

# Using ATT&CK<sup>®</sup> for Containers to Level Up your Cloud Defenses

Jen Burns @snarejen

fwd:cloudsec 2021



# whoami?

## T1033 - System Owner/User Discovery

---

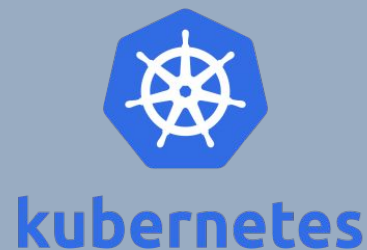
- Senior Security Engineer at HubSpot
- Former ATT&CK for Cloud Lead
- Solid drummer
- Somewhat decent climber
- Lover of bad puns
- Fan of most animals despite being allergic to most animals

**HubSpot**  
Grow Better



# What's on the docke(r)t for today?

1. What is ATT&CK?
2. Why did Containers get added to ATT&CK?
3. How can I use ATT&CK for Containers?



**MITRE | ATT&CK<sup>®</sup>**



# What is ATT&CK?

Note that I am NOT a representative of MITRE or ATT&CK!



# What is ATT&CK?

**A knowledge base of  
adversary behavior**

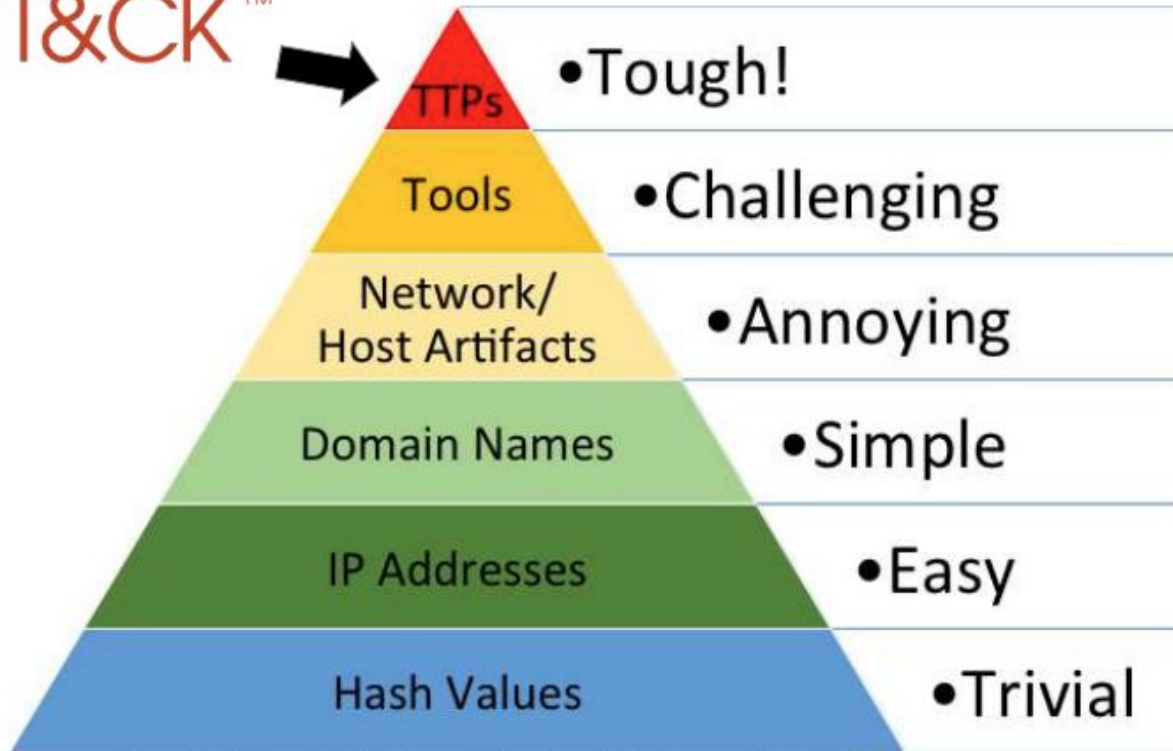
- ***Based on real-world observations***
- ***Free, open, and globally accessible***
- ***A common language***
- ***Community-driven***

Want to contribute to ATT&CK?

