Web Penetration Test
Phase I

Minds Web Penetration Test Phase I

Prepared for Minds • July 2018

MINDSv2018

1. Executive Summary

From May 2018 to September 2019, Minds engaged Coinspect to perform a series of security audits of the Minds Platform which included:

- A gray-box penetration test of their web application.
- A source code review of their web application.
- A source code review of the Minds Token contracts.

The objective of the audits was to evaluate the security of the web application and the smart contracts.

The present report contains the results of the gray-box penetration test of the Minds web application. During this first phase of the engagement, Coinspect identified the following issues:

High Risk	Medium Risk	Low Risk	Zero Risk
10	7	6	1

2. Introduction and Scope

The web application audit was divided into two phases.

As a first phase a gray-box security audit was conducted on the Minds web app in order to detect security, privacy, and availability related problems. The present report cointans the results of the first phase, completed on May 15th.

As a second phase a white-box security audit was conducted on the Minds web application (engine and core APIs) in order to detect security, privacy, and availability related problems.

The objective of the both phases of the audit was to test the Minds web application. The analysis included but was not limited to the following checks:

- Input validation
- Session Management
- Brute-force protection
- Cryptographic weaknesses
- Authentication and Authorization
- Denial of service prevention
- Command Injection
- Web attacks: CSRF, XSS, Clickjacking

3. Summary Of Findings

ID	Description	Risk	Fixed
MND-001	Paywall Bypass	High	✔
MND-002	Insecure Random Number Generation	High	✔
MND-003	Deserialization From Untrusted Source	High	✔
MND-004	Cross Site Scripting via Untrusted Source	High	✔
MND-005	Open Redirection	Low	✔
MND-006	Weak Password Policy	High	✔
MND-007	Strict Transport Security Not Enforced	Medium	✔
MND-008	Change Email Without Requiring Password	Medium	✔
MND-009	Missing Email Verification	Medium	✔
MND-010	Insecure Cookie Handling: Missing HTTPOnly Flag	Medium	✔
MND-011	Insecure Cookie Handling: Missing Secure Flag	Medium	✔
MND-012	TLS 1.0 Multiple Cryptographic Vulnerabilities	Low	✔
MND-013	Server Version Disclosure via HTTP Headers	Zero	✔
MND-014	Insecure Connections To Web Services	High	✔
MND-015	Sensitive Parameter Sent In URL	Medium	✔
MND-016	Hardcoded API Keys and Weak Secret Generation (Legacy Code)	Medium	✔
MND-017	Potential XSS and Path Disclosure	High	✔
MND-018	Server Side Request Forgery	High	✔
MND-019	Insecure call to file_get_contents()	Low	✔

MND-020	Weak Verification of Password Reset Token	Low	✔
MND-021	Administrative Interfaces Exposed	Low	✔
MND-022	Password Not Required To Disable 2FA	Low	✔
MND-023	Missing Sign Verification on Withdraw Operation	High	✔
MND-024	Missing Address Verifications on Smart Contract Event Handling	High	✔

4. Findings

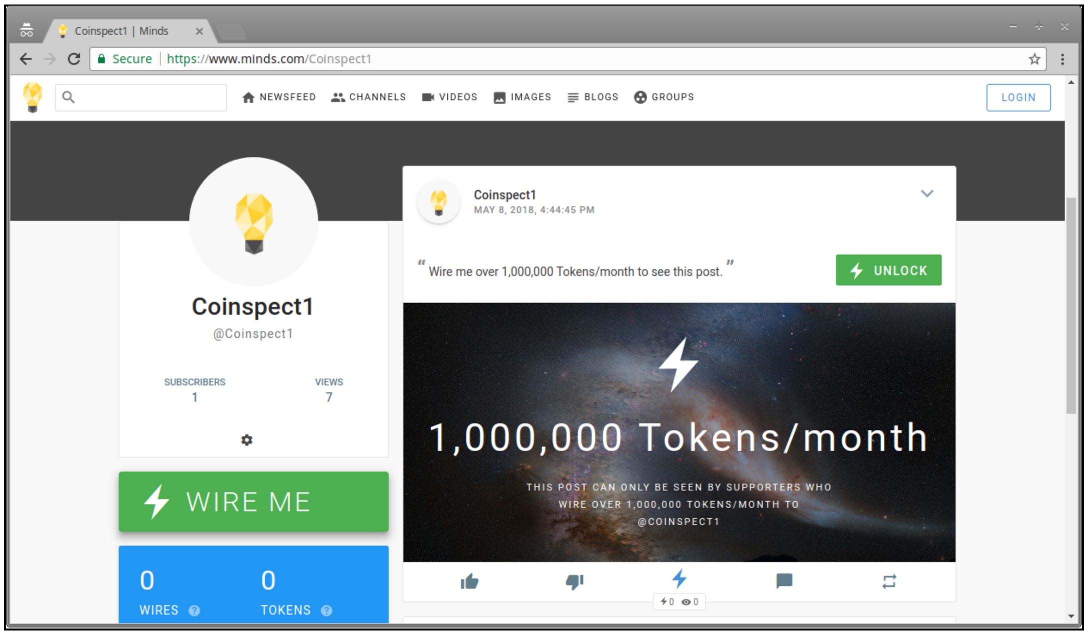
MND-001 Paywall Bypass

Total Risk	Impact	Location
High	High	https://www.minds.com/fs/v1/thumbnail https://www.minds.com/api/v1/media/thumbnails
Fixed	**Likelihood**	**Category**
✔	High	Authorization

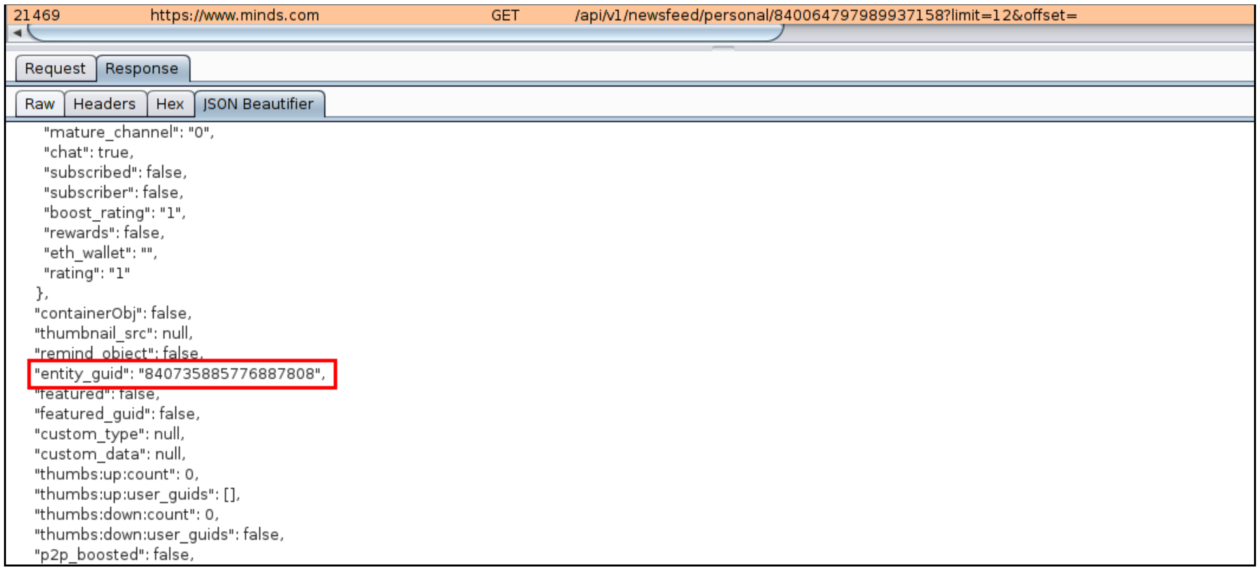
Description

The application features an option that only allows access to certain posts to paid subscribers. The posts can contain image and video attachments that are intended to become visible to users who wire a minimum amount of tokens. The missing authorization check on the affected endpoints allows access to the attachments directly without spending any token.

