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

Pentest-Report Telekube 08.2017

Cure53, Dr.-Ing. M. Heiderich, M. Wege, N. Hippert, BSc. D. Weißer, MSc. N. Krein, J. Larsson, Dipl-Ing. A. Inführ, M. Kinugawa

Index

Introduction

"Telekube combines a production hardened deployment of Kubernetes with Teleport, our multi-region SSH server, so you can effectively manage multiple deployments of Kubernetes applications across various regions, data centers and cloud providers."
From http://gravitational.com/telekube/

This report documents the findings of a penetration test and source code audit against the Telekube product. The assessment was completed by Cure53 over the course of twenty days in late August and early September of 2017, overall yielding a total of eleven security-relevant findings. The scope of the project encompassed multiple components of the **Gravitational Telekube** software compound. More specifically, it included web interfaces, CLI clients, the general Kubernetes and Docker integration, as well as the particularities of the AWS/IAM integration. Also in scope were all sources warranting audit and review.

Given the complexity and extensive scope of this assessment, Cure53 composed a team consisting of nine testers, who were carefully selected to ensure that all skillsets needed for a successful completion of this high-difficulty project can be benefitted from. As a result, various methodologies could be employed for this test and audit. It is worth noting that the Gravitational in-house team set up a test environment for Cure53. This entailed granting testers SSH access, which can also be inferred from the links and URL examples presented across different technical descriptions in this report. Further, all relevant sources have been shared with Cure53 via Github. To assist verification, Cure53 additionally made use of a cloud VM instance available to all testers. Expectedly, this instance has been destroyed afterwards. It should be mentioned that some parts of the scope overlapped with the items already investigated during a previous Cure53 test against Gravitational Teleport, so the current assessment skipped through these aspects. Finally, to facilitate and foster good communications between the testing team and the Gravitational maintainers, a dedicated Slack channel has been used.

To briefly comment on the overall findings of this project, the Cure53 team noted that a positive impression gained during the earlier Teleport pentest has persisted for this extended assessment of Telekube. In spite of considerable efforts, Cure53 was unable to find vulnerabilities with a security risk levels exceeding that of "Low". This outcome should be read as a very good sign of treating security as a key component of internal operations and development. In the following sections, this report will first discuss the coverage and scope in more details. Then, each finding is explored on a case-by-case basis, alongside relevant mitigation and fix advice. Lastly, a concluding statement expands on the overall security situation of the Gravitational Telekube software compound.

Scope

- **Telekube Web & CLI Clients**
- **Telekube Kubernetes / Docker Integration**
- **Telekube Architecture & Code Review**
    - http://gravitational.com/telekube/
    - https://gravitational.com/docs/overview/
    - https://github.com/gravitational/gravity (access was granted)
- **Ops Center**
    - https://cure53.gravitational.io
    - SSH access was granted

Test Coverage

This section lists different items of the Telekube project's scope covered during the tests performed by Cure53. The purpose is for the client to gain a better impression of the investigated realms and correlate that information with internal testing routines and documentation.

Furthermore, it is designed to facilitate requesting specific test items that might have not been included in security assessments as of yet, but are still considered necessary under the premise of achieving completeness.

- The web interface was extensively tested for weaknesses in the area of XSS, CSRF, and other classic and novel web bugs. In addition, the code was thoroughly audited. Documentation for all vulnerabilities and weaknesses spotted in this phase can be found in this report.
- Cure53 analyzed the login routine of the CLIs to determine if they hold up against the OpsCenter. No flaws were found in this realm.
- The testing team further investigated the *packaging* process, focusing on whether certain elements can be used to influence or sabotage the process (e.g. improper *tmp*-files and alike). There were no issues to report in this area.
- Cure53 also analyzed the same aspects for the installation process with no negative results to report.
- Next on the list of items subjected to testing were the cluster communications from within a container inside the same cluster. Two outside calls to Grafana and AWS could be identified, but it was impossible to use them for malicious purposes.
- The team further tried to manipulate the installation packages created to elevate privileges. This has proven futile and no flaws were discovered.