- Check out guidelines at <https://attack.mitre.org/resources/contribute/>
- Send your contribution to [attack@mitre.org](mailto:attack@mitre.org)



ATT&CK™



Source: David Bianco, <https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>

## David Bianco's Pyramid of Pain



# Tactics: the adversary's technical goals

Techniques: how the goals are achieved

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Impact
Exploit Public-Facing Application	Container Administration Command	External Remote Services	Escape to Host	Build Image on Host	Brute Force	Container and Resource Discovery	Endpoint Denial of Service
External Remote Services	Deploy Container	Implant Internal Image	Exploitation for Privilege Escalation	Deploy Container	Credential Stuffing	Network Service Scanning	Network Denial of Service
Valid Accounts	Scheduled Task/Job	Scheduled Task/Job	Scheduled Task/Job	Impair Defenses	Password Guessing		Resource Hijacking
Default Accounts	Container Orchestration Job	Container Orchestration Job	Container Orchestration Job	Disable or Modify Tools	Password Spraying		
Local Accounts	User Execution	Valid Accounts	Valid Accounts	Indicator Removal on Host	Unsecured Credentials		
	Malicious Image	Default Accounts	Default Accounts	Masquerading	Credentials In Files		
		Local Accounts	Local Accounts	Match Legitimate Name or Location	Container API		
				Valid Accounts			
				Default Accounts			
				Local Accounts			



# Technique Example

## User Execution

Sub-techniques (3) ^	
ID	Name
T1204.001	Malicious Link
T1204.002	Malicious File
T1204.003	Malicious Image

An adversary may rely upon specific actions by a user in order to gain execution. Users may be subjected to social engineering to get them to execute malicious code by, for example, opening a malicious document file or link. These user actions will typically be observed as follow-on behavior from forms of [Phishing](#).

While [User Execution](#) frequently occurs shortly after [Initial Access](#) it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after [Internal Spearphishing](#).

ID: T1204

Sub-techniques: [T1204.001](#), [T1204.002](#), [T1204.003](#)

- ① **Tactic:** Execution
  - ① **Platforms:** Containers, IaaS, Linux, Windows, macOS
  - ① **Permissions Required:** User
  - ① **Data Sources:** [Application Log](#): Application Log Content, [Command](#): Command Execution, [Container](#): Container Creation, [Container](#): Container Start, [File](#): File Creation, [Image](#): Image Creation, [Instance](#): Instance Creation, [Instance](#): Instance Start, [Network Traffic](#): Network Connection Creation, [Network Traffic](#): Network Traffic Content, [Process](#): Process Creation
- Contributors: Oleg Skulkin, Group-IB
- Version: 1.3
- Created: 18 April 2018
- Last Modified: 20 April 2021

Screenshot from <https://attack.mitre.org/techniques/T1204/>

NOTE: Techniques and sub-techniques ALSO include mitigations, detections, and procedures





Why did Containers  
get added to  
ATT&CK?



# NOT THESE CONTAINERS!!!



ATT&CK®

Matrices Tactics Techniques Mitigations Groups Software Resources Blog Contribute Search Q


Home > Groups > Ever Given

## Ever Given

Ever Given is a Panama-based threat group that laun worldwide. [1]



### Techniques Used

Domain	ID	Name	Use
Containers	T1896	Global Economy Denial of Service	
Containers	T1874	Boat-In-the-Middle	
Containers	T1877	Indicator Removal on Canal	

<https://twitter.com/MITREattack/status/1377615020465520640>



# Why did Containers get added to ATT&CK?

- 1 It's a logical expansion of ATT&CK.

Originally ATT&CK just covered the Windows platform. In the past few years it expanded to include techniques carried out in Linux, macOS, industrial control systems, the cloud, and network devices.



# Why did Containers get added to ATT&CK?

## 2 MITRE Engenuity's CTID provided support.

MITRE Engenuity's Center for Threat-Informed Defense and its members (including Microsoft and the folks who produced their K8s threat matrix) sponsored an investigation into adding containers into ATT&CK and contributed to the platform's creation.



# Why did Containers get added to ATT&CK?

## 3 The community asked for it!

The community also helped conclude that the vast majority of activity that they observed *did* lead to cryptomining. However, evidence from a number of parties led the ATT&CK team to also conclude that adversaries utilizing containers for more “traditional” purposes, such as exfiltration and collection of sensitive data, is publicly under reported.



