CRAFTY REQUESTS

Deep Dive Into a Kubernetes CVE

@IanColdwater

- Hi! I’m Ian Coldwater.
- I’m a Lead Platform Security Engineer at Heroku, specializing in hacking and hardening Kubernetes, containers and cloud infrastructure.
- On December 3, 2018, my phone blew up.

@IanColdwater

BIG NEWS!

EDITION: US

https://www.zdnet.com/article/kubernetes-first-major-security-hole-discovered/

ZDNet

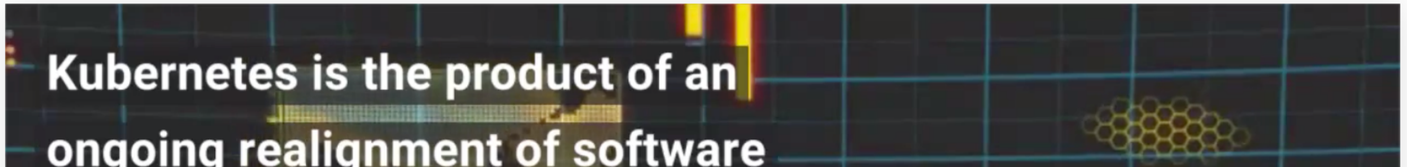
VIDEOS 5G WINDOWS 10 CLOUD AI INNOVATION SECURITY MORE

Kubernetes' first major security hole discovered

There's now an invisible way to hack into the popular cloud container orchestration system Kubernetes.

By Steven J. Vaughan-Nichols for Networking | December 3, 2018 -- 21:17 GMT (13:17 PST) | Topic: Security

7



@IanColdwater

CVE-2018-1002105

- Issue originally discovered by Darren Shepherd and filed directly with Rancher in August 2018 (https://github.com/rancher/rancher/issues/14931)
- Diagnosed by Rancher as a TCP connection reuse issue and reported privately to Kubernetes security team in November 2018
- Publicly disclosed by Kubernetes security team on December 3, 2018

SERIOUSLY, THIS WAS A BIG DEAL

- Lots of press and attention
- High to critical severity vulnerability - 8.8 to 9.8 CVSS v3 score
- Affected ALL Kubernetes versions up to fix

CVE-2018-1002105: proxy request handling in kube-apiserver can leave vulnerable TCP connections #71411

Closed **liggitt** opened this issue on Nov 26, 2018 · 49 comments

liggitt commented on Nov 26, 2018 • edited ▾ Member

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (9.8, critical)

With a specially crafted request, users that are authorized to establish a connection through the Kubernetes API server to a backend server can then send arbitrary requests over the same connection directly to that backend, authenticated with the Kubernetes API server's TLS credentials used to establish the backend connection.

Thanks to Darren Shepherd for reporting this problem.

CVE-2018-1002105 is **fixed** in the following Kubernetes releases:

- v1.10.11
- v1.11.5
- v1.12.3
- v1.13.0-rc.1

Affected components:

