Continuous Empirical Validation of Network Security Controls

November 16, 2015

Jay Houghton
CTO at Firebind
jay@firebind.com
www.firebind.com

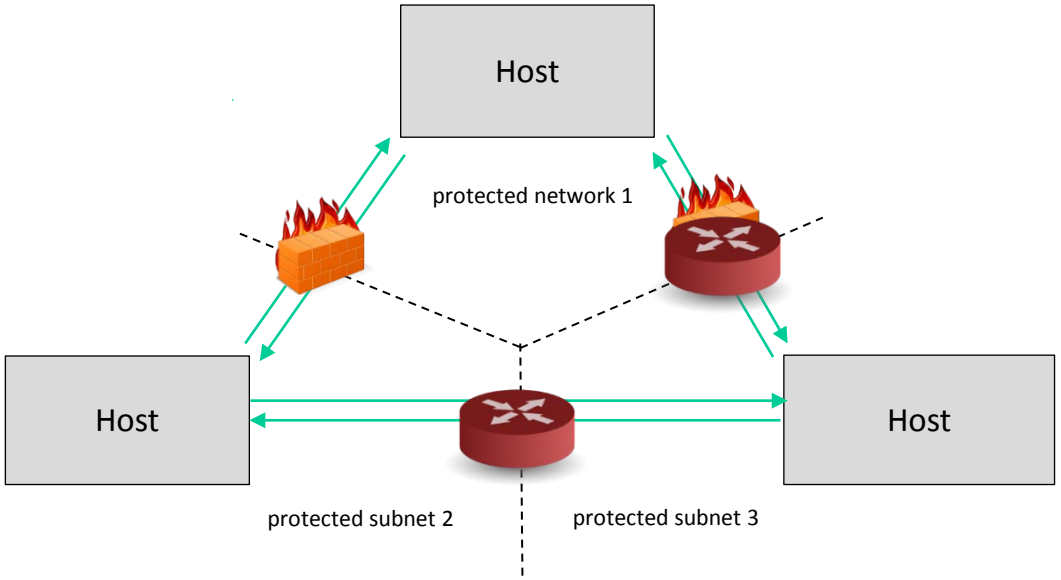
Keep Me Informed

That Mole Army Is Here...

Security Controls

Network-based Security Controls

- YES = switch/router, firewall, IDPS, FIPS199
- NO = endpoints, hosts, servers



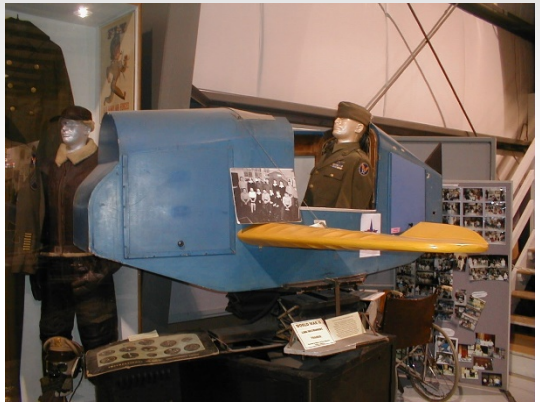
What?

Focus on data on the wire

- Continuous = constant observation
- Empirical = use real data on real network
- Validation = meets requirements

What does this mean?

Today & Beyond



Control and Config Management =
Flight Simulator

Network Pentest = Test Pilot

Continuous Empirical =
Commercial Air Traffic

747-8I_(N6067E).jpg: © Dave Subelack from YYC

Continuous

DHS CDM for FISMA

"Continuous Monitoring... Configuration & Vulnerability"

PCI DSS 3.1 section 11.2.3

"*Perform scan after any change*"

HIPAA NIST 800-66, RMF

"*assessment and evaluation of security controls on a continuous basis*"

Empirical

Use real data on real networks

Real Data:

- simulated normal traffic
- simulated attack traffic
- simulated PII traffic

Real Networks:

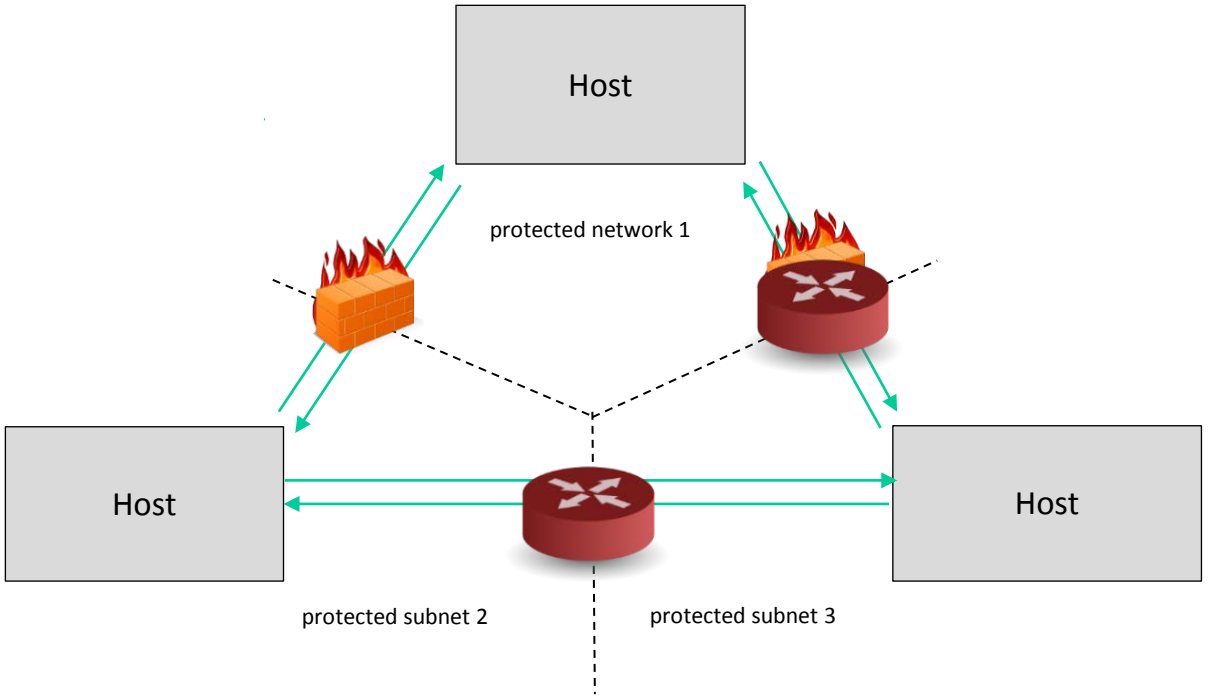
- Path to & from secure networks
- And all the devices along that path (controls)