- Cure53 looked at the RBAC mechanisms and tried to gain more privileges with an otherwise restricted user. No flaws were found in this realm.
- All the network interfaces for the Docker-containers and Kubernetes-pods were mapped in order to analyze communications whit in the cluster infrastructure and determine attack surfaces on each given host which could lead to the discovery of security flaws. However, no flaws were discovered at this stage.
- Despite thorough testing, no flaws were found in the implementations for either Kubernetes, IAM, docker or seccomp.
- Cure53 dedicated considerable research to all *pods/containers* for hardcoded secrets and sensitive information. To the best of our knowledge and understanding of the platform, no noteworthy findings emerged here.
- The test team further analyzed the cluster management component (i.e. *gravity*) and checked for security issues or related flaws. For this aspect, once again there were no significant findings.
- The security of applications deployed by customers within the cluster was not in scope of this test and is limited to the security features provided by *kubernetes/docker.*

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in a 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. *TLK-01-001*) for the purpose of facilitating any future follow-up correspondence.

TLK-01-001 Web: Copypaste XSS on MSIE/Edge due to missing Headers *(Low)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

It was noticed that several responses from the *cure53.gravitational.io* domain are missing HTTP security headers and can, for instance, be loaded inside an iframe embedding from arbitrary origins. The response is missing *X-Frame-Options* as well as other security headers.

Example Request:

```
GET /web/config.js HTTP/1.1
Host: cure53.gravitational.io
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:55.0) Gecko/20100101
Firefox/55.0
```

```
Accept: */*
Accept-Language: en-US,en;q=0.5
Referer: https://cure53.gravitational.io/web/password_reset
Connection: close
```

Response Headers:

```
HTTP/1.1 200 OK
Date: Thu, 24 Aug 2017 12:23:18 GMT
Content-Type: text/plain; charset=utf-8
Connection: close
Content-Length: 7777

var GRV_CONFIG = {"systemInfo":{"serverVersion":
{"version":"4.21.0","gitCommit":"87f6d985fd8c43bd63ee5bc6f6e22d51278006ac","gitT
reeState":"clean"},"wizard":false,"clusterName":"cure53.gravitational.io"},"auth
":{"second_factor":"otp","oidc":
[{"name":"google","displayName":"aaaaaaaaaaaaaaaaaaaaaa"}]},"user":
{"allowCreateNew":false,"allowResetPas
[...]
erator":{"enabled":true}}}}};
```

This might seem like an issue of a rather "Low" relevance at first, but it is exploitable through Copy & Paste XSS and a little bit of social engineering. The affected versions are the latest Microsoft Edge tested on the most recent Tech Preview with Edge 40. An attacker would use this to send arbitrary XHR requests and potentially steal sensitive data.

Steps to reproduce:

- Load the PoC shown below as HTML file on MS Edge;
- Copy the"*copy me*" string , which can be done automatically via Clipboard API;
- Paste the string into the iframe below;
- Observe successful XSS on *cure53.gravitational.io.*

PoC:

```
<body>
<script>a=1</script>
<hr>
<div contenteditable>
Select & Copy<span contenteditable="false">
<iframe style="height:0px;width:0px"
src="data:text/html,<script>document.write('<iframe
src=javascript:if(document.domain.match(/gravi/))alert(document.domain)>')</scri
pt>"></iframe>
</a></svg></span>Me...
</div>
```

```
<hr>
And paste me below & click once to win an AngularJS Picnic Basket!
<br>
<iframe src="/" id="ifr" style="height:300px;width:600px"></iframe>
<script>
ifr.contentDocument.designMode='on';
ifr.contentDocument.location='https://cure53.gravitational.io/web/config.js'
</script>
```

The core problem here can be split into two separate issues. The first flaw is that MSIE as well as MS Edge allow to inherit the *designMode* of an existing HTML page on one origin to any other origin the browser navigates to afterwards. This means that an attacker can first put a website on their own origin into *designMode* and then redirect the website to accomplish rendering of (any) other origin in the *designMode* as well. Secondly, both MSIE and MS Edge try to sanitize the Clipboard when it is filled with a HTML bucket from a potentially risky markup. While a wide range of XSS attack vectors and JavaScript execution sinks are successfully being removed, Edge mostly fails to properly sanitize SVG images after copying them into the Clipboard and then pasting them into a different origin. This allows the attacker to sneak active payload past a cross-origin Copy & Paste operation that is otherwise considered safe.

