BISHOPFOX

EFFICACY OF MICRO-SEGMENTATION ASSESSMENT REPORT

JUNE 04, 2020

BISHOP FOX
CONTACT INFORMATION

+1 (480) 621-8967

contact@BishopFox.com

8240 S. Kyrene Road Suite A-113 Tempe, AZ 85284

EFFICACY OF MICRO-SEGMENTATION

TABLE OF CONTENTS

EXECUTIVE REPORT

PROJECT OVERVIEW

Illumio Inc. engaged Bishop Fox to measure the effectiveness of micro-segmentation using the Illumio Adaptive Security Platform (ASP) as a control in limiting lateral network movement. The following report details the findings identified during the course of the engagement, which started on March 16, 2020.

GOALS

- Create a repeatable testing methodology that can be leveraged by third parties looking to replicate testing in their own environment
- Record time required to reach the trophy in each use case test
- Review the level of detectable network traffic generated as a result of increased micro-segmentation
- Determine the overall efficacy of micro-segmentation as it relates to the generation of detectable events and time investment required for an attacker to traverse the network

SCOPE

Illumio ASP platform

DATES

03/16/2020
Kickoff

03/23/2020 – 04/08/2020
Active testing

05/12/2020
Report delivery

ABSTRACT

Attackers spend a great deal of time on lateral movement during a breach — as they surf a network, attempting to find the 'trophies' they are after — and networks with little or no control over this movement provide an easy pathway for an attacker to their intended target. This means that once an adversary enters a network through a beachhead, a weak or insecure target used as a launching point (be it an endpoint, a workload, a server, etc.), they act as a burglar in a building where all the doors are open, calmly moving from room to room, picking up anything of value.

Lateral movement is also why malware and, in particular, ransomware can have such a crippling effect on an entire organization. All high-profile ransomware attacks exploit that same freedom of lateral movement around the network in order to spread at a devastating pace and bring a network to its knees.

Zero Trust, and specifically micro-segmentation as a capability in a Zero Trust security framework, is focused on hindering this freedom of lateral movement. Micro-segmentation forces attackers (and malware) to work harder and smarter. In the best case, micro-segmentation can nullify the threat, and in the worst case, all that increased activity leads to increased opportunity for detection by the defender.

EXECUTIVE REPORT

It is widely understood that micro-segmentation controls hamper lateral movement, but by how much? How effective are various types of micro-segmentation policy in thwarting an attacker, and do they force any changes in behavior? This is precisely what this assessment looked to measure.

The results of this engagement highlighted the importance of implementing micro-segmentation in real world environments. Overall, the team identified that the time needed to gain access to sensitive information (i.e., obtain the “crown jewels”) increased quantifiably as more strict micro-segmentation controls were enabled on the tested environments, showing a clear, measurable benefit by forcing the attacker to exhaust more time in order to access sensitive information and resulting in the generation of more detectable events, providing a blue team with a better opportunity to detect the attack.

SUMMARY OF TESTING

The assessment team performed a succession of attack simulations on network environments created by Illumio to measure the effectiveness of the product’s micro-segmentation controls against a series of attacks.

To perform the measurements, the team retained the following indicators prior to the start of the testing activities:

- Time for an attacker to obtain the “crown jewels” (referred to as *Time to Completion*)
- Number of hosts/workloads identified on the network
- Number of services identified on the network

The team ran attack simulations in Illumio-controlled environments, which possessed the following properties:

- All workloads had the Illumio VEN agent installed and were paired with the Management Platform (the Policy Compute Engine or PCE)
- The workload size was identical for the first four tests
- Two tests were repeated with higher workload sets to determine the impact to the Time to Completion metric

To obtain meaningful measurements, four test cases were defined:

- **Control Test:** Represented a flat network with no segmentation.
- **Use Case 1 –** Environmental Separation: Production workloads can only communicate to other Production workloads, Development workloads can only communicate to other Development workloads.
- **Use Case 2 –** Application Ring Fencing: Workloads associated with the same application and within the same environment can communicate with each other – e.g., Ordering Application / Production workloads can only communicate to other Ordering Application / Production workloads.
- **Use Case 3 –** Tiered Segmentation: Workloads associated with a specific tier within a specific application and environment can communicate with each other – e.g., Web Tier / Ordering Application / Production workloads can only communicate to other Web Tier / Ordering Application / Production workloads.

In an attempt to limit bias during the testing, the assessment team was not made aware of the policies implemented for the different use cases (except for the control test, which was representative of a flat network and had no micro-segmentation policy applied). The micro-segmentation policies implementation details are described later in this report.

For the attack simulations, the assessment team developed a methodology based on the main components of the MITRE ATT&CK® framework, in an attempt to map their activities against documented Tactics, Techniques, and Procedures (TTPs) used in real world scenarios. The details of this methodology can be found in the Methodology section of this report.