The following example post required 1.000.000 tokens to become visible:


Coinspect1 | Minds
Secure | https://www.minds.com/Coinspect1
NEWSFEED
CHANNELS
VIDEOS
IMAGES
BLOGS
GROUPS
LOGIN
Coinspect1
@Coinspect1
SUBSCRIBERS 1
VIEWS 7
WIRE ME
0 WIRES
0 TOKENS
Coinspect1
MAY 8, 2018, 4:44:45 PM
"Wire me over 1,000,000 Tokens/month to see this post."
UNLOCK
1,000,000 Tokens/month
THIS POST CAN ONLY BE SEEN BY SUPPORTERS WHO WIRE OVER 1,000,000 TOKENS/MONTH TO @COINSPECT1

The entity identifier (`entity_guid` parameter) can be obtained from the `api/v1/newsfeed/personal` endpoint, as shown below:

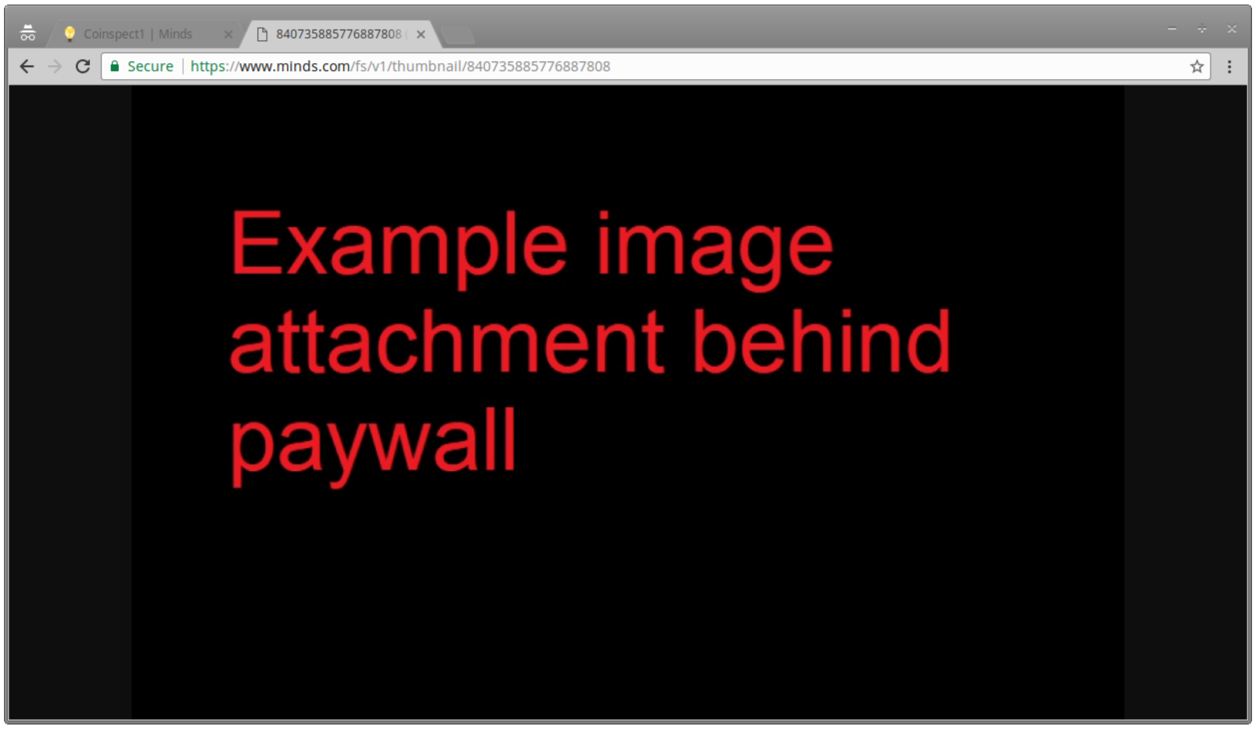


```
21469    https://www.minds.com    GET    /api/v1/newsfeed/personal/840064797989937158?limit=12&offset=
```

```
    "mature_channel": "0",
    "chat": true,
    "subscribed": false,
    "subscriber": false,
    "boost_rating": "1",
    "rewards": false,
    "eth_wallet": "",
    "rating": "1"
  },
  "containerObj": false,
  "thumbnail_src": null,
  "remind_object": false,
  "entity_guid": "840735885776887808",
  "featured": false,
  "featured_guid": false,
  "custom_type": null,
  "custom_data": null,
  "thumbs:up:count": 0,
  "thumbs:up:user_guids": [],
  "thumbs:down:count": 0,
  "thumbs:down:user_guids": false,
  "p2p_boosted": false,
```

With the `entity_guid` value, users of the platform can access the attachments behind the paywall by sending a request to the affected endpoints:

Example:
https://www.minds.com/fs/v1/thumbnail/840735885776887808
https://www.minds.com/api/v1/media/thumbnails/840385464362500096/xlarge



Coinspect1 | Minds
840735885776887808
Secure
https://www.minds.com/fs/v1/thumbnail/840735885776887808
Example image attachment behind paywall

Recommendations

Enforce authorization checks on the affected endpoints.

