PriveonLabs

Cisco Security MARS Series:

Network Tripwire - 101

Larry Boggis
Senior Security Consultant

Overview

The Cisco Security Monitoring, Analysis, and Response System (CS-MARS) is a powerful Security Threat Mitigation (STM) appliance. It delivers an in-depth view into your networks' security posture and health as seen through the "eyes and ears" of the reporting devices in your network.

In this document we will show how to leverage the features of the CS-MARS and information gathered from the network reporting devices to create a tripwire mechanism that can help identify and alert on malware and internal malicious network activity.

As a note, this detection method is meant to provide an additional level of security and is not intended to replace existing tools. A good defense in-depth strategy should include many different detection and alerting methods and mechanisms.

Network Tripwire Overview

A tripwire can be defined as a passive triggering mechanism that activates some sort of alert indication. We can relate this mechanism to our network environment to alert us of malicious activity by leveraging the network infrastructure reporting devices (Routers and Switches) and the event correlation, reporting and alerting functionality of the CS-MARS. The first step in this process is to define "malicious activity".

Malware, in one form or another, traditionally follows a 5-phase attack life cycle: Probe, Penetrate, Persist, Propagate, and Paralyze. The Probe and Propagate phases are of interest to us since we can directly relate network activity to the behavior associated with malware probing or scanning for vulnerable/exploitable systems and the propagation traffic once the malware has compromised end-systems. Traditionally this activity has been the result of worm propagation. Today this activity is more commonly seen as the result of "drive-by" web infections and targeted exploits.

Examples of traditional worm network activity:

Malware:	SQL Slammer
Propagation:	376-byte UDP packets destined for UDP port 1434 to random IP addresses

Malware:	Blaster Worm
Propagation:	Targets systems with vulnerable DCOM RPC Services running (TCP/135). DST IP based on simple algorithm (60% probability of random addresses)
Malware:	Sasser Worm
Propagation:	Targets Unpatched systems vulnerable to LSASS exploit - MS04-011. Spreads by scanning randomly selected IP addresses (TCP/445).

Additional examples of network activity as a result of endpoint infection can be found in other Priveon Labs research and whitepapers.

Malicious activity can also include the behavior of internal reconnaissance -- scanning the internal network looking for potential targets or the process of mapping the network. Some network environments leverage the use of IDS/IPS technology and devices to alert on reconnaissance behavior, but these devices are usually processing large volumes of data and a slow network scan can usually evade detection. Additionally, IDS/IPS devices are usually positioned to examine perimeter traffic or traffic between security "zones" and do not have the visibility into the entire network. By leveraging the network devices themselves as the reporting mechanism, we can greatly increase our visibility into all parts of the network to help identify malicious activity.

The question still remains on how to identify "good" traffic from potentially "bad" traffic. One of the simplest ways to do this is to define "good" traffic as network traffic destined to known applications on known servers and on internally routable subnets. That leaves us with "bad" traffic as being defined as traffic destined for unknown servers or services and traffic destined to unknown [or dark] address space.

Dark address space refers to any or all unreachable or unused networks or network hosts within a specific environment. For this example dark address space refers to all unused internal addresses and networks.

Traffic destined to dark address space is usually sourced from:

1. Mis-configured systems or applications
2. Old/Legacy applications and application settings trying to communicate to nonexistent networks or systems no longer on the network.
3. Infected systems - malware propagation
4. Network scans from individuals or automated scripts trying to map the internal network
5. Legitimate traffic from "known" internal network management or audit server(s)

A network tripwire should also have the capability of filtering out false-positive traffic patterns from "known" or "trusted" internal scanners/applications. It should be configured and tuned to only alert on untrusted activity -- since some action and follow-up investigation should be taken on each of these events.

Network Setup – Configuring the Reporting Device(s)

There are many ways to identify and alert on traffic destined for "dark" or unused internal address space:

1. Map specific unused subnets to infrastructure vlans/interfaces and advertise these subnets with the internal routing protocol – This approach assumes you will identify several subnets that are not being used and assign them to a core loopback or vlan interface for reporting. This method is valid, but it has limited visibility and provides the smallest triggering footprint since it relies on just a few pre-defined subnets/networks.