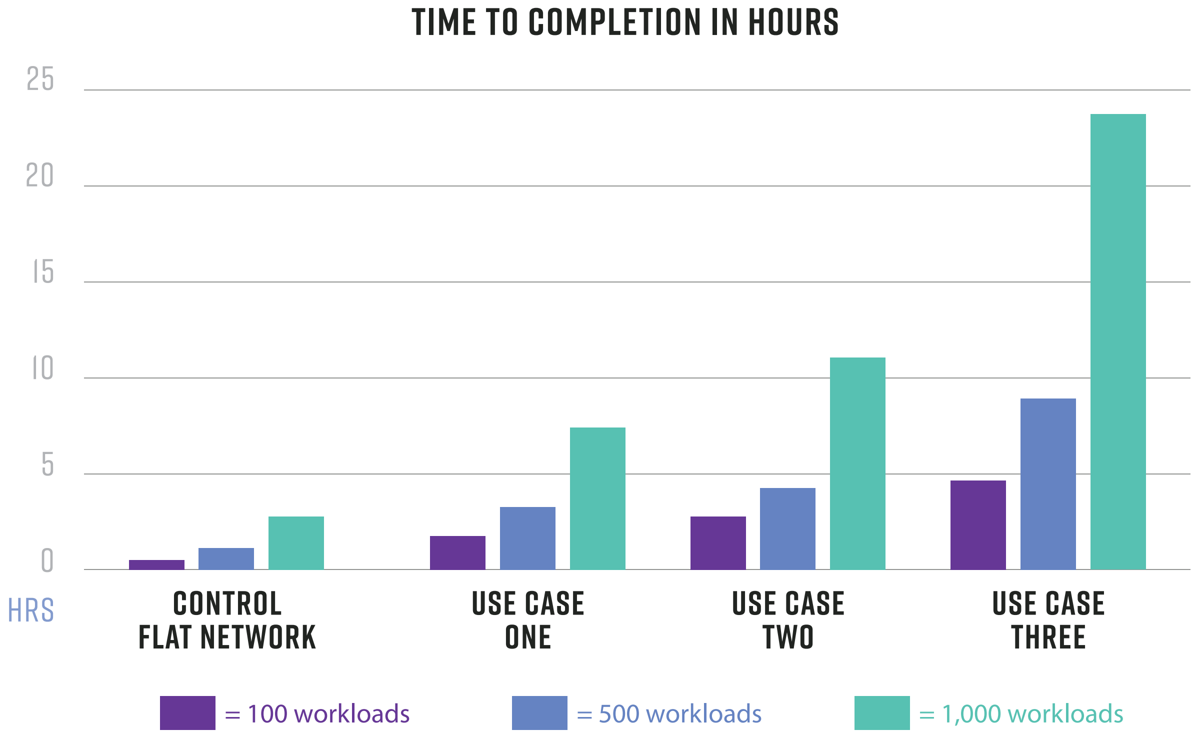


FIGURE 1 – RESULTS AND EXTRAPOLATIONS BASED ON TEST OUTCOMES

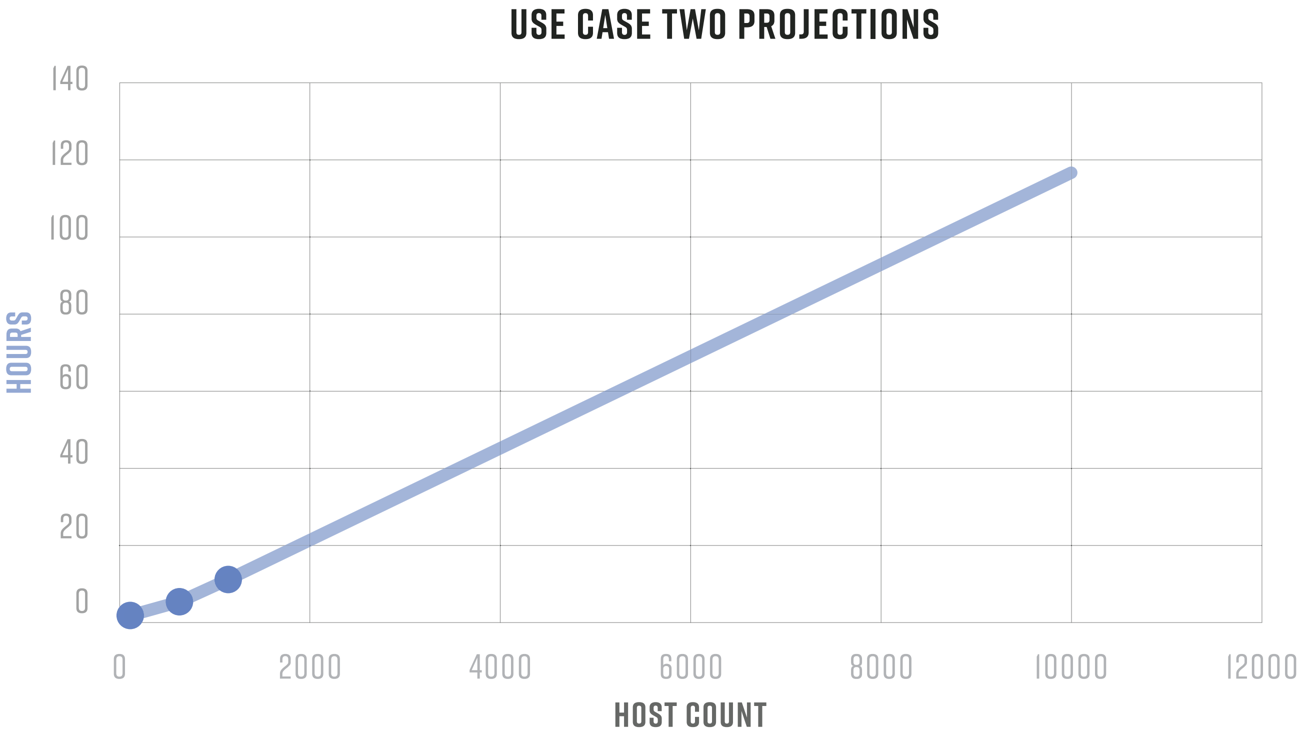


FIGURE 2 – PROJECTIONS BASED ON USE CASE 2 OUTCOMES

100 WORKLOAD TEST RESULTS

TEST	IDENTIFIED HOSTS	IDENTIFIED PORTS	REQIURED TIME (HRS)	TOTAL CONNECTIONS	BLOCKED CONNECTIONS	ALLOWED CONNECTIONS
CONTROL	99	293	0.5	13,361,949	0	13,361,949
USE CASE 01	91	219	1.5	7,705,052	16,902	7,688,150
USE CASE 02	99	173	2.25	11,905,973	64,033	11,841,940
USE CASE 03	99	130	4.75	187,564	181,772	5,792

FIGURE 3 – RESULTS OF TESTING 100 WORKLOAD ENVIRONMENTS

The assessment team observed that for a 100 workloads environment, it would respectively take an attacker 300% (Use Case 1), 450% (Use Case 2), and 950% (Use Case 3) more time to obtain the crown jewels, compared to the control environment (flat network scenario).

