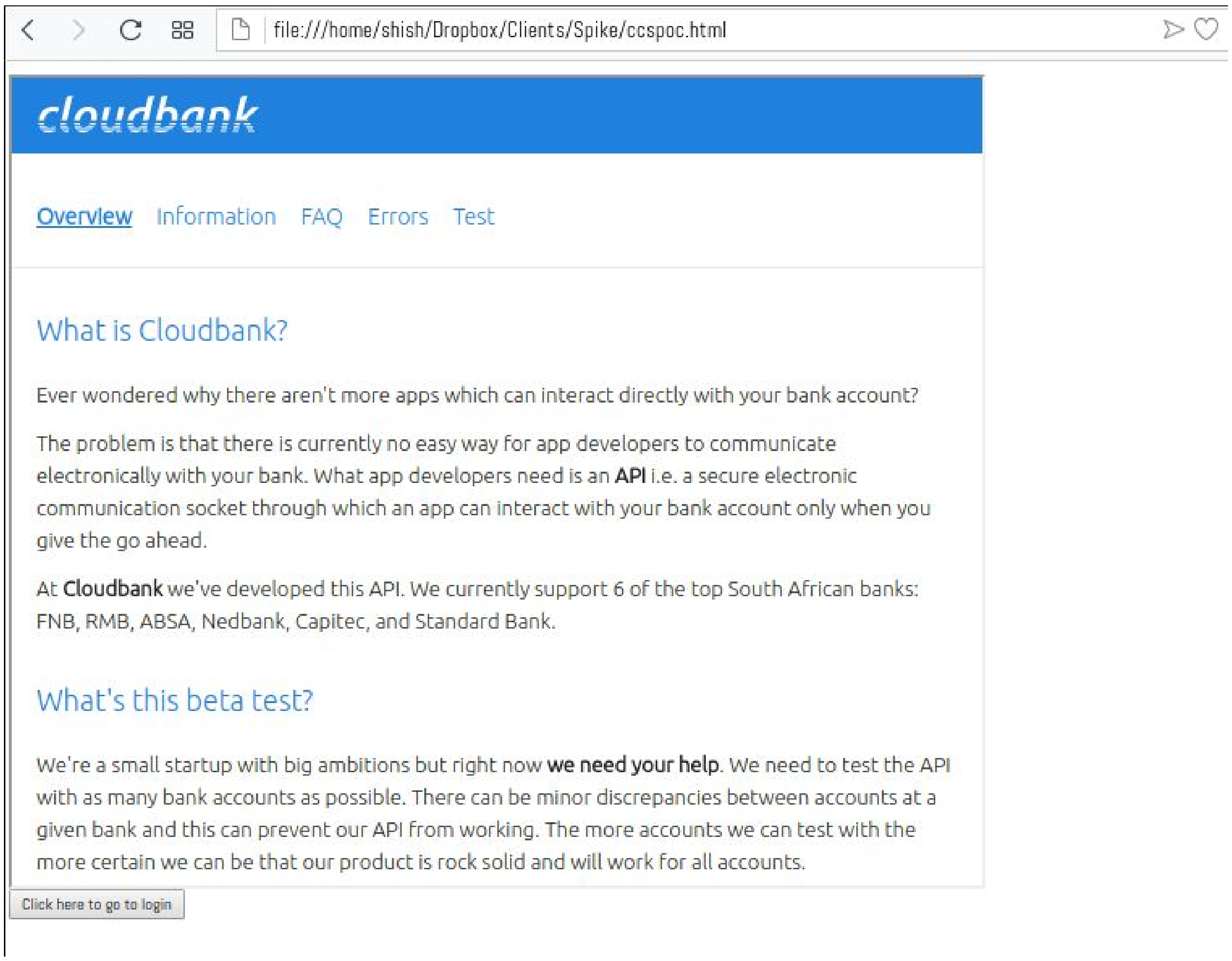
3.2 ClickJacking

The site can be set in an html iframe, this can be used in phishing campaigns

3.2.1 Vulnerability

The html below indicates how the beta app can be embedded within an iframe on an attackers site:

```
<html>

<body>

<iframe src="https://api-v5.cloudbank.co.za/beta/#FAQ"
width="600" height="500" scrolling="no"></iframe>

<form method="get" action="fk.exe">

<button>Click here to go to login</button>

</form>

</body>

</html>
```


file:///home/shish/Dropbox/Clients/Spike/ccspoc.html
cloudbank
Overview
Information
FAQ
Errors
Test
Click here to go to login

What is Cloudbank?

Ever wondered why there aren't more apps which can interact directly with your bank account?

The problem is that there is currently no easy way for app developers to communicate electronically with your bank. What app developers need is an **API** i.e. a secure electronic communication socket through which an app can interact with your bank account only when you give the go ahead.

At **Cloudbank** we've developed this API. We currently support 6 of the top South African banks: FNB, RMB, ABSA, Nedbank, Capitec, and Standard Bank.

What's this beta test?

We're a small startup with big ambitions but right now **we need your help**. We need to test the API with as many bank accounts as possible. There can be minor discrepancies between accounts at a given bank and this can prevent our API from working. The more accounts we can test with the more certain we can be that our product is rock solid and will work for all accounts.

3.2.2 Impact

This allows an attacker to impersonate the site and which would allow the attacker to capture the login credentials of any user who mistakenly visits the cloned site.

3.2.3 Likelihood

Due to objective of the beta app (namely a testing tool designed for demonstrations) it is unlikely to be a high value target for attackers to emulate. The site is not a branded app which users already trust, and hence is worth emulating by an attacker.