2. Create a summary route [or routes] to identify internal address space and advertise with the internal routing protocol – This approach relies on summary route(s) that will be used in the case where more specific internal routes do not exist. This route can be defined to route traffic to a specific core interface and ACL for reporting.

3. Use NetFlow data from core network devices to gain visibility into internal traffic flows – This assumes that the MARS device has been sized accordingly since using NetFlow data for event and incident data requires writing this information to the MARS database.

4. Leverage the default route and use event data from a device with visibility into egress internet traffic such as the egress firewall – This approach assumes that all internal traffic without a specific route will be sent to the egress firewall via the internal default route. (of course with best practices this traffic should be dropped by an internal summary route or through egress filtering mechanisms before hitting the default gateway)

All four approaches are valid and have different pros/cons. This document assumes the reader has some knowledge of routing and routing protocols and can determine the best approach for their environment. For this example we will use option #1 since it's the fastest and easiest to demonstrate a tripwire function in a lab environment.

Network Reporting Device – Setup

The first step in defining our tripwire is to identify "dark" or unused address space. For this example our network has an internal address space of 192.168.0.0/16 and we have identified an unused portion of that address space as 192.168.10.0/24. Obviously the more "dark" space you can use the more visibility you will have and the better chance of catching malicious activity. In this example we just want to demonstrate functionality so we have defined the following configuration on a Cisco switch running IOS version 12.2:

Send syslog (level 6 – informational) messages to the MARS:
```
logging on
logging trap 6
logging <ip address of MARS>
```

Define Tripwire Interface:
```
interface Vlan10
 description TripWire Example
 ip address 192.168.10.1 255.255.255.0
 ip access-group 110 out
```

Define Access-List to Report Data:
```
access-list 110 permit ip any any log-input
```

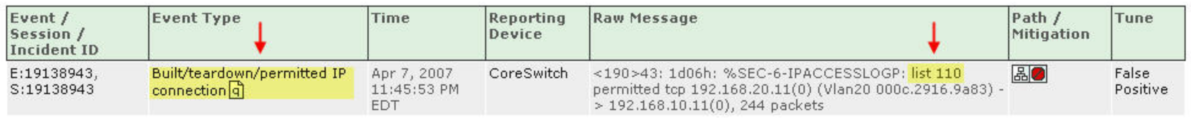
The above configuration will send an informational logging message about the packet (including the input interface and source MAC address or VC) that matches our tripwire ACL to be sent to the MARS.

CS-MARS Setup

The switch used for this example has already been defined as a reporting device on the MARS, so that base configuration is not part of this document scope.

The first step in defining our tripwire rule on the MARS is to identify the event and event type associated with the tripwire ACL syslog message. Here we have defined a query to display the raw events being sent from the CoreSwitch device configured as part of the network setup. The query results show that the raw syslog message generated by ACL 110 is mapped to the MARS event type "Build/teardown/permitted IP connection":

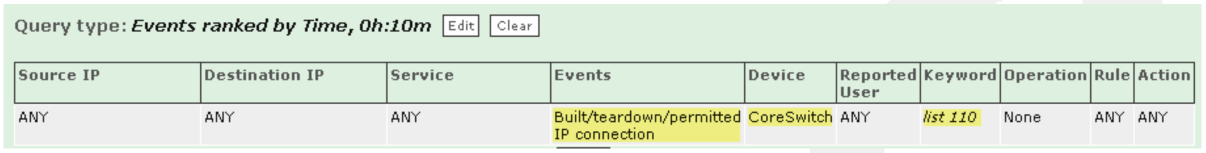
Create Query and Examine event data



Event / Session / Incident ID	Event Type	Time	Reporting Device	Raw Message	Path / Mitigation	Tune
E:19138943, S:19138943	Built/teardown/permitted IP connection	Apr 7, 2007 11:45:53 PM EDT	CoreSwitch	<190>43: 1d06h: %SEC-6-IPACCESSLOGP: list 110 permitted tcp 192.168.20.11(0) (Vlan20 000c.2916.9a83) -> 192.168.10.11(0), 244 packets		False Positive