The team identified that although the number of discovered hosts on the network varied between the different test cases, they were able to successfully enumerate all of them. However, the number of discovered services linearly decreased by 44% between the control environment test and Use Case 3 (from 293 identified services to 130).

The change in scope tests (adding more workloads to the environment), applied only to the second use case micro-segmentation policy, revealed an increased time to obtain the crown jewels of 189% for 500 workloads (i.e., 8.5x the control environment test) and 489% for 1,000 workloads environments (i.e. 22x the control environment test).

In short, the tighter the micro-segmentation policy (from control to Use Case 1 to Use Case 2 to Use Case 3), the higher the “tax” on the adversary as they attempt to move laterally. Additionally, there is significant measurable benefit in consistently applying micro-segmentation policy across an entire enterprise even if the level of micro-segmentation (the granularity of the policy) is kept constant.

TEST ENVIRONMENT & POLICIES

This was an industry-first testing methodology that was developed from scratch with the goal of facilitating repeatability. This section describes the properties of the test environment and policies used during the testing activities.

ENVIRONMENT OVERVIEW

It is important to note that none of the below information was communicated to the assessment team at any time by Illumio, nor were any details about the test environment (including the number and type of workloads). This decision was made to avoid bias during the testing activities. Details listed below were either discovered during the testing or were revealed post-testing.

The test environment in which the team conducted the attack simulations consisted of the following components:

- **Management Platform:** Illumio Policy Compute Engine (PCE) deployed as a Single Node Cluster (SNC) in AWS
- **Data Centre Workloads (Test Range):** Linux Centos AWS EC2 instances with Illumio VEN agent deployed

The test range comprised:

- One public-facing jump host (jump server or bastion host), directly accessible from the internet
- Application workloads deployed on private subnets

The Illumio VEN agent was installed on all workloads in the test range and was paired with the Management Platform.

The test range was destroyed and completely rebuilt after each round of testing. All hosts in the test range were assigned dynamic IP addresses, which prevented the team from reusing host information between test cases.

To simulate a real network, the workloads were grouped by applications. Each application possessed the following type of workloads:

- **Development:** single workloads running Apache, Tomcat, and Postgres
- **Production:**
  - » Separate workloads for Web (Apache), Processing (Tomcat), and Database (Postgres) tiers
  - » Apache was listening on 80/tcp and 443/tcp, Tomcat on 8080/tcp, and Postgres on 5432/tcp
- **Application jump hosts:** a jump host inside an application bubble
- **All workloads allowed SSH on either 22/tcp or 1357/tcp**

The test environment did not include real applications served by the workloads, but only unconfigured services exposed to the network. Therefore, the assumed way to pivot inside the test environment was to use SSH, as all the hosts shared the same credentials. These credentials also allowed `root` access on every host by using the `sudo` command.

The decision was made to use the same credentials on all the hosts to simplify the environment and limit any variables that might skew the result metrics as the goal was to test the impact of micro-segmentation (vs. trying to test the assessment team's ability to harvest credentials).

The "crown jewels" (data targeted by an attacker) were materialized using fake personally identifiable information (PII) stored by a Postgres server in the test environment. As no real application was deployed in the environment, the Illumio team stored instructions in a file for each test case that contained a clue indicating what the trophy was.

The content of this file was the following:

```
#Postgres commands
sudo -u postgres psql #local connection
\l #list all databases
\c crown_jewels #crown_jewels db
select * from users; #get ssn
```

FIGURE 4 – CONTENT OF THE `README.txt` FILE LEAVING A CLUE FOR THE ATTACKER

This file simulated a configuration file containing credentials and was the first checkpoint in the attack scenario. Once the attacker obtained it, their goal was to find the Postgres server hosting the `crown_jewels` database and extract its data.

POLICIES OVERVIEW

This section describes the different policies applied to the test environment. For every policy, all network flows (allowed and blocked) were logged and forwarded to the PCE, and then to a SIEM server on Illumio's infrastructure.

CONTROL TEST

For the control test, all the workloads had their Illumio VEN agent configured in "Build Mode." This is a non-blocking mode where all ingress and egress connections are allowed by `iptables.` Finally, the policy in place was the "Allow All" policy, although it had no impact since the VEN agent was configured in Build Mode.

This policy was put in place to simulate the activities of an attacker in a flat network and to serve as a reference to compare the metrics obtained with progressively tighter micro-segmentation policies.

USE CASE 1 - ENVIRONMENTAL SEPARATION

In this use case, all workloads had their Illumio VEN agent configured in "Enforce Mode." This is a blocking mode in which `iptables` is programmed to allow only ingress and egress traffic that has been explicitly allowed by the policy defined at the PCE, and all other connections are blocked.

The policy for this use case was defined as follows:

- SSH access from the internet to the public jump host
- SSH access from the public jump host to the individual application jump hosts
- All Production workloads can communicate with each other
- All Development workloads can communicate with each other
- Individual Application jump hosts can SSH to any other workload within the same Application
- All other traffic is denied

USE CASE 2 – APPLICATION RINGFENCING

In this use case, all workloads had their Illumio VEN agent configured in "Enforce Mode". This is a blocking mode in which `iptables` is programmed to allow only ingress and egress traffic that has been explicitly allowed by the policy defined at the PCE, and all other connections are blocked.