MND-002 Insecure Random Number Generation

Total Risk **High**	Impact High	Location https://www.minds.com
Fixed ✓	Likelihood High	Category Authentication

Description

The application uses insecure random generators to create security tokens such as: two factor authentication secret, XSRF protection token, JWT session cookie and the reset code for a forgotten password. In all cases, the application generated non-cryptographically secure values with the rand() and mt_rand() functions. An attacker can leverage the insecure number generation to predict security tokens.

engine-master/Core/Security/Password.php:72

```
    public static function reset($user)
    {
        $code = self::generate($user, microtime() . rand(), 'md5');
        $user->password_reset_code = $code;
        $user->save();
        return $code;
    }
}
```

engine-master/Core/Security/TwoFactor.php:31

```
  public function createSecret($secretLength = 16)
    {
        $validChars = $this->_getBase32LookupTable();
        unset($validChars[32]);

        $secret = '';
        for ($i = 0; $i < $secretLength; $i++) {
            $secret .= $validChars[array_rand($validChars)];
        }
        return $secret;
    }
```

The XSRF protection token is created engine-master/Core/Security/XSRF.php:11

```
    public static function buildToken()
    {
        $user = Core\Session::getLoggedinUser();
        return md5($_SESSION['__elgg_session'] . rand(1000, 9000));
    }
```

engine-master/Core/Session.php:45

```
    // Generate a simple token (private from potentially public session id)
    if (!isset($_SESSION['__elgg_session'])) {
        $_SESSION['__elgg_session'] = md5(microtime() . rand());
    }
```

The ‘JWT-Secret’ session cookie is created using the insecure mt_rand() function as input of a MD5 hash:

engine-master/Core/Provisioner/Installer.php:47-51

```
        usleep(mt_rand(1, 9999));
        $this->defaults['site-secret'] = md5(microtime() . mt_rand());

        usleep(mt_rand(1, 9999));
        $this->defaults['jwt-secret'] = md5(microtime() . mt_rand());
```

Recommendations

Replace unsafe random sources with a cryptographically secure alternative such as: openssl_random_pseudo_bytes() or random_bytes() .
http://php.net/manual/en/function.openssl-random-pseudo-bytes.php

MND-003 Deserialization From Untrusted Source

Total Risk	Impact	Location
High	High	./minds-master/plugins/spam_login_filter/start.php
Fixed ✓	Likelihood: Medium	Category: Input Validation

Description

The application includes a spam filter that verifies email address against blacklist databases. During this process, the server establish an insecure communication via HTTP to http://www.stopforumspam.com and deserialize the data without performing integrity check. This could be leveraged by an attacker to execute arbitrary code via a compromise of “stopforumspam” or performing a man-in-the-middle attack against the Minds server.

./minds-master/plugins/spam_login_filter/start.php:115

```
//StopForumSpam
            if(elgg_get_plugin_setting("use_stopforumspam") == "yes"){

                  //check the e-mail adress
                  $url =
"http://www.stopforumspam.com/api?email=".$register_email."&f=serial";

                  $return = file_get_conditional_contents($url);

                  if ($return != false) {
                        $data = unserialize($return);
                        $email_frequency = $data[email][frequency];
                              if($email_frequency != '0'){

register_error(elgg_echo('spam_login_filter:access_denied_mail_blacklist'))
;

spam_login_filter_notify_admin($register_email, $register_ip,
"Stopforumspam e-mail blacklist");
                                    $spammer = true;
                              }
                  }
```

```
                   if($spammer != true){
                   //e-mail not found in the database, now check the ip
                         $url =
"http://www.stopforumspam.com/api?ip=".$register_ip."&f=serial";

                         $return = file_get_conditional_contents($url);

                         if ($return != false) {
                               $data = unserialize($return);
                               $ip_frequency = $data[ip][frequency];
                                     if($ip_frequency != '0'){

register_error(elgg_echo('spam_login_filter:access_denied_ip_blacklist'));

spam_login_filter_notify_admin($register_email, $register_ip,
"Stopforumspam IP blacklist");
                                           $spammer = true;
                                     }
```

Recommendations

Do not use unserialize() function with untrusted data, if possible use JSON functions instead. Additionally, replace the insecure URLs with HTTPS.

MND-004 Cross Site Scripting via Untrusted Source

Total Risk	Impact	Location
High	High	engine-master/Controllers/api/v1/newsfeed/oembed/soundcloud.php:39
Fixed ✓	Likelihood: Medium	Category: Input Validation

Description

The application server connects to an external source (soundcloud.com) via an insecure HTTP connection to retrieve resources and display embedded content. This content is rendered without any sanitization using the `bypassSecurityTrustHtml` function. Therefore, a man-in-the-middle attack against the server or malicious content returned by http://soundcloud.com/oembed endpoint will lead to Cross-Site-Scripting in Mind's web application.

engine-master/Controllers/api/v1/newsfeed/oembed/soundcloud.php:39

```
$endpoint = 'http://soundcloud.com/oembed?' . http_build_query($query);
```

front-master/src/app/common/components/rich-embed/rich-embed.ts:152

```
return this.sanitizer.bypassSecurityTrustHtml(response.html);
```

Recommendations

Do not disable Angular's built-in sanitizer when rendering untrusted sources. Replace the URL to use the secure HTTPS protocol. For further recommendations, refer to OWASP's Cross-site Prevention Cheat sheet at:
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

MND-005 Open Redirection

Total Risk	Impact	Location
Low	Low	https://cdn.minds.com/thumbProxy?src=
Fixed	Likelihood	Category
✓	Low	Input Validation

Description

An open redirection vulnerability can facilitate phishing attacks against the users of the application. The victim would trust the mind.com domain in the URL, and might not notice the redirection to the attacker's site.

Example open redirection URL:
https://cdn.minds.com/thumbProxy?src=http://coinspect.com

Recommendations

Create a server-side whitelist of allowed domains before issuing the redirect. For further recommendations, refer to OWASP's Unvalidated Redirects and Forwards Cheat Sheet at: https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet

MND-006 Weak Password Policy

Total Risk **High**	Impact High	Location https://www.minds.com/api/v1/register https://www.minds.com/api/v1/settings
Fixed ✓	Likelihood High	Category Authentication

Description

During the registration process the application accepts passwords of 8 characters minimum without any complexity requirement. Moreover, once the account is created, users can change the password from the profile settings to a password of any length (ie: a single character).
Attackers can easily identify weak passwords. Users of the application may choose weak passwords such as having the password match the same value as the username, and expose themselves to brute force attack scenarios.

Recommendations

Create a password policy that requires a minimum password length of 8 characters and the use of upper case, lower case, numeric and special characters.

MND-007 Strict Transport Security Not Enforced

Total Risk	Impact	Location
Medium	Medium	https://www.minds.com
Fixed ✓	Likelihood	Category
	Medium	Communications Security

Description

The 'Strict Transport Security' response header (HSTS) enforces web browsers to establish all communications over HTTPS to the specified domain. All attempts to access the site using HTTP are automatically converted to secure HTTPS. The application server did not include the HSTS header.

Recommendations

Include the HTTP `Strict-Transport-Security` header in all HTTPS responses.
The following link contains more information and configuration guide for Nginx server:
https://www.nginx.com/blog/http-strict-transport-security-hsts-and-nginx

MND-008 Change Email Without Requiring Password

Total Risk	Impact	Location
Medium	Medium	https://www.minds.com/api/v1/settings
Fixed ✓	Likelihood	Category
	High	Authorization

Description

The application allows users to change the email address without requiring the current password.
This allows attackers with access to an open session (via XSS or other vulnerability) to gain permanent access to the account by changing the current email to an attacker's controlled address and then requesting a password reset.

Recommendations

Always require current password to change the email address associated to the account.

MND-009 Missing Email Verification

Total Risk Medium	Impact Medium	Location https://www.minds.com/api/v1/settings
Fixed ✓	Likelihood High	Category Authentication

Description

The registration process does not require confirmation of the account or the email address, this behavior can be leveraged by malicious users to create bots or perform malicious actions against the application.

Recommendations

Validate the email address before account activation. Additionally, consider implementing a CAPTCHA solution to prevent automated account creation.