- Kubernetes API server

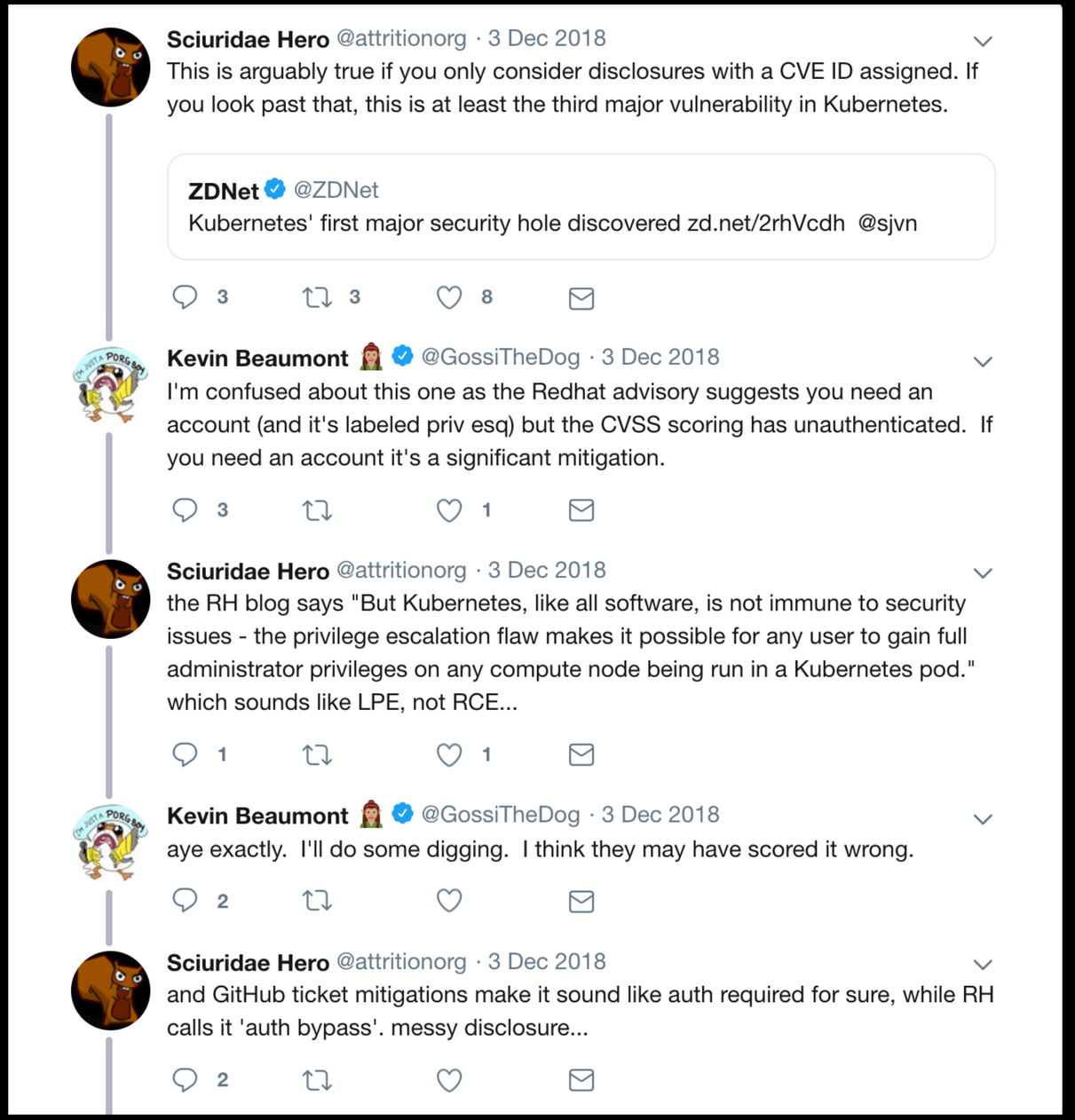
https://github.com/kubernetes/kubernetes/issues/71411