The policy for this use case was defined as follows:

- SSH access from the internet to the public jump host
- SSH access from the public jump host to the individual application jump hosts
- Individual Application jump hosts can SSH to any other workload within the same Application
- All Development workloads within a specific Application can talk to all other Development workloads within the same application without restrictions
- All Production Workloads within a specific Application can talk to all other Production Workloads within the same application without restrictions
- All other traffic is denied

USE CASE 3 – TIER BASED SEGMENTATION

In this use case, all workloads had their Illumio VEN agent configured in "Enforce Mode." This is a blocking mode where `iptables` is programmed to allow only ingress and egress traffic that has been explicitly allowed by the policy defined at the PCE, and all other connections are blocked.

The policy for this use case was defined as follows:

- SSH access from the internet to the public jump host
- SSH access from the public jump host to the individual application jump hosts
- All Production workloads within a specific tier (Web, Processing, Database) in a specific Application can talk to all other Production workloads in the same tier of that application without restrictions
- All Development workloads within a specific tier (Web, Processing, Database) in a specific application can talk to all other Development workloads in the same tier of that application without restrictions
- Traffic allowed from any source to ports 80/tcp or 443/tcp in the Development Web Tier
- Traffic allowed from any source to ports 80/tcp or 443/tcp in the Production Web Tier
- Traffic allowed from an application in the Production Web Tier to the same application in the Production Processing Tier on port 8080/tcp
- Traffic allowed from an application in the Production Processing Tier to the same application in the Production Database Tier on 5432/tcp
- Application jump hosts can SSH to any other workload within the same application, except in the cases of five Production applications, where the SSH paths are described below
- All other traffic is denied

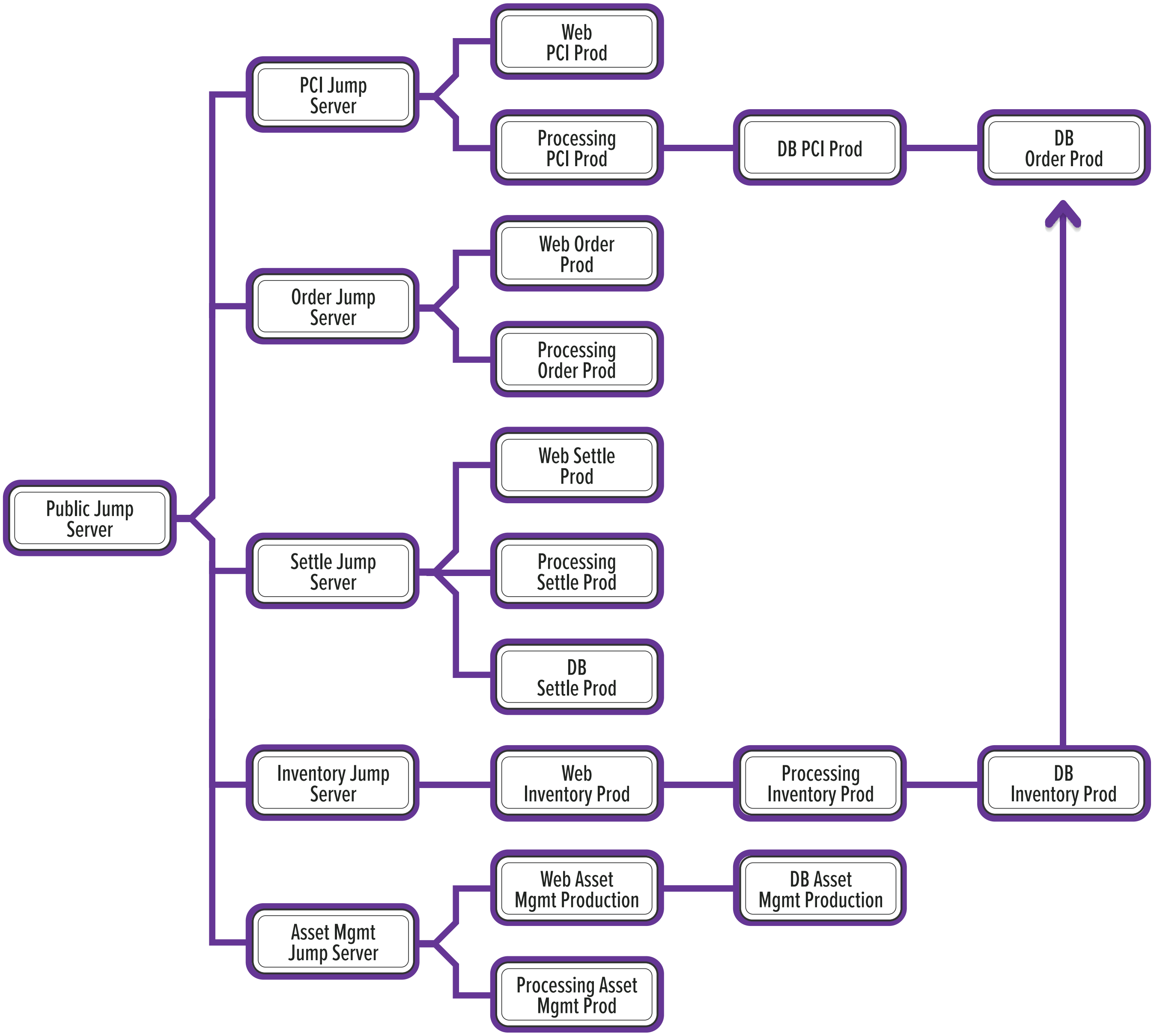


FIGURE 5 – SSH ACCESS SPECIAL CASES

This figure represents the SSH special access cases for the following five applications:

- PCI
- Order
- Settle
- Inventory
- Asset Management

The “DB Order Prod” server contained the `crown_jewels` database in which the trophy (a table containing simulated PII) was stored.