# ATT&CK for Containers Contributors

---

- Brad Geesaman, @bradgeesaman
- Center for Threat-Informed Defense (CTID)
- Cisco
  - Idan Frimark, Ariel Shuper
- Palo Alto Networks
  - Yuval Avrahami, Jay Chen, Nathaniel Quist
- Rory McCune, Aqua Security
- Team Nautilus Aqua Security
  - Yaniv Agman (@AgmanYaniv), Ziv Karliner (@ziv\_kr), Michael Katchinskiy (@michael64194968), Roi Kol (@roykol1), Assaf Morag (@MoragAssaf), Idan Revivo (@idanr86), Gal Singer (@galsinger29)
- Trend Micro
  - David Fiser (@anu4is), Pawan Kinger (@kingerpawan), Magno Logan (@magnologan), Alfredo Oliveira
- Vishwas Manral, McAfee
- Yossi Weizman, Azure Defender Research Team



<https://www.vecteezy.com/free-vector/industrial>



# How can I use ATT&CK for Containers?



# Four Main Use Cases for ATT&CK

How can ATT&CK for Containers help?

- Assessments and Engineering ←
- Threat Intelligence ←
- Adversary Emulation ←
- Building Detections

Getting Started with ATT&CK blog series:  
<https://medium.com/mitre-attack/getting-started/home>

## Pro Tip!

If it's hard to understand how an ATT&CK technique relates to containers, check out the MITRE Engenuity Medium blogs that have some container specific definitions.

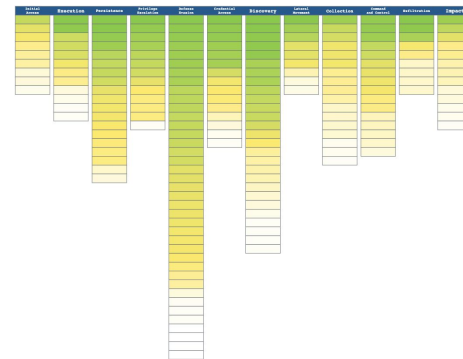
- <https://medium.com/mitre-engenuity/update-help-shape-att-ck-for-containers-bfcd24515df5>
- <https://medium.com/mitre-engenuity/att-ck-for-containers-now-available-4c2359654bf1>



# Assessments and Engineering

How can ATT&CK for Containers help?

- Helps us measure the status of our current defenses
  - *What's our confidence level that we would detect a particular technique?*
  - *Which techniques are sufficiently mitigated?*
  
- Gives us an understanding of where we can improve
  - *Do I need to capture different data sources in my environment?*
  - *Should I buy product X because we aren't detecting a particular set of techniques?*



# Assessments and Engineering Example

## External Remote Services

Adversaries may leverage external-facing remote services to initially access and/or persist within a network. Remote services such as VPNs, Citrix, and other access mechanisms allow users to connect to internal enterprise network resources from external locations. There are often remote service gateways that manage connections and credential authentication for these services. Services such as [Windows Remote Management](#) can also be used externally.

Access to [Valid Accounts](#) to use the service is often a requirement, which could be obtained through credential pharming or by obtaining the credentials from users after compromising the enterprise network.<sup>[1]</sup> Access to remote services may be used as a redundant or persistent access mechanism during an operation.

Access may also be gained through an exposed service that doesn't require authentication. In containerized environments, this may include an exposed Docker API, Kubernetes API server, kubelet, or web application such as the Kubernetes dashboard.<sup>[2][3]</sup>

Consider starting with a single technique....



# External Remote Services

(Testing unauthenticated initial access into your environment)

## 01

---

Exposed Docker  
Daemon/API

External access to port 2375? Is the TCP socket enabled? Is an nmap scan successful? Are you gathering Docker daemon logs for processing?

## 02

---

K8s API Server

Is authentication required to access the API server? Can the API server be reached externally? Are you gathering audit logs for analysis?

## 03

---

Kubelet

Is the kubelet locked down with something like webhook token auth? Are you gathering kubelet logs for analysis?

## 04

---

K8s Dashboard (or  
other web app)

Is the K8s dashboard or other control plane apps externally accessible? Does it require authentication to access?



# Assessments and Engineering Example

Initial Access 3 techniques	Execution 4 techniques	Persistence 4 techniques	Privilege Escalation 4 techniques	Defense Evasion 6 techniques	Credential Access 2 techniques	Discovery 2 techniques	Impact 3 techniques
Valid Accounts <sup>(0/2)</sup>	User Execution <sup>(0/1)</sup>	Valid Accounts <sup>(0/2)</sup>	Escape to Host	Build Image on Host	Brute Force <sup>(0/2)</sup>	Container and Resource Discovery	Network Denial of Service <sup>(0/0)</sup>
External Remote Services	Container Administration Command	External Remote Services	Valid Accounts <sup>(0/2)</sup>	Impair Defenses <sup>(0/1)</sup>	Unsecured Credentials <sup>(0/2)</sup>	Network Service Scanning	Resource Hijacking
Exploit Public-Facing Application	Deploy Container	Implant Internal Image	Exploitation for Privilege Escalation	Valid Accounts <sup>(0/2)</sup>			Endpoint Denial of Service <sup>(0/0)</sup>
	Scheduled Task/Job <sup>(0/1)</sup>	Scheduled Task/Job <sup>(0/1)</sup>	Scheduled Task/Job <sup>(0/1)</sup>	Indicator Removal on Host <sup>(0/0)</sup>			
				Masquerading <sup>(0/1)</sup>			
				Deploy Container			