MND-010 Insecure Cookie Handling: Missing HTTPOnly Flag

Total Risk **Medium**	Impact Medium	Location https://www.minds.com
Fixed ✓	Likelihood High	Category Cookie Handling

Description

Cookies without the “HTTPOnly” attribute can be read by client-side JavaScript.
This can be leveraged by attackers during a Cross-Site-Scripting attack to access cookie's values such as the session identifier.

Affected cookies:

- XSRF-TOKEN
- minds
- mwa
- socket_jwt

Recommendations

Set the “HttpOnly” attribute on all sensitive cookies. This limits the exposure of information that can be gained through exploiting other vulnerabilities.

MND-011 Insecure Cookie Handling: Missing Secure Flag

Total Risk **Medium**	Impact Medium	Location https://www.minds.com
Fixed ✓	Likelihood High	Category Cookie Handling

Description

Cookies without the “Secure” attribute may be sent to the site during an unencrypted session, which could allow an attacker sniffing the application’s traffic to obtain sensitive information such as the session cookie and put the application at risk of a session hijacking attack.

The following cookies containing sensitive information are affected:

- XSRF-TOKEN
- minds
- mwa
- socket_jwt

Recommendations

Set the “Secure” attribute on the session cookies. This limits the exposure of information that can be gained through exploiting other vulnerabilities.

MND-012 TLS 1.0 Multiple Cryptographic Vulnerabilities

Total Risk **Low**	Impact Medium	Location https://www.minds.com
Fixed ✓	Likelihood Low	Category Configuration Management

Description

The server mentioned above supports version 1.0 of the TLS protocol. Version 1.0 of the TLS protocol suffers from multiple well-known cryptographic flaws.

Attackers may exploit vulnerabilities in the 1.0 version of the TLS protocol with the intention of conducting man in the middle attacks, decrypting communications between clients and the affected servers.

Recommendations

Disable version 1.0 of the TLS protocol. Instead use TLS 1.1 or higher:
https://aws.amazon.com/blogs/security/how-to-control-tls-ciphers-in-your-aws-elastic-beanstalk-application-by-using-aws-cloudformation

MND-013 Server Version Disclosure via HTTP Headers

Total Risk	Impact	Location
Zero	Low	https://www.minds.com
Fixed	Likelihood	Category
❌	Low	Configuration Management

Description

The web server disclosed software name and version information on the HTTP headers.
The information revealed could help attackers to plan and launch further attacks against the application. The following header was included on the HTTP responses:

```
Server: nginx/1.13.6
```

Recommendations

Turn off the server_tokens directive in '/etc/nginx/nginx.conf' configuration file.
http://nginx.org/en/docs/http/ngx_http_core_module.html#server_tokens

Fix: Minds considers that being an open source project, hiding version information does not make any difference.

MND-014 Insecure Connections To Web Services

Total Risk	Impact	Location
High	High	https://www.minds.com
Fixed ✓	Likelihood: Medium	Category: Communications Security

Description

The application server communicates to several external sites via insecure HTTP connections. An attacker with man-in-the-middle ability could read and modify data to gain further access to the infrastructure or additional systems.
The following list is not complete. A source code security audit is recommended to identify additional instances.

Code snippets that demonstrates this issue:

minds-master/plugins/spam_login_filter/start.php:122

```
$url =
"http://www.stopforumspam.com/api?email=".$register_email."&f=serial";
$url = "http://www.stopforumspam.com/api?ip=".$register_ip."&f=serial";
```

engine-master/Controllers/api/v1/newsfeed/preview.php:30

```
curl_setopt($ch, CURLOPT_URL,
"http://open.iframe.ly/api/iframely?origin=".$iframelyConfig['origin']."&ap
i_key=".$iframelyConfig['key']."&url=".urlencode($url));
```

Recommendations

Enforce secure HTTPS connections for all external systems.

MND-015 Sensitive Parameter Sent In URL

Total Risk
Medium

Fixed ✓

Impact
Medium

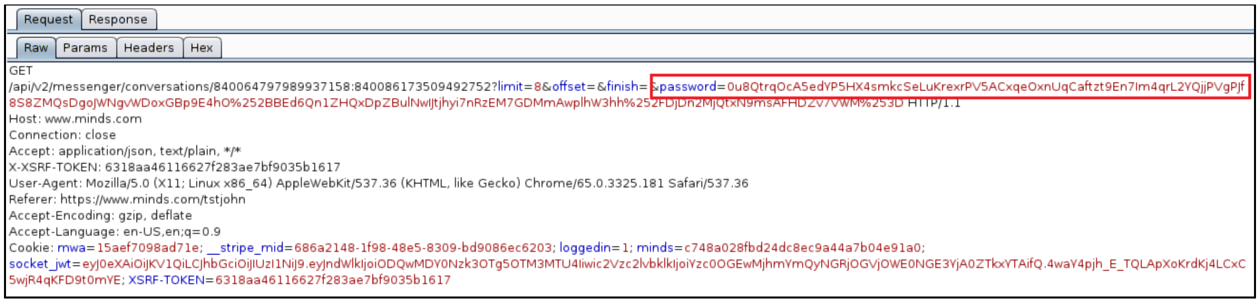
Likelihood
Medium

Location
https://www.minds.com/api/v2/messenger/conversations/840064797989937158:840086173509492752?limit=8&offset=&finish=&password=...

Category
Communications Security

Description

When sensitive data is passed in parameters of an URL, attackers might be able to obtain the data from several locations such as: web server's access logs and client's browser cache.
In this case, the "password" parameter used to unlock the private key and access chat content was sent in the URL.



```
GET
/api/v2/messenger/conversations/840064797989937158:840086173509492752?limit=8&offset=&finish=&password=0u8QtrqOcA5edYP5HX4smkcSeLuKrexrPV5ACxqeOxnUqCaftzt9En7Im4qrL2YQjjPVgPJf
8S8ZMQsDgoJWNgvWDoxGBp9E4hO%252BBEd6Qn1ZHQxDpZBulNwlJtjhyi7nRzEM7GDMmAwplhW3hh%252FDjDn2MjQtxN9msAFHDZv7VWM%253D HTTP/1.1
Host: www.minds.com
Connection: close
Accept: application/json, text/plain, */*
X-XSRF-TOKEN: 6318aa46116627f283ae7bf9035b1617
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
Referer: https://www.minds.com/tstjohn
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: mwa=15aef7098ad71e; __stripe_mid=686a2148-1f98-48e5-8309-bd9086ec6203; loggedin=1; minds=c748a028fbd24dc8ec9a44a7b04e91a0;
socket_jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJndWlkIjoiODQwMDY0Nzk3OTg5OTM3MTU4Iiwic2Vzc2lvbklkIjoiYzc0OGEwMjhmYmQyNGRjOGVjOWE0NGE3YjA0ZTkxYTAifQ.4waY4pjh_E_TQLApXoKrdKj4LCxC
5wjR4qKFD9t0mYE; XSRF-TOKEN=6318aa46116627f283ae7bf9035b1617
```

Recommendations

Use POST method to send sensitive parameters.