3.2.4 Risk Evaluation

This is a medium risk security flaw for which simple remedial action should be taken.

3.2.5 Recommendation

The following solutions are recommended in order to remediate against clickjacking (from the OWASP security advice):

Medium	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).

Low	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).

4. Additional security recommendations

Exocet Security recommends the following additional steps be implemented along with the above highlighted vulnerability recommendations:

4.1 Continuous monitoring for hack attempts

Multiple solutions for continuous monitoring exist with many different uses and applications. Exocet Security recommends that a few of these tools are set up with particular focus on ensuring that web request logs are analysed and maintained. We also recommend that some automated detections and autobanning capabilities (e.g. fail2ban, modsecurity) be implemented.(e.g. see thread discussing the issue: https://news.ycombinator.com/item?id=17014818)

4.2 Use 2-factor authentication on ssh service

Exocet Security recommend implementing a 2-factor authentication service in conjunction with ssh. Many providers exist for example: https://duo.com This will greatly improve security and ensure that only authorised logins are occuring on the ssh service. It will also minimise the risk of brute force or other password cracking attempts.

5. Summary

No exploitable security problems were detected by the scanning software. Any vulnerabilities detected were assessed to be of low risk to the Spike Data API. We highlight the clickjacking vulnerability as our main area of concern and have recommended remedial action.

Our penetration tests were likewise unable to uncover any serious flaws – we were unable to perform any code injection attacks against the API.

In our assessment this is a highly secure system with no readily apparent or exploitable security flaws.

Appendix: A OWASP Top Ten

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

1. Injection

Injection is the process of causing malicious code to be executed on the server. There a variety of ways in which a server-side app could be vulnerable to this type of exploit: it depends on the server-side stack in use as well as the pattern of requests which are made by the client-side app.
In this case only one function was called (namely https://api-v5.spikedata.co.za/transactions) which is where we focussed our efforts. From inspection the function takes the following json inputs:

```
{"bank":"x","user":"x","pass":"x","pin":"x","contact":{"email":"x","cell":"x"}}
```

We attempted to fuzz the bank variable with various snippets of code. We were not able able to cause these snippets to execute on the server.

In addition we looked at the URL changes in the web app. Fuzzing the url string did not uncover any undesirable results or vulnerabilities either e.g.

https://api-v5.spikedata.co.za/beta/#test.1

https://api-v5.spikedata.co.za/beta/#test.2

https://api-v5.spikedata.co.za/beta/#test.3

https://api-v5.spikedata.co.za/beta/#test.16

https://api-v5.spikedata.co.za/beta/#test.22

This allowed us to progress the client side app to later steps without having to fill in earlier steps - i.e. bypassing the apps intended mode of operation - but did not allow us to do anything malicious.

2. Broken Authentication

The prevalence of broken authentication is widespread due to the design and implementation of most identity and access controls. Session management is the bedrock of authentication and access controls, and is present in all stateful applications.

Spike seems to make use of a simple authentication model - name a pair of keys:

x-api-key:
`88246a40-88dc-11e7-bbe2-4f7182116a75`
x-user-key:
`ccf55bd0-898f-11e7-9851-a5eee2319d1d`

These keys are accessible in the web app. Spike have confirmed that these are testing keys and have restricted access to the system. We have alerted Spike to the possibility of a brute force attack to guess other client keys.

3. Sensitive Data Exposure

Over the last few years, this has been the most common impactful attack. The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm, protocol and cipher usage is common, particularly for weak password hashing storage techniques. For data in transit, server side weaknesses are mainly easy to detect, but it’s difficult to detect encrypted data at rest vulnerabilities in an external audit.

We have discussed this with Spike and they have confirmed that they do not store any data supplied in the request, the clients data simply passes through their infrastructure and is never stored.

4. XML External Entities

By default, many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. SAST tools can discover this issue by inspecting dependencies and configuration. SAST tools require additional manual steps to detect and exploit this issue.

There is no XML in any of the Spike requests.

5. Broken Access Control

Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers.
Access control detection is not typically amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs PUT, etc), controller, direct object references, etc.

All API requests effectively require authorization in each request. There is no session. Hence there are no “open” functions which would need to be protected with access controls. In addition the functions are access via https and using POSTs - not GETs - inline with secure app design best practices.

6. Security Misconfiguration

Attackers will often attempt to exploit unpatched flaws or access default accounts, unused pages, unprotected files and directories, etc to gain unauthorized access or knowledge of the system.

All of these were tested and scanned for, this can be seen in the OWASP ZAP scans and the Nexpose scans. No additional protected resources could be detected other than the static resources used by the web app and the Spike API function which is called by the web app (namely https://api-v5.spikedata.co.za/transactions).

7. XSS

The API is inherently immune to XSS. The web app was assessed and likewise does not provide an opportunity for any of the three XSS attacks (Stored, Reflected, and DOM-Based) - there is no saving of data from any user and hence no opportunity for XSS.

8. Insecure Deserialization

Applications and APIs will be vulnerable if they deserialize hostile or tampered objects supplied by an attacker.
This threat has been raised with Spike and they have confirmed that the json parser software in use is up to date and has a clean NPM security audit.

9. Using Components with Known Vulnerabilities

Spike was scanned using various vulnerability scanning tools including web spidering scanners and no vulnerabilities were found.
In addition Spike have conducted a security audit of the NPM modules in use by the app and resolved all security warnings.

10. Logging

In discussion with Spike they have both http request level logging as well as microservice logging throughout their infrastructure though currently no automated checking of this access. We have raised our concerns with Spike that they need to investigate an automated DoS and intrusion detection process - e.g. fail2ban.