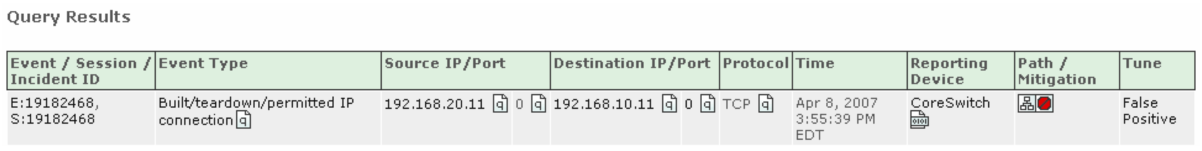
Next, we further define our query to make sure it is as specific as possible:

Event:	Build/teardown/permitted IP connection
Device:	CoreSwitch
Keyword:	list 110



Query type: *Events ranked by Time, 0h:10m*

Source IP	Destination IP	Service	Events	Device	Reported User	Keyword	Operation	Rule	Action
ANY	ANY	ANY	Built/teardown/permitted IP connection	CoreSwitch	ANY	*list 110*	None	ANY	ANY



Query Results

Event / Session / Incident ID	Event Type	Source IP/Port	Destination IP/Port	Protocol	Time	Reporting Device	Path / Mitigation	Tune
E:19182468, S:19182468	Built/teardown/permitted IP connection	192.168.20.11 0	192.168.10.11 0	TCP	Apr 8, 2007 3:55:39 PM EDT	CoreSwitch		False Positive

Once we are satisfied that the query is returning the correct results, we select "Save As Rule" to begin the rule definition.

Make sure that best practices are used for the Rule Name and Description:

Rule Name:	`PRIV-Tripwire`
Rule Description:	`Tripwire to alarm on internal malicious activity.`

For this example we are not concerned about the Source IP, Destination IP or Service. We want to be notified if ANY traffic hits the tripwire:

Source IP:	`ANY`
Destination IP:	`ANY`
Service Name:	`ANY`

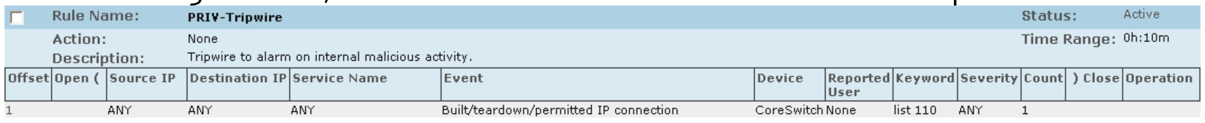
We also want to be notified if even a single packet hits the tripwire:

Severity:	`ANY`
Counts:	`1`

At this point we first want to make sure the rule is triggering correctly before we apply a Rule Action, so we will leave it blank:

Rule Action:	`<blank>`
Time Range:	`10 Min`

After defining the rule, we hit "Submit" and view the new PRIV-Tripwire rule:



Rule Name:	PRIV-Tripwire	Status:	Active
Action:	None	Time Range:	0h:10m
Description:	Tripwire to alarm on internal malicious activity.		

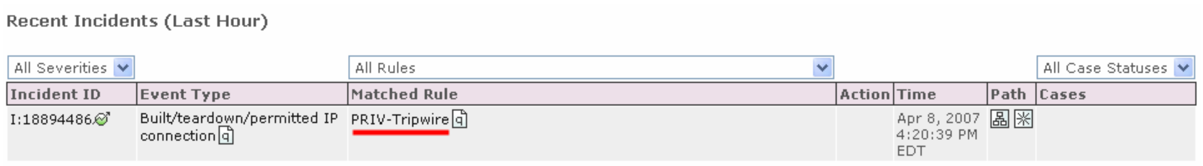
Offset	Open (	Source IP	Destination IP	Service Name	Event	Device	Reported User	Keyword	Severity	Count	) Close	Operation
1		ANY	ANY	ANY	Built/teardown/permitted IP connection	CoreSwitch	None	list 110	ANY	1		

Make sure you also "Activate" to compile and apply the newly created rule to the MARS LOGIC service. The rule will not fire until this happens.

The Results

In the previous steps we defined a tripwire rule to fire on any traffic hitting an ACL on our Core Switch. This ACL matched traffic destined to a pre-defined and unused [dark] subnet we identified for our network environment.