MND-016 Hardcoded API Keys and Weak Secret Generation (Legacy Code)

Total Risk	Impact	Location
Medium	Medium	Entities/Video.php:39 Controllers/api/v1/geolocation.php:22 lib/users.php:837,849 Core/clusters.php:234
Fixed ✓	Likelihood Medium	Category Security Best Practices

Description

Coinspect identified security issues in code that later was recognized as **legacy code**:

Hardcoded API keys for CineMR[1] and MapQuest[2] services stored insecurely on the source code repository. In the case of CineMR service, consultants attempted to connect and interact with the cinemr.minds.com server using all the methods exposed on the client's SDK. In all cases, the server returned a HTTP 500 error.

1. CineMR Entities/Video.php:39

```
public function cinemr()
{
   return new cinemr\sdk\client(array(
           'account_guid' => '335988155444367360',
                       'secret' =>
'+/rW1ArsueEjXK++0zkxlBrbLkb5suHqvqZJ64kX8rk=',
           'uri' => 'http://cinemr.minds.com'
       ));
}
```

2. MapQuest Controllers/api/v1/geolocation.php:22

```
public function get($pages)
{
$url=
"http://open.mapquestapi.com/nominatim/v1/search.php?key=ohEcFAArFVNvzTlwGQ
S5C9XGkAZ4iW9p&format=json&q=" . urlencode($_GET['q']) .
"&addressdetails=1";
```

In addition, instances of insecure generation authentication tokens, passwords and salts were detected. Coinspect confirmed during an interview with Minds that the issues mentioned on this finding are part of legacy unused code.

lib/users.php:837

```
function generate_random_cleartext_password() {
  return substr(hash('sha256', microtime() . rand()), 0, 8);
}
```

lib/users.php:849

```
function   generate_user_password(ElggUser   $user,   $password,   $algo   =
'sha256') {
  if($algo == 'md5')
        return md5($password . $user->salt);
  return hash('sha256', $password . $user->salt);
}
```

Core/clusters.php:234

```
public static function generateSecret($length = 128)
{
$chars="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%
^&*()_-=+;:,.?";
   return hash('sha256', substr(str_shuffle($chars), 0, $length));
}
```

Recommendation

Remove legacy unused code from the source code repository.

MND-017 Potential XSS and Path Disclosure

Total Risk	Impact	Location
High	High	Controllers/fs/v1/banners.php:100
Fixed ✓	Likelihood Medium	Category Input Handling

Description

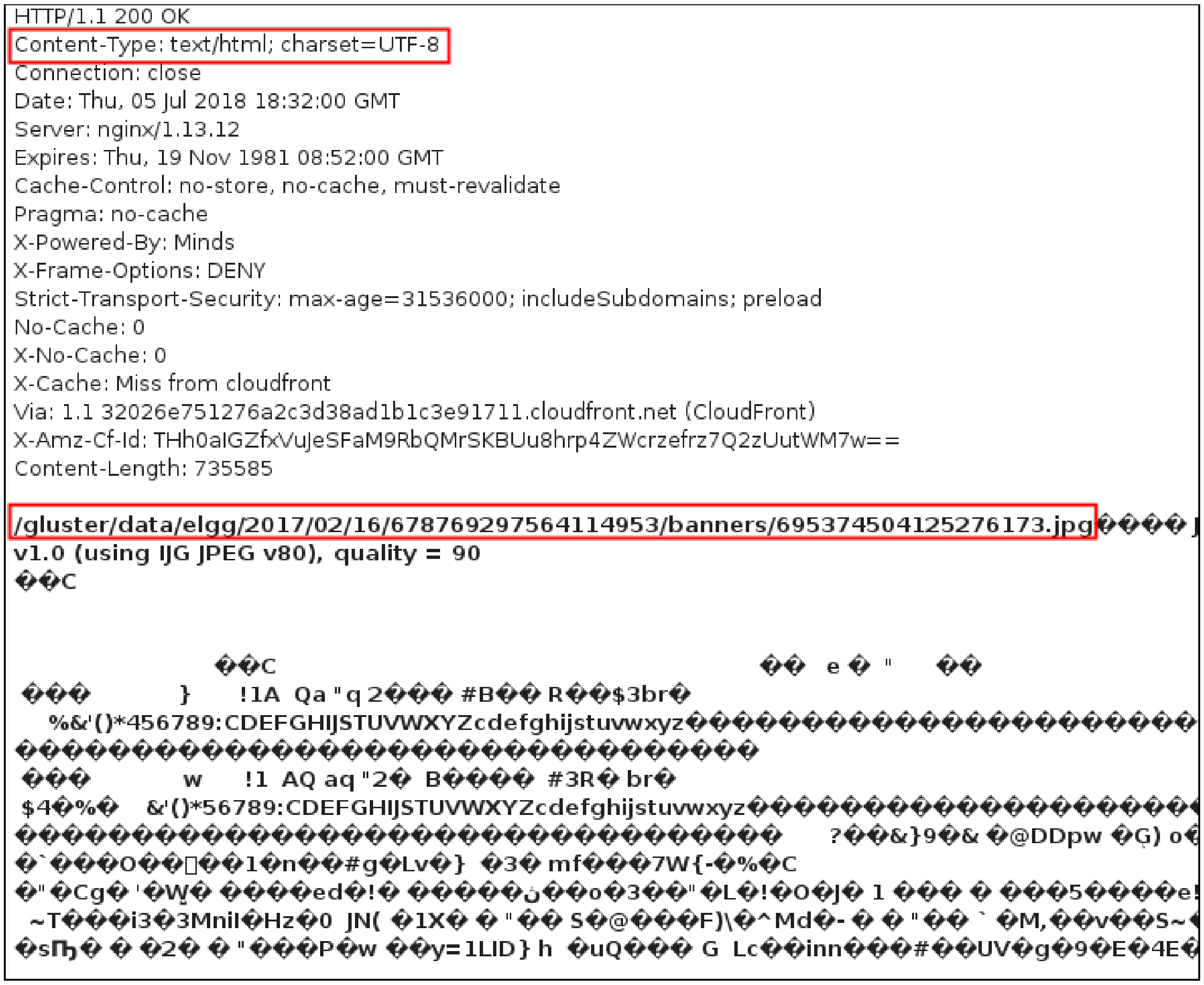
The Banner FS endpoint contains a debug case that could lead to an exploitable Cross-Site-Scripting condition. Specifically, when the 'testing' parameter is appended to the 'fs/v1/banners' URL, the application returns an internal path followed by the image on the response body.
An attacker could attempt to upload an specially crafted image that when its processed by the resizing function (get_resized_image_from_uploaded_file) results in valid HTML code.
This could be leveraged to create a Cross-Site-Scripting attack as the endpoint returns the resized image with "`Content-Type: text/html`" header.
Additionally, when the debug parameter present, the internal file path disclosed (ie: /gluster/data/elgg/2017/02/16/678769297564114953/banners/695374504125276173.jpg)

Controllers/fs/v1/banners.php:100

```
if($_GET['testing']){
    echo $f->getFilenameOnFilestore();
    echo $content;exit;
}
```

Example URL and HTTP response.