Combining the aforementioned two tricks with an additional social engineering component (i.e. having the user paste a harmless-looking string into a dedicated area, with the logic of "*copy the ball into the basket to win...*") will result in an XSS attack. Notably, JavaScript executes on the domain the user pastes into, here *cure53.gravitational.io.* The accumulation of issues, as well as every problem on its own, has been reported to Microsoft multiple times. However, they have never yielded a fix so, ultimately, this browser-behavior remains highly unlikely to be altered.

It is strongly recommended to deploy the HTTP Security Headers that are used on other resources for error pages **and** static resources consistently. Note that the same trick can also be used when an *image* is being iframed. This is due to the fact that MSIE and Edge will "*upgrade*" the *content-type* or render a HTML scaffolding around an iframed image. It is paramount that all resources are in fact applied with security headers, which are currently limited to HTML documents and areas that are deemed interesting with reference to Clickjacking attacks.

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

TLK-01-002 Web: Arbitrary Redirect during Login allows Phishing *(Low)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

While analyzing the login functionality and its OAuth flow, it was noticed that the OPs Center allows to specify an arbitrary URL when initiating authentication. This can be observed by browsing the following link.

URL PoC:
https://cure53.gravitational.io/proxy/v1/webapi/oidc/login/web?redirect_url=http://evil.com&connector_id=google

After completing the steps required under the OAuth flow, the authentication is confirmed by having the user sent back to *cure53.gravitational.io* together with the authentication token. Once accepted, another redirect to the previously set *redirect_url* happens. An illustration for this process can be consulted through the provided URL.

Last step of the OAuth flow:
https://cure53.gravitational.io/portalapi/v1/oidc/callback?code=eBUI6KJ9MWyRW-LK&state=159852804a5b984402cb97765f0bfb58

Response:
```
HTTP/1.1 302 Found
Location: http://evil.com
Set-Cookie:
session=7b2275736572223a226e696b6f406375726535332e6465222c22736964223a2262616330
38353265656535323035336432356462323734643838366631306230227d; Path=/; HttpOnly;
Secure
Date: Fri, 25 Aug 2017 13:33:01 GMT
Content-Length: 38
Content-Type: text/html; charset=utf-8
Connection: close

<a href="http://evil.com">Found</a>.
```

The problem here lies within the fact that the specified URL is not restricted by a whitelist. Therefore, it allows attackers to specify a domain with, for example, an error message prompting for additional login. This might trick less security-aware users into initiating the login again, but this time the process will take place on an attacker-controlled domain. In essence, the account credentials are being handed over upon being typed-in. The vulnerable code can be viewed in the following excerpt from the application's sources.

Affected File:
/gravity-master/vendor/github.com/gravitational/teleport/lib/web/apiserver.go

Affected Code:
```
func (m *Handler) oidcLoginWeb(w http.ResponseWriter, r *http.Request, p
httprouter.Params) (interface{}, error) {
	log.Infof("oidcLoginWeb start")

	query := r.URL.Query()
	clientRedirectURL := query.Get("redirect_url")
	if clientRedirectURL == "" {
		return nil, trace.BadParameter(
			"missing redirect_url query parameter")
	}
	connectorID := query.Get("connector_id")
	if connectorID == "" {
		return nil, trace.BadParameter(
			"missing connector_id query parameter")
	}
	response, err := m.cfg.ProxyClient.CreateOIDCAuthRequest(
		services.OIDCAuthRequest{
			ConnectorID:       connectorID,
			CreateWebSession:  true,
			ClientRedirectURL: clientRedirectURL,
			CheckUser:         true,
		})
	if err != nil {
		return nil, trace.Wrap(err)
	}
	http.Redirect(w, r, response.RedirectURL, http.StatusFound)
```

As one can see here, the URL is directly taken from the *redirect_url* parameter, without being compared against a whitelist of allowed URLs beforehand. It is thus recommended to add the necessary whitelist-check at the correct moment in the sequence. This will prevent potential attackers from luring victims onto arbitrary URLs in this case.

TLK-01-006 Web: Missing Login Rate Limiting allows for OTP brute-force *(Low)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

It was found that the system does not implement proper rate-limiting on user-login. An attacker can perform unlimited attempts to login to an account, which introduces issues like brute-forcing of user-credentials. More critically, it leads to OTP brute-forcing. This basically renders the 2FA futile since the entropy of the OTP value is rather small.