The network tripwire has been created and now any network traffic that "triggers" the tripwire will create an incident on the MARS. Here we see that our PRIV-Tripwire rule fired and created Incident ID: 18894486

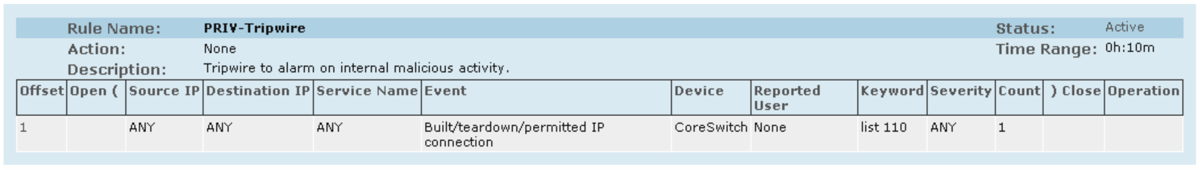


Recent Incidents (Last Hour)

All Severities | All Rules | All Case Statuses

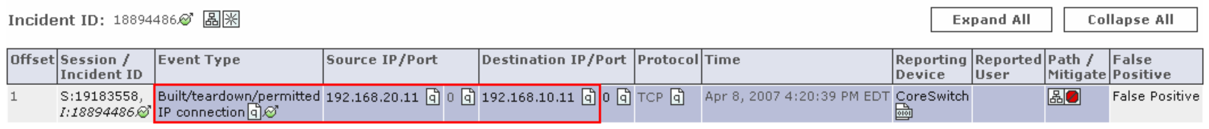
Incident ID	Event Type	Matched Rule	Action	Time	Path	Cases
I:18894486	Built/teardown/permitted IP connection	PRIV-Tripwire		Apr 8, 2007 4:20:39 PM EDT		

If we click on the incident ID, we are taken to the associated Session and Event data that triggered the rule. Here we see that the source IP address 192.168.20.11 triggered the tripwire by sending a TCP connection to 192.168.10.11:



Rule Name: PRIV-Tripwire
Action: None
Description: Tripwire to alarm on internal malicious activity.
Status: Active
Time Range: 0h:10m

Offset	Open (	Source IP	Destination IP	Service Name	Event	Device	Reported User	Keyword	Severity	Count	) Close	Operation
1		ANY	ANY	ANY	Built/teardown/permitted IP connection	CoreSwitch	None	list 110	ANY	1		

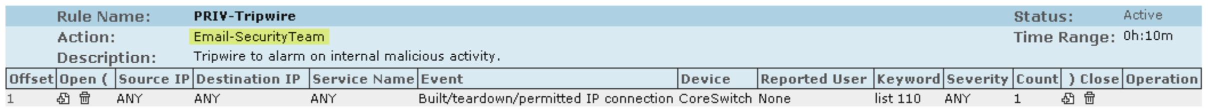


Incident ID: 18894486

Expand All | Collapse All

Offset	Session / Incident ID	Event Type	Source IP/Port		Destination IP/Port		Protocol	Time	Reporting Device	Reported User	Path / Mitigate	False Positive
1	S:19183558, *I:18894486*	Built/teardown/permitted IP connection	192.168.20.11	0	192.168.10.11	0	TCP	Apr 8, 2007 4:20:39 PM EDT	CoreSwitch			False Positive

Now that we know our tripwire rule is functioning correctly, we can add an appropriate Rule Action. Here we have added an action of "Email-SecurityTeam" to be notified by email for every tripwire incident:



Rule Name: PRIV-Tripwire
Action: Email-SecurityTeam
Description: Tripwire to alarm on internal malicious activity.
Status: Active
Time Range: 0h:10m

Offset	Open (	Source IP	Destination IP	Service Name	Event	Device	Reported User	Keyword	Severity	Count	) Close	Operation
1		ANY	ANY	ANY	Built/teardown/permitted IP connection	CoreSwitch	None	list 110	ANY	1		

Summary

The CS-MARS is a valuable tool, providing visibility into internal and external network traffic flows and behavior. The MARS' ability to use data from the network devices themselves and to have those devices act as the eyes and ears to help identify malicious network behavior is a powerful solution.

In this example we demonstrated a way to use the MARS and the network devices reporting to the MARS to help create a tripwire mechanism. By identifying and alerting on traffic destined to 'dark' address space we can better detect and alarm on malicious reconnaissance activity, malware propagation, and mis-configured internal systems/application traffic.

For additional whitepapers, research and training information on the Cisco MARS and other Cisco security products, please visit our web site at:
http://www.priveon.com