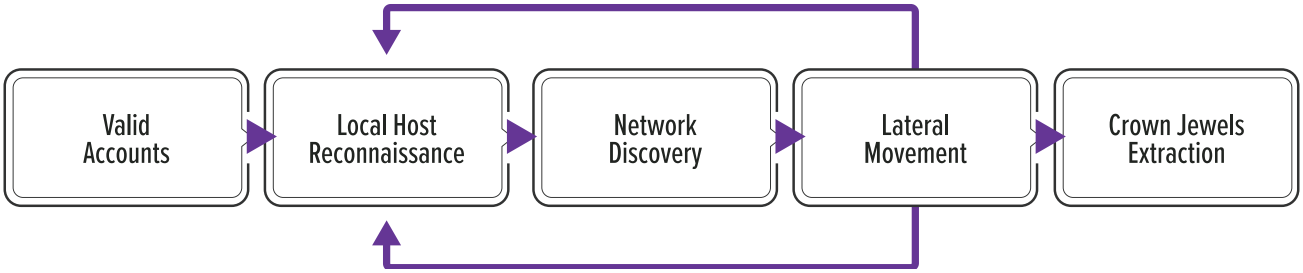
METHODOLOGY

To conduct the attack simulations, the assessment team extracted the relevant TTPs from the MITRE ATT&CK® and Pre-ATT&CK frameworks, based on the test environment expectations and the principal mitigation that is relevant in Illumio's context: network segmentation and lateral movement.

The team divided those TTPs in the following sections:

LOCAL HOST RECONNAISSANCE	NETWORK DISCOVERY	LATERAL MOVEMENT
T1057 - Process Discovery	**T1526** - Cloud Discovery	**T1184** - SSH Hijacking
T1016 - System Network Config Discovery	**T1046** - Network Scanning	**T1527** - Application Access Token
T1007 - System Service Discovery	**T1040** - Network Sniffing	**T1075** - Pass the Hash
T1083 - File and Directory Discovery	**T1018** - Remote System Discovery	**T1097** - Pass the Ticket
T1139 - Bash History	**TA0014** - Target Selection	**T1078** - Valid Accounts
T1081 - Credentials in Files		**T1199** - Trusted Relationship
T1145 - Private Keys		**T1489** - Stopping Services

Based on this table, the following approach was retained to perform the testing activities:



This approach was used for all the test cases, and for each host the team successfully pivoted to, they recorded the number of neighbors (accessible hosts from the current one) as well as available services through network scans.

TEST RESULTS

The assessment team performed a total of six different attack simulations. The control test and the first three use cases were run on 100 workload environments. To measure the effects of a change in environment, the team also performed the Use Case 2 (Application Ringfencing) scenario in both 500 and 1,000 workload environments to better understand how network size impacted the overall complexity.

This section describes the observable results of these tests and includes the time, host, and service metrics observed by Bishop Fox and the traffic events (allowed/denied connections) identified by Illumio.

CONTROL TEST

The control environment was composed of 99 workloads. As described in the previous section, this environment had a flat network topology, allowing every host to communicate with each other.

The team followed the methodology described in the previous section to perform reconnaissance, credential gathering, target analysis, and pivoting activities.

In this round of testing, the public jump host was `10.0.0.98.` From there, the team listed the running processes and quickly identified that the Illumio VEN agent was running (although in "Build Mode"):

```
[centos@ip-10-0-0-98 ~]$ ps auxf
…omitted for brevity…
root       1498  0.0  0.7 122932  7160 ?        S    Mar22   0:02
/opt/illumio_ven/bin/venAgentMgr -D
root       1587  0.1  0.7 637700  7940 ?        Sl   Mar22   0:52
/opt/illumio_ven/bin/venPlatformHandler -D
root       1652  0.0  0.4 184976  4360 ?        Sl   Mar22   0:02
/opt/illumio_ven/bin/venVtapServer -D
```

FIGURE 6 – RUNNING PROCESSES ON THE PUBLIC JUMP HOST

The team continued their local host reconnaissance and discovered the following:

- One established connection to the PCE server, hosted on `100.20.217.186:8444`
- Two SSH keys were accessible in `/home/centos/.ssh`
- The shell history contained references to the `10.0.1.0/24` subnet

The team then performed network scans targeting the `10.0.1.0/24` subnet and identified 99 hosts, exposing 293 services. From there, the team reused the SSH keys from the public jump host to connect to all the hosts and start the local reconnaissance loop again.

The team discovered the interesting `README.txt` file on `10.0.1.42`, as illustrated below:

```
[centos@ip-10-0-1-42 ~]$ cat README.txt
#Postgres commands
sudo -u postgres psql   #local connection
\l                      #list all databases
\c crown_jewels         #crown_jewels db
select * from users;    #get ssn
```

FIGURE 7 – FIRST CLUE RETRIEVED ON `10.0.1.42`

With this information, the next step was to run these commands on all Postgres servers to identify the one hosting the crown jewels and extract them:

```
[root@ip-10-0-0-98 listening]# for h in $(cat /tmp/postgres_list); do echo
"Trying $h" && ssh -o StrictHostKeyChecking=no -l centos -i
/home/centos/.ssh/id_rsa $h -p 1357 "sudo -u postgres psql -d crown_jewels
-c 'select * from users;'" 2>/dev/null; done
Trying 10.0.1.108
Trying 10.0.1.111
Trying 10.0.1.120
Trying 10.0.1.144
Trying 10.0.1.14
Trying 10.0.1.164
Trying 10.0.1.168
Trying 10.0.1.185
Trying 10.0.1.192
Trying 10.0.1.197
Trying 10.0.1.214
 id | first_name  | last_name |    ssn
----+-------------+-----------+-----------
  1 | test        | test      | 157783680
(1 row)

Trying 10.0.1.251
Trying 10.0.1.27
…omitted for brevity…
```

FIGURE 8 – CROWN JEWELS FOUND ON `10.0.1.42`

The following table summarizes the quantifiable results of this first round of testing:

MEASUREMENT TYPE	VALUE
Number of identified hosts	99
Number of identified open ports	293
Time to obtain the crown jewels	4 hours
Outgoing connections	12,500,294
Blocked connections	0
Allowed connections	12,500,294

The time to obtain the crown jewels appears high because it was the first time the assessment team was confronted with this environment. As explained in the environment overview section, the team had no prior information about what they should look for or extract. As expected, the number of blocked connections is zero because, by definition, there is no micro-segmentation in place for this flat network scenario.