It is recommended to implement proper rate-limiting per account to prevent attackers from guessing the correct password and/or the OTP of an account.

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.

TLK-01-003 Web: Login CSRF due to state parameter not validated *(Low)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

Following the discovery of TLK-01-002, it was later found that the last step of the OAuth flow does not validate whether the *state* parameter belongs to the user initiating the authentication process. This allows an attacker to force a user into being authenticated as the attacker.

Steps to Reproduce:

1. Login in via https://cure53.gravitational.io/proxy/v1/webapi/oidc/login/web?redirect_url=https%3A%2F%2Fcure53.gravitational.io%2Fweb&connector_id=google
2. While completing the authentication process, intercept and stop the request to https://cure53.gravitational.io/portalapi/v1/oidc/callback?code=DzmPKmDttgqFAs1W&state=0e30f4c7445fdd7ede2abd0e78018669
3. Use another browser profile to visit the above link to simulate another user accepting the link.
4. Observe being logged in into another browser profile

Despite the presence of the *state* parameter in the last step, it appears that it is not being validated. Notably, the OAuth 2.0 Specification enforces[1] this protection against CSRF attack. Failure to comply can result in Login CSRF, which might not be directly exposed to exploitation, but might aid attackers in seeking and amassing other low severity issues, such as self-XSS.

It is recommended to actually validate the *state* parameter.

[1] https://tools.ietf.org/html/rfc6749#section-10.12

TLK-01-004 Web: Missing CSRF protection on logout endpoint *(Low)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

It was found that the logout endpoint at */web/logout* has no protections against CSRF attacks. This enables an attacker to invalidate a user's session and log them out. Additionally, the */proxy/*-route is able to achieve the same result, also by invalidating the user's session.

PoC:
https://cure53.gravitational.io/web/logout
https://cure53.gravitational.io/proxy/v1/webapi/sites/cure/connect

Reponse:
```
HTTP/1.1 302 Found
Location: /web/login
Set-Cookie: grv_grafana=; Path=/; Max-Age=0; HttpOnly; Secure; SameSite=Strict
Set-Cookie: session=; Path=/; HttpOnly; Secure
Date: Mon, 28 Aug 2017 16:03:21 GMT
Content-Length: 33
Content-Type: text/html; charset=utf-8

<a href="/web/login">Found</a>.
```

By simply having a user visit the endpoint, one can see their session cookie being removed. Although this issue can only cause annoyance rather than actual security risks, it is nevertheless recommended to fix it by implementing CSRF protection on the affected endpoint.

TLK-01-005 Web: CSP bypass with Angular bundled with Grafana *(Info)*

Note: This issue was not fixed, the risk was accepted by Gravitational.

Several pages deployed by the tested application make use of the Content Security Policy to mitigate XSS attacks. It was found the application, however, offers resources which can be used for bypassing CSP and rendering it useless. Those resources are various versions of AngularJS bundled with Grafana. In this context, it should be mentioned that recent research by Cure53[2] and Google[3] demonstrated that AngularJS can be successfully used to bypass CSP.

[2] https://www.slideshare.net/x00mario/an-abusive-relationship-with-angularjs
[3] https://github.com/google/security-research-pocs/blob/48...ular_exploit.php

Affected Files:
Note: Referrer from *https://cure53.gravitational.io* is required.

https://cure53.gravitational.io/web/grafana/public/app/boot.16792590.js
https://cure53.gravitational.io/web/grafana/public/vendor/angular/angular.js
https://cure53.gravitational.io/web/grafana/public/vendor/angular/angular.min.js

PoC:
```
<script
src="https://cure53.gravitational.io/web/grafana/public/app/boot.16792590.js"></
script>
<div ng-app ng-csp>
  <div ng-click="x=$event">CLICK</div>
  <div ng-repeat="(key, value) in x.view">
    <div ng-if=key=="window">{{ value.alert = [1].reduce(value.alert,
1337) }}</div>
  </div>
</div>
```

Bypassed CSP:
```
Content-Security-Policy:script-src 'self';style-src 'self' 'unsafe-
inline';object-src 'none';img-src 'self' data: blob:;worker-src 'self' blob:
```

It is recommended to move AngularJS to another domain or simply delete it if possible. By revising the approach in this realm, one can ensure that the page protected by CSP stays safe.