https://cdn.minds.com/fs/v1/banners/678769297564114953/fat/1487289749?testing=true



```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Connection: close
Date: Thu, 05 Jul 2018 18:32:00 GMT
Server: nginx/1.13.12
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-Powered-By: Minds
X-Frame-Options: DENY
Strict-Transport-Security: max-age=31536000; includeSubdomains; preload
No-Cache: 0
X-No-Cache: 0
X-Cache: Miss from cloudfront
Via: 1.1 32026e751276a2c3d38ad1b1c3e91711.cloudfront.net (CloudFront)
X-Amz-Cf-Id: THh0aIGZfxVuJeSFaM9RbQMrSKBUu8hrp4ZWcrzefrz7Q2zUutWM7w==
Content-Length: 735585

/gluster/data/elgg/2017/02/16/678769297564114953/banners/695374504125276173.jpg���� J
v1.0 (using IJG JPEG v80), quality = 90
��C



                ��C                                          ��  e � "    ��
 ���         }     !1A  Qa "q 2��� #B�� R��$3br�
   %&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz����������������������
����������������������������������
 ���          w     !1  AQ aq "2�  B����  #3R� br�
 $4�%�   &'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz�������������������
�����������������������������������     ?���&}9��& �@DDpw �G) o�
�`���O���▯��1�n��#g�Lv�}  �3� mf����7W{-�%�C
�"�Cg� '�W̤� ����ed�!� ������ؘ��o�3��"�L�!�O�J� 1 ��� � ���5�����e!
 ~T����i3�3Mnil�Hz�0  JN( �1X� � "�� S�@����F)\�^Md�- � � "�� ` �M,��v��S~�
�sЋ� � �2� � "���P�w ��y=1LID} h  �uQ��� G  Lc��inn���#��UV�g�9�E�4E�
```

Recommendation

Remove the testing parameter and debugging information from the affected function.

MND-018 Server Side Request Forgery

Total Risk	Impact	Location
High	High	Controllers/thumbProxy.php:48
Fixed ✓	Likelihood	Category
	High	Input Handling

Description

The server application downloads files from an user-provided URL parameter without any restriction on the protocol nor target address.
An attacker can force the Minds server to connect to his own server and attempt to exploit existing vulnerabilities on several protocols of the PHP's curl library (https://curl.haxx.se/docs/vulnerabilities.html) or attempt to enumerate internal network hosts by submitting IP ranges and observing the different server error responses. More information can be found on the following link: https://www.owasp.org/index.php/Server_Side_Request_Forgery.

Controllers/thumbProxy.php:48

```
$src = urldecode($_GET['src']);
//assume https if url not set
if (strpos($src, 'http') === false) {
   $src = "https:$src";
}

if (strpos($src, 'blog/header/') !== false) {
   $src = str_replace('blog/header/', 'fs/v1/banners/', $src);
}

//get the original file
$ch = curl_init($src);
curl_setopt($ch, CURLOPT_USERAGENT, 'Minds/2.0 (+http://www.minds.com/)');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_NOSIGNAL, 1);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 4);
curl_setopt($ch, CURLOPT_TIMEOUT_MS, 2000);
$image = curl_exec($ch);
$errorno = curl_errno($ch);
```

```
curl_close($ch);
```

Recommendation

Avoid curl connections to user-provided URLs. If avoiding curl connections with user-provided URLs is not possible, consider to limit the protocol to HTTP/HTTPS and exclude internal IP ranges to prevent enumeration of internal hosts.

MND-019 Insecure call to file_get_contents()

Total Risk	Impact	Location
Low	Medium	Controllers/api/v1/webhooks/awssns.php:52
Fixed	Likelihood	Category
✓	Low	Input Handling

Description

The application attempts to retrieve an arbitrary file from a user-provided parameter in a `file_get_contents()` call. The function call return value is never processed. This behavior, suggests that it is most likely a development artifact and can be removed.
An attacker could supply a special linux filename such as: "/dev/zero" to trigger a Denial Of Service condition.
Depending on how the code changes over time, this call could lead to exploitable conditions (ie: read local files).

Controllers/api/v1/webhooks/awssns.php:52

```
// Check if we're getting a subscription confirmation URL
if ($message['Type'] === 'SubscriptionConfirmation') {
   // Dump to the error log
   error_log('[AWS-SES] Subscribed to URL: ' . $message['SubscribeURL']);
   // Subscribe
   file_get_contents($message['SubscribeURL']);

   return Factory::response([ ]);
}
```

Recommendation

Remove the `file_get_contents` function call.

MND-020 Weak Verification of Password Reset Token

Total Risk	Impact	Location
Low	High	Controllers/api/v1/forgotpassword.php:85
Fixed	**Likelihood**	**Category**
✔	Low	Authentication

Description

The password reset token code does not verify the type of the variables being compared.
The password_reset_code is a hexadecimal string (SHA-512) generated from a secure random source.
If the generation of the SHA-512 hash results in a hex string beginning with "0e" followed by all digits (0-9), the whole string is treated as a float due to the loose comparison (!=). Therefore, an attacker can send '0' as hash, and the compare operation will return success allowing the attacker to set a new password.
The likelihood of generating a hash value with those properties are low. Nonetheless, the application should use strict type comparison.

Controllers/api/v1/forgotpassword.php:85

```
if ($user->password_reset_code && $user->password_reset_code !=
$_POST['code']) {
   $response['status'] = "error";
   $response['message'] = "The reset code is invalid";
   break;
}
```

Recommendation

Enforce a type-safe compare of ‘password_reset_code’ and $_POST[‘code’] variables (!==).
More information can be found at: http://php.net/manual/en/language.types.type-juggling.php

MND-021 Administrative Interfaces Exposed

Total Risk	Impact	Location
Low	Medium	./Controllers/api/v1/admin/pages.php ./Controllers/api/v2/blockchain/transactions.php [..]
Fixed	Likelihood	Category
❌	Low	Access Control

Description

Administrative interfaces are exposed to Internet. If an attacker compromises administrative credentials via bruteforce, phishing or a vertical privileges escalation vulnerability, he would be able to gain privileged access to administration functionality. The application relies on a session variable to differentiate privileged accounts and no other network access control were identified.

Controllers/api/v1/settings.php:31

```
if (Core\Session::getLoggedInUser()->isAdmin() && isset($pages[0])) {
   $user = new Entities\User($pages[0]);
} else {
   $user = Core\Session::getLoggedInUser();
}
```

Affected Endpoints

./Controllers/api/v1/block.php
./Controllers/api/v1/blog.php
./Controllers/api/v1/channel.php
./Controllers/api/v1/entities.php
./Controllers/api/v1/groups/group.php
./Controllers/api/v1/groups/membership.php
./Controllers/api/v1/monetization/ads.php
./Controllers/api/v1/monetization/ledger.php
./Controllers/api/v1/monetize.php
./Controllers/api/v1/payments/plans.php
./Controllers/api/v1/settings.php
./Controllers/api/v1/wire/threshold.php
./Controllers/api/v2/blockchain/transactions.php
./Controllers/api/v2/settings/emails.php

Recommendation

On the long term, segregate administrative functionality from the public web application to minimize its exposure.