As all rounds of testing were performed on the same environment, the team re-ran this test on a fresh setup in order to avoid skewing the results based on the learning curve encountered in the first control test. This new run allowed for more pertinent results to be compared to the next rounds of testing. The following table illustrates the results from this second control run:

MEASUREMENT TYPE	VALUE
Number of identified hosts	99
Number of identified open ports	293
Time to obtain the crown jewels	0.5 hours
Outgoing connections	13,361,949
Blocked connections	0
Allowed connections	13,361,949

As expected, it took the team only 30 minutes to reach the objective in this second round of testing (vs. four hours in the original run of this test). This second round represents the activities of an attacker optimizing their attacks against the test environment, without making any kind of mistake that could delay their actions.

This second round of testing represents the control test case. All subsequent test case's time measurements are compared with this second round of testing, because only the filtering policies changed; otherwise, the environment was the same in nature.

USE CASE 1 – ENVIRONMENTAL SEPARATION

This test scenario took place in an environment with the same properties as the control environment, where the only differences were that the Illumio VEN agent was enabled and micro-segmentation was in place. At a simple level, environmental separation boils down to Production hosts only being able to talk to Production, Development to Development, etc.

The entry point system for this round of testing was `10.0.0.90`. The team once again started by gathering local information from the public jump host, which led to similar observations:

- No established egress connections except from the one to the PCE server
- One SSH key located in `/home/centos/.ssh/`
- The shell history was empty

The team then moved on to the network discovery phase, and quickly noticed a first difference: Both ingress and egress ICMP traffic from and to the public jump host were filtered by the `iptables` rules, enforced by the network policy:

```
[centos@ip-10-0-0-90 scans]$ time sudo nmap -sn 10.0.1.0/24 -oA
results/live_hosts_10.0.1.0_24

Starting Nmap 6.40 ( http://nmap.org ) at 2020-03-26 05:13 PDT
sendto in send_ip_packet_sd: sendto(6, packet, 28, 0, 10.0.1.1, 16) =>
Operation not permitted
…omitted for brevity…
```

FIGURE 9 – OUTGOING ICMP TRAFFIC BLOCKED BY THE LOCAL HOST

The team resorted to layer four scanning to identify the reachable workloads from the public jump host. From these scans, nine new hosts were discovered in the 10.0.1.0/24 subnet:

- 10.0.1.10
- 10.0.1.100
- 10.0.1.124
- 10.0.1.151
- 10.0.1.156
- 10.0.1.174
- 10.0.1.177
- 10.0.1.195
- 10.0.1.250

The next step was to connect to each of these hosts to then start the loop again, as described in the methodology. The team ran local discovery on each host, then started scanning the address space from each of them.

After analyzing the results from this iteration of the loop on the second jump hosts, the team identified the interesting README.txt file hosted on 10.0.1.118:

```
[centos@ip-10-0-1-195 results]$ time for h in $(cat 22_up.txt); do echo “Trying
$h” && ssh -l centos -o StrictHostKeyChecking=no  $h cat README.txt; done
Trying 10.0.1.118
#Postgres commands
sudo -u postgres psql #local connection
\l #list all databases
\c crown_jewels #crown_jewels db
select * from users; #get ssn

real    0m0.401s
user    0m0.014s
sys 0m0.004s
```

FIGURE 10 – FIRST CLUE RETRIEVED ON 10.0.1.118

From there, the team went back to the already collected results to enumerate all available Postgres servers. Finally, they ran the following command from each host in the second jump list to retrieve the crown jewels:

```
[centos@ip-10-0-1-195 tmp]$ time for h in $(cat /tmp/postgres_hosts); do echo
“Trying $h” && ssh -p 1357 -o StrictHostKeyChecking=no $h “sudo -u postgres
psql -d crown_jewels -c ‘select * from users;’” 2>/dev/null; done
Trying 10.0.1.134
Trying 10.0.1.153
Trying 10.0.1.216
Trying 10.0.1.223
Trying 10.0.1.239
Trying 10.0.1.252
id   | first_name   | last_name |   ssn
----+------------+-----------+-----------
1    | test        | test      | 157783680
(1 row)

Trying 10.0.1.51
Trying 10.0.1.69
Trying 10.0.1.94

real    0m3.863s
user    0m0.128s
sys     0m0.031s
```

FIGURE 11 – RETRIEVING THE CROWN JEWELS FROM `10.0.1.252`

As highlighted in the above figure, the team accessed the host running the Postgres server containing the crown jewels via `10.0.1.195`. The final attack path was the following: `10.0.0.90 > 10.0.1.195 > 10.0.1.252.`

The following table illustrates the measurements related to this test round:

MEASUREMENT TYPE	VALUE
Number of identified hosts	91
Number of identified open ports	219
Time to obtain the crown jewels	1.5 hours
Outgoing connections	7,705,052
Blocked connections	16,902
Allowed connections	7,688,150

The main observation from this round of testing is that the time to get to the crown jewels increased by a factor of three (90 mins vs. 30 mins). This is explained by the fact that lateral movement was significantly reduced vs. the freedom afforded in a flat network. The number of identified hosts slightly decreased, because the team stopped the assessment once the objective was achieved. This fact also explains why the number of outgoing connections observed by the Illumio team decreased compared to the control environment testing.