TLK-01-007 Web: Password reset link not invalidated after use (*Info*)

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

It was found that the *password reset link* is not invalidated once used. This allows an attacker able to leak the link to reset the password of an account, even though it might have been already employed on the website. A perfect scenario for an attacker would be, for example, to have the link leaked through the HTTP referrer header to an external website, which would mean a possibility to reuse it for the adversary.

In order to prevent potential dangers in this area, it is recommended to invalidate the password reset link immediately upon its usage. This constitutes a best practice approach, as suggested by "*OWASP Periodic Table of Vulnerabilities - Insufficient Password Recovery*[4]".

[4] https://www.owasp.org/index.php/OWASP_Periodic_T...ord_Recovery#Generic_Framework_Solution

TLK-01-008 Web: Login CSRF on login endpoint *(Low)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

It was found that the login endpoint at */proxy/v1/webapi/sessions* is missing CSRF protections. This allows an attacker to force a user into being authenticated as the attacker.

Request:
```
POST https://cure53.gravitational.io/proxy/v1/webapi/sessions HTTP/1.1
Host: cure53.gravitational.io
Connection: keep-alive
Content-Length: 90
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/60.0.3112.113 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8,zh-TW;q=0.6,zh;q=0.4,zh-CN;q=0.2

{"user":"filedescriptor@cure53.de","pass":"filedescriptor","second_factor_token"
:"895203"}
```

Response:
```
HTTP/1.1 200 OK
Content-Type: application/json
Set-Cookie:
session=7b2275736572223a2266696c6564657363726970746f724063757265535332e6465222c22
736964223a2234396531613335373365633234646530386363323865373664653666646263338227d
; Path=/; HttpOnly; Secure
Date: Wed, 06 Sep 2017 14:41:41 GMT
Content-Length: 327

{"type":"Bearer","token":"bc64a44d1c37df451bb64852497ad266","user":
{"email":"filedescriptor@cure53.de","name":"filedescriptor@cure53.de","type":"ad
min","account_owner":false,"account_id":"00000000-0000-0000-0000-
000000000001","site_domain":"","password":"","hotp":null,"allowed_logins":null,"
identities":null},"expires_in":599}
```

It is recommended to deploy CSRF protection on the affected endpoint in order to prevent attackers from abusing it.

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

TLK-01-009 Web: Login CSRF on password reset endpoint *(Low)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

Following the discovery of TLK-01-008, it was also found that the */portalapi/v1/recoveries/complete* password reset endpoint lacks CSRF protections.

Request:
```
POST https://cure53.gravitational.io/portalapi/v1/recoveries/complete HTTP/1.1
Host: cure53.gravitational.io
Connection: keep-alive
Content-Length: 139
Accept: application/json, text/javascript, */*; q=0.01
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/60.0.3112.113 Safari/537.36
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8

{"password":"ZmlsZWRlc2NyaXB0b3I=","hotp_value":"028964","secret_token":"ba79704
a20242abb2fdc27b7176080b2935a1fd1a2a8effbad1d0982c4fe9534"}
```

Response:
```
HTTP/1.1 200 OK
Content-Type: application/json
Set-Cookie:
session=7b2275736572223a2266696c65646573637269970746f72406375726535332e6465222c22
736964223a22666236633364613936613634643266396264353866333761646662303130373922 7d
; Path=/; HttpOnly; Secure
Date: Wed, 06 Sep 2017 15:00:09 GMT
Content-Length: 16

{"message":"OK"}
```

It is recommended to deploy CSRF protection on the affected endpoint to mitigate the potential risks.

TLK-01-010 Web: Email enumeration via password reset *(Info)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

It was discovered that during the initial password reset process, the system will return the information on whether the supplied email address exists as registered in the system. This could ease attacker's efforts when attempting to perform login brute-forcing attacks. In other words, it eliminates the need of confirming if the email is valid or not.

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

A message response to requesting password reset on an existing account:

Confirmation email has been sent to filedescriptor@cure53.de. Please check your Inbox for a link to complete your password reset request.

A message response to requesting password reset on a non-existing account:

user(name="a@a.com") not found

It is recommended not to reveal the information about the existence of an account, as suggested by the *OWASP Testing Guide*[5].

TLK-01-011 Web: Manipulation of installation links *(Info)*

Note: This issue was reported by Cure53 while the pentest was still ongoing. The fix was deployed by Gravitational and successfully verified by Cure53.