MND-022 Password Not Required To Disable 2FA

Total Risk	Impact	Location
Low	Low	./Controllers/api/v1/twofactor.php:124
Fixed	Likelihood	Category
✓	Low	Authentication

Description

The application does not require the current password to disable the second factor authentication. A session hijacker can remove the second factor protection by sending a DELETE request to '/api/v1/authenticate' endpoint.

Controllers/api/v1/twofactor.php:124

```
public function delete($pages)
   {
       $user = Core\Session::getLoggedInUser();
       $user->twofactor = false;
       $user->telno = false;
       $user->save();
       return Factory::response([]);
   }
}
```

Recommendation

Require the user's current password to disable second factor authentication. Additionally, consider sending an email to notify the user.

MND-023 Missing Sign Verification on Withdraw Operation

Total Risk **High**	Impact High	Location /Controllers/api/v2/blockchain/transactions.php:125
Fixed ✓	Likelihood High	Category Input Handling

Description

The blockchain transactions API endpoint does not verify the sign of the submitted withdraw amount. When a new withdraw is requested, the client sends a transaction to the blockchain and a record for such transaction is stored in the backend retrieving the amount from a POST parameter, which can be tampered with a negative amount by the user.

Controllers/api/v2/blockchain/transactions.php:125

```
case "withdraw":
   $request = new Withdraw\Request();
   $request->setTx($_POST['tx'])
       ->setUserGuid(Session::getLoggedInUser()->guid)
       ->setAddress($_POST['address'])
       ->setTimestamp(time())
       ->setGas($_POST['gas'])
       ->setAmount((string) BigNumber::_($_POST['amount']));
```

The following code checks if the requested amount is available on the account using less-than operator. Therefore, a negative amount will pass this check:

Core/Rewards/Withdraw/Manager.php:89

```
if ($available->lt($request->getAmount())) {
$readableAvailable = round(BigNumber::fromPlain($available,
18)->toDouble(),4);
   throw new \Exception("You can only request {$readableAvailable}
tokens.");
}
```

After the transaction is completed, the application will attempt to debit the user's balance converting the amount from the request to negative (`$request->getAmount())->neg()`). If the amount in the request is already a negative value, the transaction will be created with a positive amount value.

Core/Rewards/Withdraw/Manager.php:137

```
//debit the users balance
$user = new User;
$user->guid = (string) $request->getUserGuid();

try {
   $this->offChainTransactions
        ->setUser($user)
        ->setType('withdrawal')
        //->setTx($request->getTx())
        ->setAmount((string) BigNumber::_($request->getAmount())->neg())
        ->create();
```

Coinspect attempted to confirm this issue on the production website by sending a POST request to '/api/v2/blockchain/transactions/withdraw', however the server did not return the expected exception messages ("You can only request n tokens", "A withdrawal has already been requested in the last 24 hours", etc.) and always returned HTTP 500 error. The source of this error is a comparison made between the stored record on the backend database and the amount sent to the blockchain transaction, which is obtained from the "WithdrawalRequest" event emitted by the Minds smart contracts. As the amount listed in the transaction and the one stored in the record do not match, the withdrawal process fails.

Recommendation

Always verify the requested withdrawal amount is a positive value.

MND-024 Missing Address Verifications on Smart Contract Event Handling

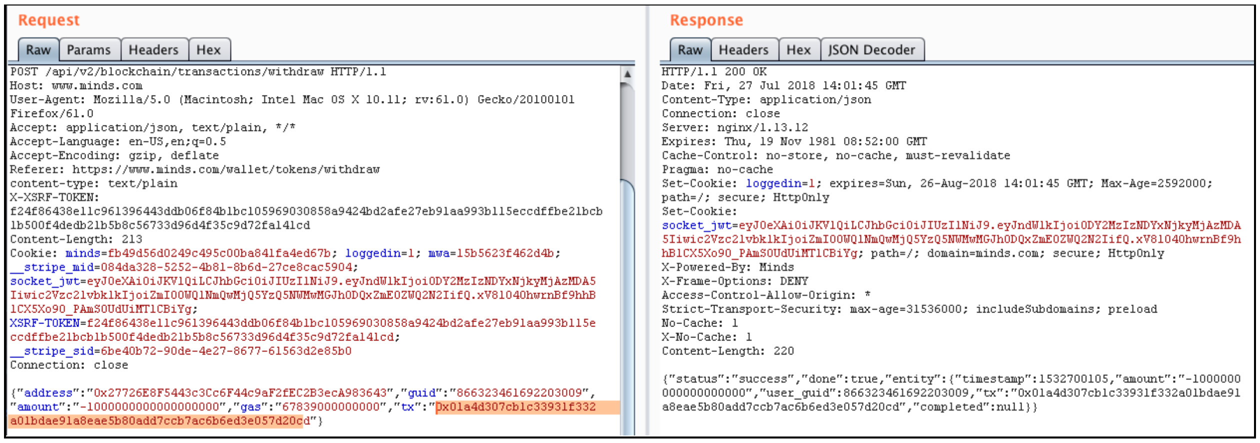
Total Risk	Impact	Location
High	High	/Core/Blockchain/Events/WithdrawEvent.php
Fixed ✓	Likelihood: High	Category: Authorization

Description

The backend handlers for smart contract events do not verify the address of the contract that emitted the event. Attackers can create smart contracts to emit events with the same topic codes used by Minds contracts. For example, for the withdrawal operation, attackers can submit the transaction ID of a call to a malicious contract and the backend won't distinguish the events produced by that call from a legitimate call to the Minds contract.

Coinspect attempted to exploit MND-23 by spoofing smart contract events as illustrated in the following transaction:

https://rinkeby.etherscan.io/tx/0x01a4d307cb1c33931f332a01bdae91a8eae5b80add7ccb7ac6b6ed3e057d20cd#eventlog