USE CASE 2 – APPLICATION RINGFENCING

This test scenario took place in an environment with the same properties as the control environment, where the only differences were that the Illumio VEN agent was enabled and additional micro-segmentation was in place. Simply put, application ringfencing is where hosts and workloads within the same application and the same environment can communicate, but nothing else is permitted.

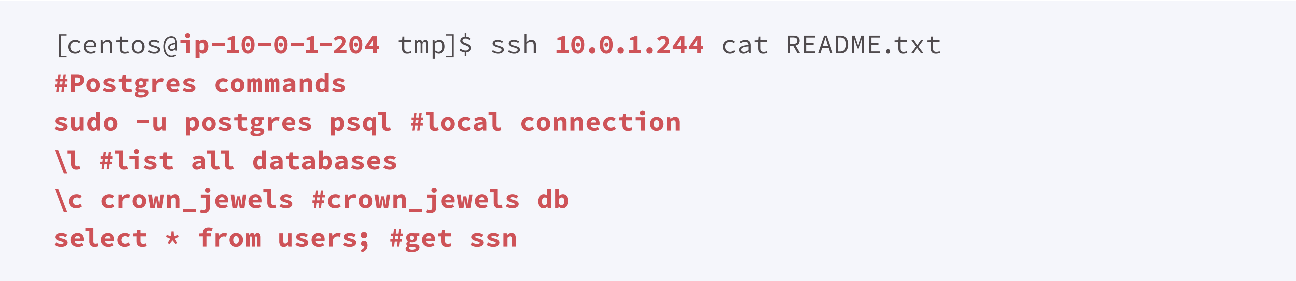
The entry point system for this round of testing was `10.0.0.186`. The local host reconnaissance phase from the public jump host gave identical results compared to the Use Case 1.

After running network scans from the public jump host, the team once again identified nine new jump hosts:

- `10.0.1.128`
- `10.0.1.134`
- `10.0.1.139`
- `10.0.1.17`
- `10.0.1.184`
- `10.0.1.204`
- `10.0.1.48`
- `10.0.1.77`
- `10.0.1.8`

Then, the team restarted the attack loop sequentially from each of these hosts. By reviewing the network scans results, the team noticed a difference in services exposed from the second jump hosts, compared to what they could access from the public jump host.

The team finally found the interesting `README.txt` file on `10.0.1.244`, accessible through `10.0.1.204`:



```
[centos@ip-10-0-1-204 tmp]$ ssh 10.0.1.244 cat README.txt
#Postgres commands
sudo -u postgres psql #local connection
\l #list all databases
\c crown_jewels #crown_jewels db
select * from users; #get ssn
```

FIGURE 12 – FIRST CLUE RETRIEVED ON `10.0.1.244`

From there, the team built a list of all the discovered hosts running a Postgres instance and attempted to run the highlighted command via SSH to retrieve the crown jewels. Eventually, the team found the right server on 10.0.1.167:

```
[centos@ip-10-0-1-139]$ time for h in $(cat /tmp/postgres_hosts); do echo
“Trying $h” && ssh -p 1357 -o StrictHostKeyChecking=no $h “sudo -u postgres
psql -d crown_jewels -c ‘select * from users;’” 2>/dev/null; done
Trying 10.0.1.139
Trying 10.0.1.152
Trying 10.0.1.101
Trying 10.0.1.120
Trying 10.0.1.121
Trying 10.0.1.133
Trying 10.0.1.167
 id | first_name  | last_name |    ssn
----+-------------+-----------+-----------
  1 | test        | test      | 157783680
(1 row)

Trying 10.0.1.187
Trying 10.0.1.196
Trying 10.0.1.205
```

FIGURE 13 – RETRIEVING THE CROWN JEWELS FROM 10.0.1.167

The final attack path to obtain the crown jewels was the following: 10.0.0.186 > 10.0.1.139 > 10.0.1.167.

The following table illustrates the measurements related to this test round:

MEASUREMENT TYPE	VALUE
Number of identified hosts	99
Number of identified open ports	173
Time to obtain the crown jewels	2.25 hours
Outgoing connections	11,905,973
Blocked connections	64,033
Allowed connections	11,841,940

Once again, fewer hosts were discovered during this test round. This is mainly due to the tighter micro-segmentation policy enforced on the test environment, which also reduced the number of identified services. The team also tested significantly more trial and error cases, which is illustrated by the higher number of outgoing connections. More scans had to be run to identify the server hosting the crown jewels.

Finally, the time for the team to obtain the crown jewels increased by a factor of 1.5 compared to the first use case, and by 3 compared to the control test. This demonstrates once again the efficacy of a micro-segmentation policy.

USE CASE 3 – TIER BASED SEGMENTATION

This test scenario took place in an environment with the same properties as the control environment, where the only differences were that the Illumio VEN agent was enabled and micro-segmentation was in place. Tier-based segmentation is one of the most granular forms of micro-segmentation where only workloads in the same environment, same app and same app tier are able to communicate with each other. For an adversary this represents a significant reduction in the number of open pathways to exploit in order to move laterally.

The entry point system for this round of testing was `10.0.0.146`. The local host reconnaissance phase from the public jump host gave identical results compared to the Use Cases 1 and 2.