AFFECTED VERSIONS

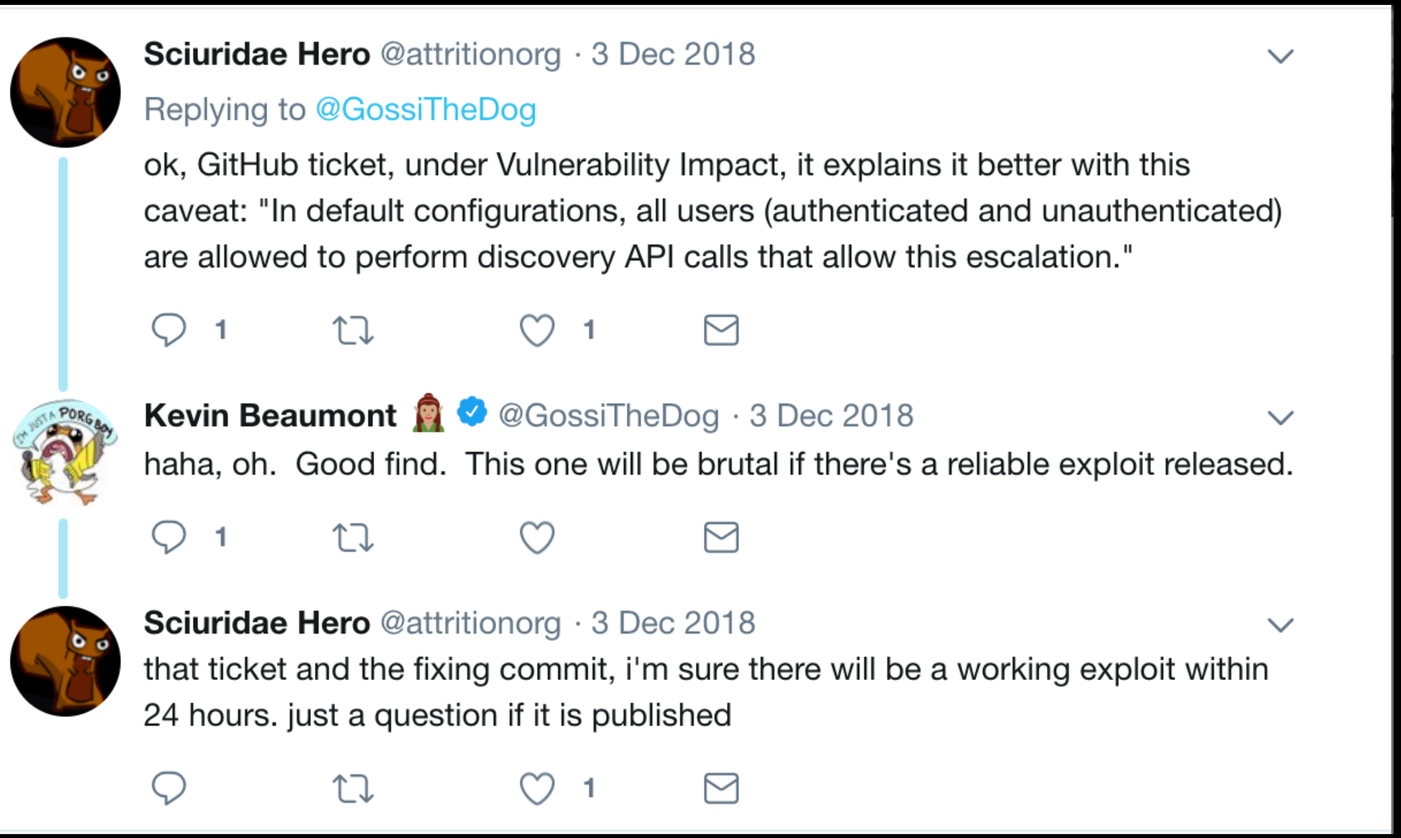
- Kubernetes 1.0.x-1.9.x
- Kubernetes 1.10.0-1.10.10 (fixed in 1.10.11)
- Kubernetes 1.11.0-1.11.4 (fixed in 1.11.5)
- Kubernetes 1.12.0-1.12.2 (fixed in 1.12.3)

THE GOOD NEWS?

- If you were already running everything as admin, no unauthorized user could escalate privileges to admin.
- This is a bad idea. Please don’t actually do this.