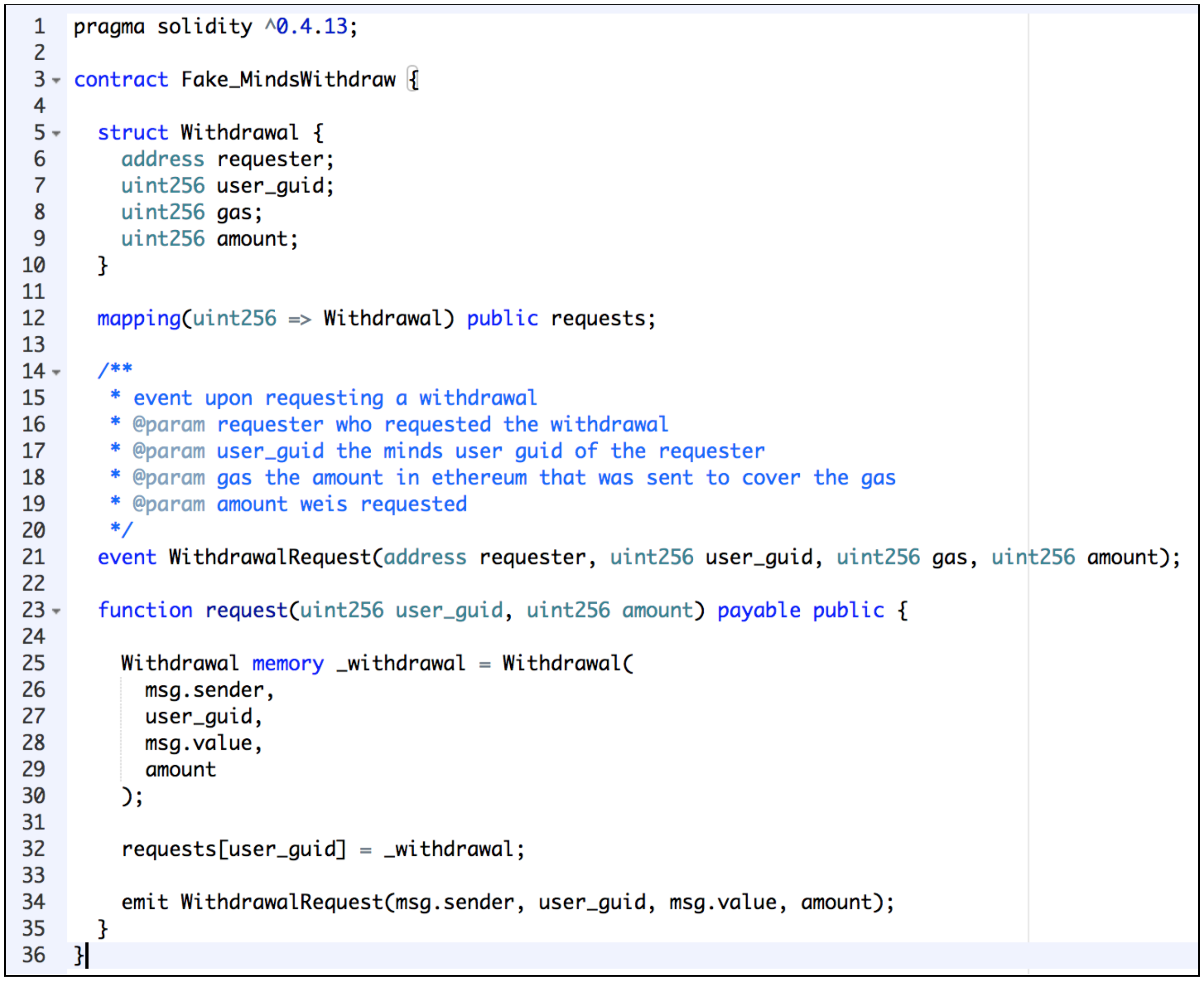
```
POST /api/v2/blockchain/transactions/withdraw HTTP/1.1
Host: www.minds.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:61.0) Gecko/20100101
Firefox/61.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.minds.com/wallet/tokens/withdraw
content-type: text/plain
X-XSRF-TOKEN:
f24f86438e11c961396443ddb06f84b1bc105969030858a9424bd2afe27eb91aa993b115eccdffbe21bcb
1b500f4dedb21b5b8c56733d96d4f35c9d72fa141cd
Content-Length: 213
Cookie: minds=fb49d56d0249c495c00ba841fa4ed67b; loggedin=1; mwa=15b5623f462d4b;
__stripe_mid=084da328-5252-4b81-8b6d-27ce8cac5904;
socket_jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJndWlkIjoiODY2MzIzNDYxNjkyMjAzMDA5
Iiwic2Vzc2lvbklkIjoiZmI0OWQ1NmQwMjQ5YzQ5NWMwMGJhODQxZmE0ZWQ2N2IifQ.xV81040hwrnBf9hhB
1CX5Xo90_PAmSOUdUiMT1CBiYg;
XSRF-TOKEN=f24f86438e11c961396443ddb06f84b1bc105969030858a9424bd2afe27eb91aa993b115e
ccdffbe21bcb1b500f4dedb21b5b8c56733d96d4f35c9d72fa141cd;
__stripe_sid=6be40b72-90de-4e27-8677-61563d2e85b0
Connection: close

{"address":"0x27726E8F5443c3Cc6F44c9aF2fEC2B3ecA983643","guid":"866323461692203009",
"amount":"-1000000000000000000","gas":"67839000000000","tx":"0x01a4d307cb1c33931f332
a01bdae91a8eae5b80add7ccb7ac6b6ed3e057d20cd"}
```

```
HTTP/1.1 200 OK
Date: Fri, 27 Jul 2018 14:01:45 GMT
Content-Type: application/json
Connection: close
Server: nginx/1.13.12
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: loggedin=1; expires=Sun, 26-Aug-2018 14:01:45 GMT; Max-Age=2592000;
path=/; secure; HttpOnly
Set-Cookie:
socket_jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJndWlkIjoiODY2MzIzNDYxNjkyMjAzMDA
5Iiwic2Vzc2lvbklkIjoiZmI0OWQ1NmQwMjQ5YzQ5NWMwMGJhODQxZmE0ZWQ2N2IifQ.xV81040hwrnBf9h
hB1CX5Xo90_PAmSOUdUiMT1CBiYg; path=/; domain=minds.com; secure; HttpOnly
X-Powered-By: Minds
X-Frame-Options: DENY
Access-Control-Allow-Origin: *
Strict-Transport-Security: max-age=31536000; includeSubdomains; preload
No-Cache: 1
X-No-Cache: 1
Content-Length: 220

{"status":"success","done":true,"entity":{"timestamp":1532700105,"amount":"-1000000
000000000000","user_guid":866323461692203009,"tx":"0x01a4d307cb1c33931f332a01bdae91
a8eae5b80add7ccb7ac6b6ed3e057d20cd","completed":null}}
```

The attacker submits a request to the withdraw endpoint passing the transaction ID of a call to his own contract.



```
pragma solidity ^0.4.13;

contract Fake_MindsWithdraw {

  struct Withdrawal {
    address requester;
    uint256 user_guid;
    uint256 gas;
    uint256 amount;
  }

  mapping(uint256 => Withdrawal) public requests;

  /**
   * event upon requesting a withdrawal
   * @param requester who requested the withdrawal
   * @param user_guid the minds user guid of the requester
   * @param gas the amount in ethereum that was sent to cover the gas
   * @param amount weis requested
   */
  event WithdrawalRequest(address requester, uint256 user_guid, uint256 gas, uint256 amount);

  function request(uint256 user_guid, uint256 amount) payable public {

    Withdrawal memory _withdrawal = Withdrawal(
      msg.sender,
      user_guid,
      msg.value,
      amount
    );

    requests[user_guid] = _withdrawal;

    emit WithdrawalRequest(msg.sender, user_guid, msg.value, amount);
  }
}
```

The fake Minds contract used by the attacker to emit events Withdraw events.

Recommendation

Implement stronger verifications of smart contract consensus data. In each event handler add checks to verify the address of the contract tha emitted the event.

See:
https://github.com/ethereum/EIPs/issues/781

5. Disclaimer

The information presented in this document is provided as is and without warranty. Vulnerability assessments are a “point in time” analysis and as such it is possible that something in the environment could have changed since the tests reflected in this report were run. This report should not be considered a perfect representation of the risks threatening the analysed system, networks and applications.