2 - High Confidence of Detection  
 1 - Low Confidence of Detection  
 0 - No Confidence of Detection

...or maybe

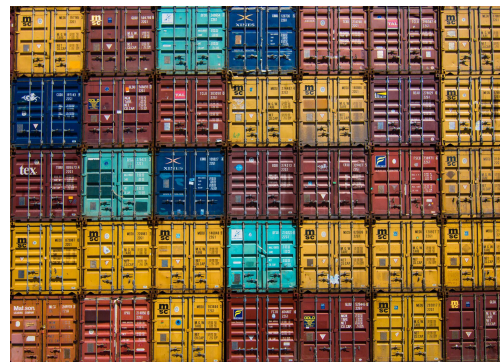
2 - Sufficiently Mitigated  
 1 - Partially Mitigated  
 0 - Not Mitigated



# Threat Intelligence

How can ATT&CK for Containers help?

- Gives us a common language across teams
  - *Red Team*
  - *Blue Team*
  - *Cyber Threat Intel Team*
  - *Cloud Security/Operations Team*
  - *CISO*
  - ...
  
- Helps us better understand what behaviors adversaries are actually doing in Kubernetes, Docker, etc.



[https://unsplash.com/photos/uBe2mknURG4?utm\\_source=unsplash&utm\\_medium=referral&utm\\_content=creditShareLink](https://unsplash.com/photos/uBe2mknURG4?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink)



# Threat Intelligence Example

## Initial Access

`kubelet` is an agent running on each Kubernetes node. It takes RESTful requests from various components (mainly `kube-apiserver`) and performs pod-level operations. Depending on the configuration, `kubelet` may or may not accept unauthenticated requests. Standard Kubernetes deployments come with `anonymous access to kubelet` by default. However, most managed Kubernetes services such as Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE) and Kubernetes operations (Kops) all enforce proper authentication by default.

We discovered that TeamTNT gained initial access with the Hildegard malware by `executing commands on kubelets that allow anonymous access`. This was achieved by accessing the `kubelet's run command API` and executing commands on running containers.

## Execution

Hildegard `uses kubelet's API to execute commands inside containers`. The initial commands create a `tmate reverse shell` that allows the attacker to carry out the subsequent operation. Unlike the techniques that TeamTNT used in the past, this malware campaign did not pull or run any new container image.

## Privilege Escalation

Although Unit 42 researchers have not observed an attempt to `perform privilege escalation`, the malware dropped two adversarial tools, `Peirates` and `BCG`, which are capable of `breaking out of containers via known vulnerabilities` or accessing cloud resources via `exposed cloud credentials`.

Excerpt from Palo Alto Networks about the Hildegard malware from TeamTNT

<https://unit42.paloaltonetworks.com/hildegard-malware-teamtnt/>

T1133 - External Remote Services

T1609 - Container Administration Command

T1611 - Escape to Host

T1552.005 - Unsecured Credentials:  
Cloud Instance Metadata API



# Adversary Emulation

How can ATT&CK for Containers help?

- Helps ensure that you can actually detect/mitigate what you expect
- Provides a sanity check for your red team on what behaviors are being carried out by adversaries in-the-wild



[https://unsplash.com/photos/ayog-lKRp8?utm\\_source=unsplash&utm\\_medium=referral&utm\\_content=creditShareLink](https://unsplash.com/photos/ayog-lKRp8?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink)