After running network scans from the public jump host, the team again identified nine new jump hosts:

- `10.0.1.18`
- `10.0.1.22`
- `10.0.1.38`
- `10.0.1.57`
- `10.0.1.67`
- `10.0.1.73`
- `10.0.1.138`
- `10.0.1.208`
- `10.0.1.250`

The team then restarted the attack loop sequentially from these discovered hosts to collect local host information, and then perform network scans. This second round of reconnaissance and discovery revealed that the README.txt file was present on 10.0.1.146, reachable from 10.0.1.18:

```
[centos@ip-10-0-1-18 ~]$ ssh 10.0.1.146 cat README.txt
#Postgres commands
sudo -u postgres psql #local connection
\l #list all databases
\c crown_jewels #crown_jewels db
select * from users; #get ssn
```

FIGURE 14 – FIRST CLUE RETRIEVED ON 10.0.1.146

From there, the team analyzed the local reconnaissance results extracted from all the hosts from the second jump list above to identify the running Postgres server instances. Then they sequentially attempted to connect to all of them through SSH to retrieve the crown jewels.

This method was not successful. As a result, the team restarted the attack loop from the hosts accessible from the second jump location, but with more targeted scans (focusing on SSH exposed ports). By analyzing the results, the team identified one server, 10.0.1.137, that had access to a completely new set of targets, including the server that stored the crown jewels: 10.0.1.248.

```
[centos@ip-10-0-1-137 ~]$ for x in $(cat up.txt); do echo “=== $x ===”; ssh -o
StrictHostKeyChecking=no -p 1357 $x “sudo -u postgres psql -d crown_jewels
-c ‘select * from users;’ 2>/dev/null”; done
=== 10.0.1.7 ===
=== 10.0.1.244 ===
=== 10.0.1.248 ===
 id | first_name  | last_name |    ssn
----+-------------+-----------+-----------
  1 | test        | test      | 157783680
(1 row)
```

FIGURE 15 – CROWN JEWELS RETRIEVAL ON 10.0.1.248

The following table illustrates the measurements related to this test round:

MEASUREMENT TYPE	VALUE
Number of identified hosts	99
Number of identified open ports	130
Time to obtain the crown jewels	4.75 hours
Outgoing connections	187,564
Blocked connections	181,772
Allowed connections	5,792

Due to the tight tier-based micro-segmentation policy, the team had to enumerate all the hosts available in the test environment to identify the one storing the trophy data. The decreased number of discovered ports is explained by both the tighter network filtering policy and the change in the approach operated near the end of the testing round. As the team found the first clue early in the test round, the number of scanned ports decreased due to more targeted scans. This is the main reason behind the low number of outgoing connections observed by the Illumio team.

Finally, the time for the team to obtain the crown jewels increased by a factor of 2.11 compared to the second use case, and by 9.5 compared to the control test.

USE CASE 2 – 500 WORKLOADS

This scenario is a replica of Use Case 2, but with the number of workloads increased to 500 (instead of 100) in order to understand how the total number of workloads impacted the time needed to enumerate the network and capture the trophy.

The approach and initial observations made by the assessment team were identical to those for Use Case 2. The only noticeable difference was the time it took to perform the sequential network scanning activities.

The following table illustrates the measurements related to this test round:

MEASUREMENT TYPE	VALUE
Number of identified hosts	499
Number of identified open ports	2,489
Time to obtain the crown jewels	4.25 hours
Outgoing connections	13,796,977
Blocked connections	123,068
Allowed connections	13,673,909

Multiplying the environment size by five had a direct impact on the time required for the assessment team to retrieve the crown jewels. Time to reach the objective in this scenario increased by a factor of 1.88 compared to Use Case 2, and by 8.5 compared to the control environment.

The number of outgoing connections was slightly lower because at this stage of the testing, the team leveraged information from the local host firewall rules to identify the next reachable hosts instead of blindly scanning the network to discover the hosts. This also allowed the team to identify all the hosts in the environment.

USE CASE 2 – 1,000 WORKLOADS

This scenario is a replica of Use Case 2, but with the number of workloads increased to 1,000 (instead of 100).

The approach and initial observations made by the assessment team were identical to those for Use Case 2. The only noticeable difference was the time it took to perform the network scanning activities.

The following table describes the measurements related to this test round:

MEASUREMENT TYPE	VALUE
Number of identified hosts	999
Number of identified open ports	5,168
Time to obtain the crown jewels	11 hours
Outgoing connections	23,360,897
Blocked connections	185,233
Allowed connections	23,175,664

As the size of the environment was ten times bigger than the environment for the control test, the team decided to run the network discovery phase in parallel whenever possible. Until then, all the network scanning activities were run sequentially, which would have taken significantly longer in this scenario. Of course, in a real-world scenario, the adversary is trying to optimize penetration of the network as quickly as possible vs. staying under the radar, not being detected, and not triggering any tripwires before the objective is reached.

Using this approach, it took the assessment team a little less than two hours (one hour and 51 minutes) to obtain the crown jewels. To be fair and to get comparable results, in the experiment the team obtained the retained time in the above table by cumulating the time of each network scan as if they had been running sequentially.

The team used the same approach for remote host discovery as for the 500 workloads use case. By doing so, the team was able to rapidly identify new hosts without running network scans, which resulted in a limited number of outgoing network connections.

CONCLUSIONS

In conclusion, the assessment team identified that properly applied micro-segmentation policies increased the difficulty of lateral movement and pivoting through the tested network, resulting in an overall increased time to compromise and production of detectable events in order for the attacker to reach the targeted sensitive information. **Specifically:**

- Even the most basic segmentation policy to keep application environments separate requires the attacker to spend at least three times more effort
- Increasing coverage size of micro-segmentation capabilities, while maintaining the same policy state, results in measurable gains in delaying the attacker
- Use of micro-segmentation forces the attacker to change techniques in order to traverse the network more efficiently

Combined, these findings highlight the importance of adopting micro-segmentation as part of an organization's enterprise security posture, given the control's measurable effectiveness in putting the brakes on lateral movement.