Another peculiar behavior was discovered in relation to installation links. Specifically, if an administrator generates an application installation link for a specific application, the receiver of this link will be able to install any of the applications from the "*Application Bundle*". Notably, this means any apps sharing the location from which the administrator generated the installation link. The receiver only needs to know the name of the application and can change the installation link to correspond with another application in the bundle.

Applications available from the Cure53 OPS center:
- *Telekube* (ver. 0.1.0-alpha-49-g510b344);
- *Mattermost* (ver. 2.2.0);
- *Mattermost-2* (ver. 2.2.0).

Steps to Reproduce:

- Generate the installation link for *mattermost*
  https://cure53.gravitational.io/web/installer/new/gravitational.io/mattermost/2.2.0?install_token=322c403fd21990a8becc26e76baacb16
- The recipient of the link can actually change it to install *mattermost-2*
  https://cure53.gravitational.io/web/installer/new/gravitational.io/mattermost-2/2.2.0?install_token=322c403fd21990a8becc26e76baacb16

[5] https://www.owasp.org/index.php/Testing_for_Account_Enum...ble_User_Account_(OTG-IDENT-004)

Both of the above links work but only one link was generated intentionally by the administrator. Clear recommendation here is to generate a token that also specifies which application is meant to be installed with the one-time installation link.

Conclusions

The results of this Cure53 penetration test and audit of the Telekube entities are generally positive, yet must also be contextualized in the broader context of a tremendously complex design and extensive scope. In other words, the impressions shared by nine Cure53 testers involved in the completion of this test in 2017 reinforced the conviction that security is taken seriously at the Gravitational compound. However, it should be emphasized that the nature of the project - especially with reference to the massive scope - does not warrant complete certainty when it comes to judging potential security risks existing on the Telekube platform.

Despite small reservations, it must be stated that the Telekube held itself strong in the face of Cure53's attack attempts. Due to the system foundation based on Docker and Kubernetes, as well as having *golang* as its primary language, the security/resilience of the tested software compound has been assessed as being on an already very high level. Similarly, for the tested RBAC system based on Kubernetes' built-in mechanisms and IAM (Amazon Identity and Access Management), also no major flaws could be identified during the test. The only real and relevant bug pattern that Cure53 could spot in the web application revolved around the misconfiguration of response headers and its resulting vulnerabilities and weaknesses. It appears that the maintainers have a great awareness that the web interface in combination with an XSS vulnerability would wreak havoc. For this arena, it was pivotal that only strange fringe cases were identified, while no XSS issues were found. Given the enforcement of tokens for each request, the same held for CSRF and other classic web attacks and issues.

Although the report discusses eleven findings, among which three were marked as vulnerabilities and eight as general weaknesses, the important indicator is the absence of high-level problems. In fact, neither "Critical", nor "High", nor even "Medium" risks were spotted. At present, a very determined attacker would be required to chain several issues or rely on heavy user-interaction to succeed with a working exploit. Conclusively, best practices and approaches in the security realm are obviously known and implemented by the Gravitational team. This is also evident from a previous and now public report[6] on the Teleport entity, which is a part of the Gravitational family as well.

[6] https://cure53.de/pentest-report_teleport.pdf

As already indicated, one of the biggest remaining concerns is the fact the the code audit could only reach limited coverage. This stems from the fact that the code base is extremely large, so that the time resources allocated to this project only allowed a detailed investigation of selected code parts. The Cure53 strategically focused on critical elements, determining the usefulness and gravity of different components for the overall verdict. Quite clearly, a good recommendation going forward would be to engage in more code audits. Nevertheless, Cure53 suggests that these shall be done in-house rather than consuming valuable resources of external security assessments and penetration tests.

To conclude, it is very much noticeable and praiseworthy that Gravitational has woven security into their processes successfully. The projects exhibit exceptional awareness of the limitations and tailor security strategies to their corresponding threat models. The majority of security risks are averted through proper and top-notch measures of attack mitigation and prevention. Therefore, the Telekube product should be considered safe and in line with its security promises.

Cure53 would like to thank Sasha Klizhentas, Alexey Kontsevoy, Dmitry Shelenin of Gravitational for their excellent project coordination, support and assistance, both before and during this assignment.