https://twitter.com/attritionorg/status/1069737718228246528



https://twitter.com/attritionorg/status/1069741629899079680

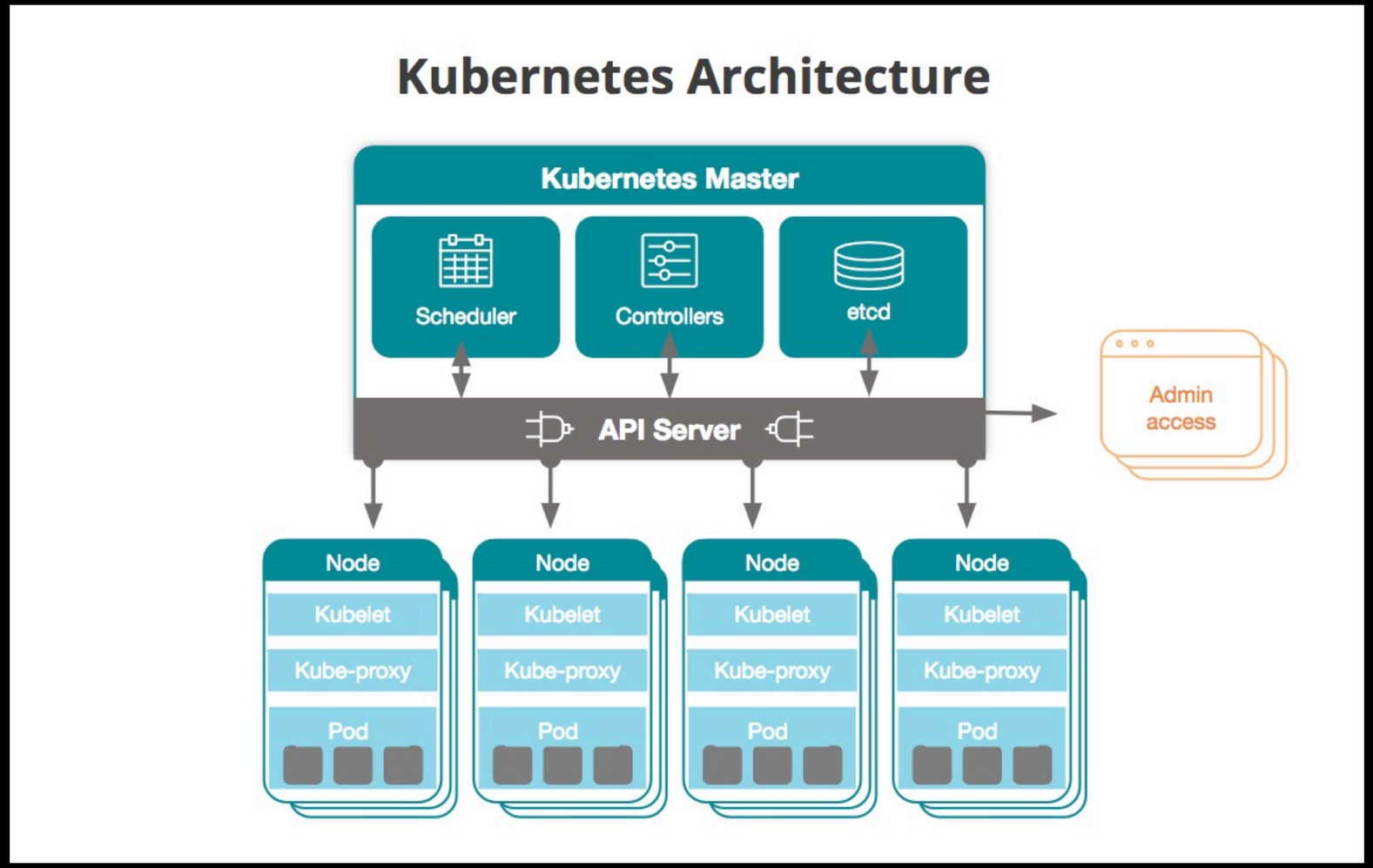
@IanColdwater

HOW DID THIS FLAW WORK?

Connections were allowed to upgrade without checking for error codes, allowing users who sent a specially crafted request to communicate directly with backend servers.

TO FIGURE OUT HOW THIS HAPPENED, LET'S TAKE A LOOK AT THE MOVING PARTS.

KUBERNETES CONTROL PLANE



API SERVER

- provides the REST API endpoint through which Kubernetes operations are made
- acts as a gateway between the user and backend servers, such as extension API servers and kubelets
- accessible to all pods by default

CONNECTION FLOW

- User sends a request to API server
- API server authenticates and authorizes user
- API server uses TLS credentials to establish connection with backend server
- API server acts as a reverse proxy, routing requests between user and backend server

CONNECTION: UPGRADE

- Reverse proxies such as the Kubernetes API Server can upgrade HTTP connections to websockets, which allow back-and-forth communication in a more efficient way than having to constantly open and close connections.