# Adversary Emulation Example - Kinsing Malware

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	Command and Scripting Interpreter	Account Manipulation	Abuse Elevation Control Mechanism	Abuse Elevation Control Mechanism	Brute Force	Account Discovery	Exploitation of Remote Services	Archive Collected Data	Application Layer Protocol	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	Visual Basic	Boot or Logon Autostart Execution	Boot or Logon Autostart Execution	Build Image on Host	Credentials from Password Stores	Browser Bookmark Discovery	Internal Spearphishing	Audio Capture	DNS	Data Transfer Size Limits	Data Destruction
External Remote Services	JavaScript	Boot or Logon Initialization Scripts	Boot or Logon Initialization Scripts	Dashboard/Device File or Information	Exploitation for Credential Access	Container and Resource Discovery	Lateral Tool Transfer	Automated Collection	Mail Protocols	Exfiltration Over Alternative Protocol	Data Encrypted for Impact
Hardware Additions	Unix Shell	Browser Extensions	Create or Modify System Process	Deploy Container	Forge Web Credentials	File and Directory Discovery	Remote Service Session Hijacking	Clipboard Data	Web Protocols	Exfiltration Over C2 Channel	Data Manipulation
Phishing	Python	Compromise Client Software Library	Escape to Host	Execution Guardrails	Input Capture	Network Service Scanning	Remote Services	Data from Information Repositories	File Transfer Protocols	Exfiltration Over Other Network Medium	Defacement
Supply Chain Compromise	Container Administration Command	Create Account	Event Triggered Execution	Exploitation for Defense Evasion	Man-in-the-Middle	Network Share Discovery	VNC	Data from Local System	Communication Through Removable Media	Exfiltration Over Physical Medium	Disk Wipe
Trusted Relationship	Deploy Container	Create or Modify System Process	Exploitation for Privilege Escalation	File and Directory Permissions Modification	Modify Authentication Process	Network Sniffing	SSH	Data from Network Shared Drive	Data Encoding	Exfiltration Over Web Service	Endpoint Denial of Service
Valid Accounts	Exploitation for Client Execution	Event Triggered Execution	Hijack Execution Flow	Host-to-Host Remote Malware	Network Sniffing	Password Policy Discovery	Software Deployment Tools	Data from Removable Media	Data Obfuscation	Scheduled Transfer	Firmware Corruption
	Native API	External Remote Services	Process Injection	Hide Artifacts	OS Credential Dumping	Permission Groups Discovery		Data Staged	Dynamic Resolution	Inhibit System Recovery	
	Scheduled Task/Job	Hijack Execution Flow	Scheduled Task/Job	Impair Defenses	Steal Web Session Cookie	Process Discovery		Input Capture	Encrypted Channel	Network Denial of Service	
	At (Linux)	Implant Internal Image	At (Linux)	Indicator Removal on Host	Two-Factor Authentication Interception	Remote System Discovery		Man-in-the-Middle	Fallback Channels	Resource Hijacking	
	Cron	Modify Authentication Process	Cron	Masquerading	Unsecured Credentials	Software Discovery		Screen Capture	Ingress Tool Transfer	Service Stop	
	Systemd Timers	Pre-OS Boot	Systemd Timers	Masquerading	Bash History	System Information Discovery			Multi-Stage Channels	System Shutdown/Reboot	
	Container Orchestration Job	Scheduled Task/Job	Container Orchestration Job	Masquerading	Private Keys	System Location Discovery			Non-Application Layer Protocol		
	Software Deployment Tools	At (Linux)	Valid Accounts	Obfuscated Files or Information	Credentials in Files	System Network Configuration Discovery			Non-Standard Port		
	User Execution	Cron	Pre-OS Boot	Pre-OS Boot	Container API	System Network Connections Discovery			Protocol Tunneling		
		Systemd Timers	Process Injection	Process Injection	System Owner/User Discovery	System Owner/User Discovery			Proxy		
		Container Orchestration Job	Rootkit	Rootkit	Subvert Trust Controls	Subvert Trust Controls			Remote Access Software		
		Server Software Component	Traffic Signaling	Traffic Signaling	Traffic Signaling	Traffic Signaling			Traffic Signaling		
		Traffic Signaling	Valid Accounts	Valid Accounts	Valid Accounts	Valid Accounts			Web Service		

Carry out an engagement across Containers and Linux  
<https://attack.mitre.org/software/S0599/>

Use procedure examples from ATT&CK and threat intelligence as your guide!



# Adversary Emulation/Assessments

There are lots of tools to help!

---

- [red-kube](#) - collection of kubectl commands written to evaluate security posture of K8s clusters
- [kube-hunter](#) - tool that searches for security weaknesses in K8s clusters
- [kubernetes-goat](#) - intentionally vulnerable cluster to learn/practice K8s security
- [kubescape](#) - tool for testing if K8s is deployed securely as defined in [Kubernetes Hardening Guidance by NSA and CISA](#)
- [kube-bench](#) - checks K8s deployment against the CIS K8s Benchmark



# Thank you

Jen Burns  @snarejen

<https://bit.ly/fwdcs21-burns>