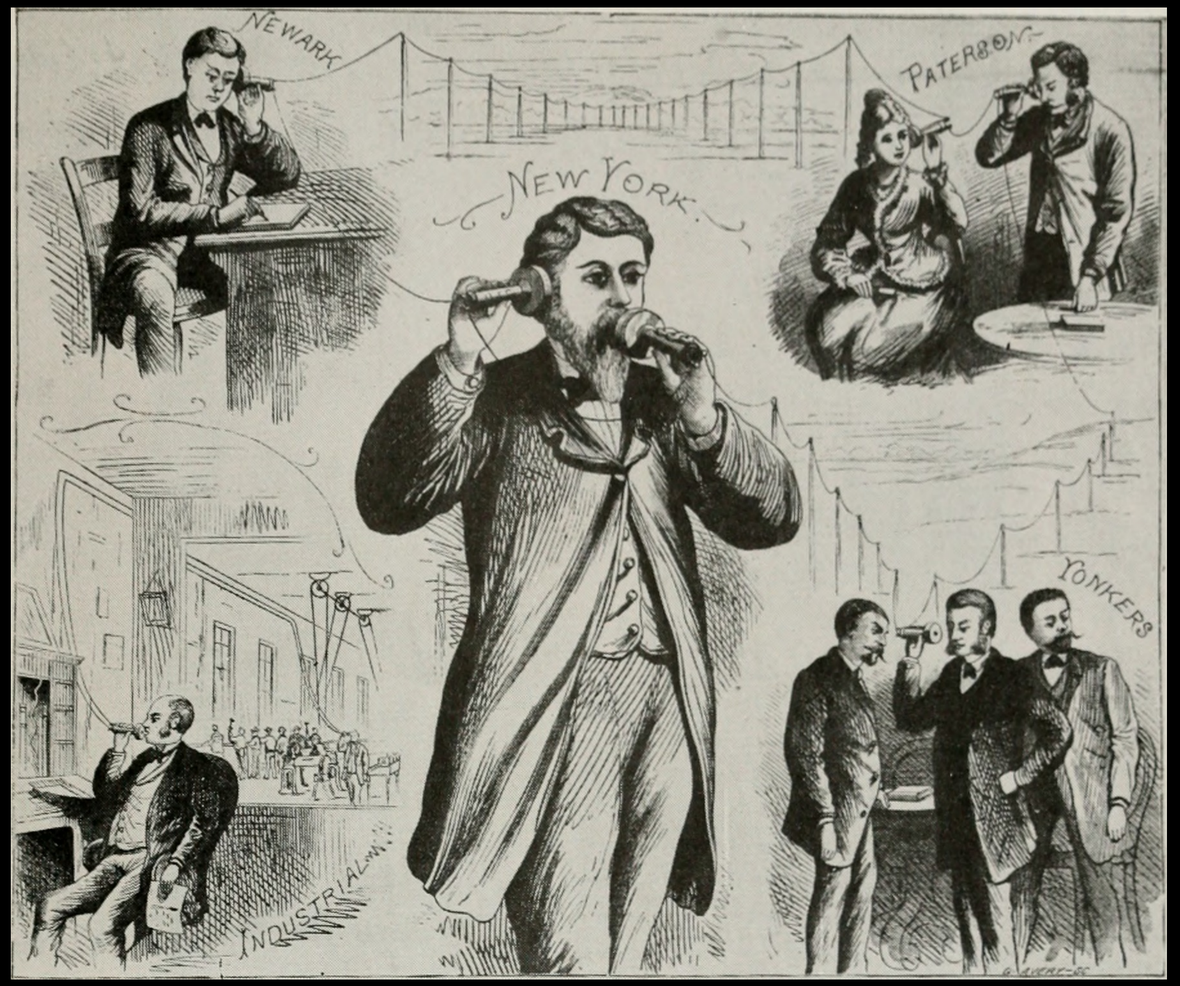
- If a connection upgrade request is sent that isn’t valid, an error code is returned. If the request returns success, a “dumb pipe” is created that remains open.

IN THEORY, THIS WORKS

- This is common behavior in scenarios such as load balancing
- It becomes a problem when dumb pipes meet more complicated gateways that perform actions like routing, authentication and authorization.
- Kubernetes does exactly this, leading to a privilege escalation issue with multiple attack paths.

DUMB PIPES ARE DUMB

- With this flaw, attackers could trick HTTP connections to upgrade to websockets despite returning errors. This allowed them to communicate directly with backend servers, bypassing the API server controls around authorization
- Such traffic is very difficult to detect in logs



ATTACK VECTORS

- exec/attach/port-forward
- extension/aggregated API servers

EXEC/ATTACH/PORT-FORWARD

- users who are authorized to pod exec/attach/port-forward
- can escalate to broader cluster API access via kubelet
- affected all Kubernetes deployments before fixed versions
- CVSS 8.8 (high)

EXTENSION/AGGREGATED API SERVERS

- authorized users for API discovery. By default, this is anybody!
- can escalate to anything on downstream API servers
- only affects deployments with extension API servers
- CVSS 9.8 (critical)!

```
root@kali:~# kubectl get clusterroles system:discovery -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  [redacted]
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:discovery
  resourceVersion: "60"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterroles/system%3Adiscovery
  [redacted]
rules:
- nonResourceURLs:
  - /api
  - /api/*
  - /apis
  - /apis/*
  - /healthz
  - /openapi
  - /openapi/*
  - /swagger-2.0.0.pb-v1
  - /swagger.json
  - /swaggerapi
  - /swaggerapi/*
  - /version
  - /version/
  verbs:
  - get
```

```
root@kali:~# kubectl get clusterrolebindings system:discovery -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  [redacted]
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:discovery
  resourceVersion: "114"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/system%3Adiscovery
  [redacted]
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:discovery
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:unauthenticated
```

WHAT COULD POSSIBLY GO WRONG?

- Cryptomining or other hijacking compute resources
- Exfiltrating secrets or other sensitive data
- Injecting malicious code, supply chain attacks
- Total cluster takeover. Whatever an attacker can dream up!

EXPLOIT DEMO!

From Twistlock: https://asciinema.org/a/215929

PUBLISHED EXPLOITS

- https://github.com/evict/poc_CVE-2018-1002105
- https://github.com/gravitational/cve-2018-1002105
- https://www.exploit-db.com/exploits/46052
- https://www.exploit-db.com/exploits/46053
- https://www.twistlock.com/labs-blog/demystifying-kubernetes-cve-2018-1002105-dead-simple-exploit/
- https://blog.appsecco.com/analysing-and-exploiting-kubernetes-apiserver-vulnerability-cve-2018-1002105-3150d97b24bb
- These are just some published ones. There are more :)

THE FIX

- 37 line change commit https://github.com/kubernetes/kubernetes/pull/71412/files
- Checks for error codes and closes the connection rather than upgrading the connection request if an invalid code is returned.

```
+       // determine the http response code from the backend by reading from rawResponse+backendConn
+       rawResponseCode, headerBytes, err := getResponseCode(io.MultiReader(bytes.NewReader(rawResponse), backendConn))
+       if err != nil {
+               klog.V(6).Infof("Proxy connection error: %v", err)
+               h.Responder.Error(w, req, err)
+               return true
+       }
+       if len(headerBytes) > len(rawResponse) {
+               // we read beyond the bytes stored in rawResponse, update rawResponse to the full set of bytes read from the backend
+               rawResponse = headerBytes
+       }
+
```

```
+       if rawResponseCode != http.StatusSwitchingProtocols {
+               // If the backend did not upgrade the request, finish echoing the response from the backend to the client and return, closing the connection.
+               klog.V(6).Infof("Proxy upgrade error, status code %d", rawResponseCode)
+               _, err := io.Copy(requestHijackedConn, backendConn)
+               if err != nil && !strings.Contains(err.Error(), "use of closed network connection") {
+                       klog.Errorf("Error proxying data from backend to client: %v", err)
+               }
+               // Indicate we handled the request
+               return true
+       }
+
```

https://github.com/kubernetes/kubernetes/commit/b84e3dd6f80af4016acfd891ef6cc50ce05d4b5b

MITIGATIONS

- Update your Kubernetes versions!
- There are other workarounds, but they are disruptive and impractical
- Updating is easiest, most effective and I really hope you've done it by now
- If you are on a public cloud, they updated it for you.

MITIGATING FUTURE FLAWS

- This wasn’t the first and won’t be the last vulnerability like this.
- How can we protect our architecture better?

DEFENSE IN DEPTH

- Operate on a zero-trust model
- Firewalls and gateways aren’t enough on their own

WATCH YOUR DEPENDENCIES

- Supply chain attacks are a real issue, with potentially catastrophic results.

KEEP UP TO DATE!

- Kubernetes moves fast, and security continues to improve.
- Let’s all move fast and improve our security along with it!

WHAT ELSE CAN WE LEARN FROM THIS?

- Kubernetes Product Security postmortem report: https://github.com/kubernetes/kubernetes/files/2700818/PM-CVE-2018-1002105.pdf
- Cloud-native and security folks need to communicate better, on this side of the news cycle.
- Greater understanding is needed all around.

YOU CAN DO IT!

I believe in you.

RESOURCES

- https://github.com/kubernetes/kubernetes/issues/71411
- https://groups.google.com/forum/#!topic/kubernetes-announce/GVllWCg6L88
- https://nvd.nist.gov/vuln/detail/CVE-2018-1002105
- https://www.twistlock.com/labs-blog/demystifying-kubernetes-cve-2018-1002105-dead-simple-exploit/
- https://gravitational.com/blog/kubernetes-websocket-upgrade-security-vulnerability/
- https://rancher.com/blog/2018/2018-12-04-k8s-cve/