Motivations

- Network doesn't know Δ between real and simulated traffic
  - Send over all protocols, all ports & all transports
- Send traffic to all paths between points
  - Send simulated threat traffic (malware signatures)
  - Requires knowledge of vulnerability
  - Hand crafted is state of the art
  - BUT there are sources of structured threat definitions

Approach

Produce and consume data between segments, via scripting, tools, other software



- Requires resources on both sides of controls = complexity
- Mitigated by recent adoption of VMs, cloud and inexpensive form factors (Raspberry Pi, Intel NUC, etc)

Normal Traffic

All transports, ports & protocols

Complete baseline profile of path access control

- e.g. complete profile of all firewall rules in play

Sender– use looped netcat

```
for i in 1 .. 65535; do
    cat file.txt | netcat -v <target IP> $i
done
```

Receiver – use iptables and netcat

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 1:65535 -j DNAT --to-destination 127.0.0.1:8010
netcat –kl 127.0.0.1 8010 > /dev/null
```

Normal Traffic

Considerations

- Silent discard is a timeout = long testing cycle
- A form of port scanning = IDPS alerts
- Firewall state table = be sure not to crash (UDP)
- Results show as netcat output, how to report?
- Source file for data = layer 7 protocols
- Limits of these tools
  - What about response (two-way open?, catch mismatch)
  - iptables bind conflicts

Threat Simulation

Goal: pass malware through network defense

- Simulated signature under controlled conditions !
- **Best Case**: Controls mitigate attack
- **Worst Case**: YOU KNOW BEFORE A REAL ATTACK

<u>Simulated malware requires expertise</u>

Candidate sources

- IDPS rule sets (typically regex)
- CVE® and researcher analysis (typically posted online)
- Machine format from MAEC™
  - **Malware Attribute Enumeration and Characterization**

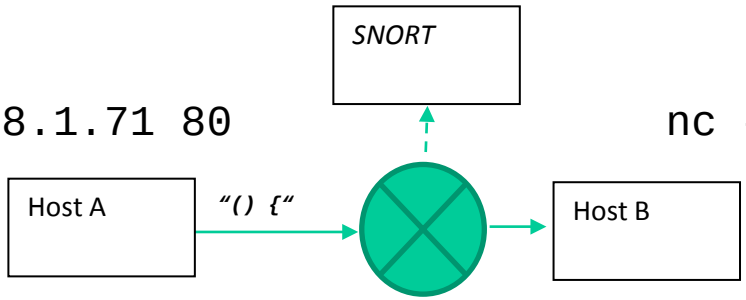
Shellshock MAEC

```
<maecBundle:MAEC_Bundle xmlns:maecBundle=http://maec.mitre.org/XMLSchema/maec-bundle-4 …>
  <maecBundle:Malware_Instance_Object_Attributes>
    <cybox:Description>CVE-2014-6271 Shell Shock #1</cybox:Description>
...
  <maecBundle:Strategic_Objective id="maec-shell-shock-obt-1">
    <maecBundle:Name xsi:type="maecVocabs:CommandandControlStrategicObjectivesVocab-
   1.0">innoculate controls against CVE-2014-6271</maecBundle:Name>
    </maecBundle:Strategic_Objective>
    <maecBundle:Tactical_Objective id="maec-shell-shock-obt-2">
      <maecBundle:Name xsi:type="maecVocabs:CommandandControlTacticalObjectivesVocab-
   1.0">send shell shock payload</maecBundle:Name>
...
  <maecBundle:Description>An injection of CVE-2014-6271 (shellshock 1 of 6) across
   security controls. The expression \(\) &#123; represents the first form of shellshock
   exploit " () {" </maecBundle:Description>
...
  <HTTPSessionObj:HTTP_Request_Line>
    <HTTPSessionObj:HTTP_Method datatype="string">POST</HTTPSessionObj:HTTP_Method>
    <HTTPSessionObj:Value>http://target/cgi-script.cgi</HTTPSessionObj:Value>
</HTTPSessionObj:HTTP_Request_Line>
  <HTTPSessionObj:HTTP_Message_Body>
    <HTTPSessionObj:Message_Body condition="FitsPattern" pattern_type="Regex">
        \(\) &#123;
    </HTTPSessionObj:Message_Body>
...
</maecBundle:MAEC_Bundle>
```

Simulated Malware

- Shellshock through Snort - CVE-2014-6271
- Payload `() {` = start of attack signature
  - Real world `() { :; }; cat /etc/passwd`

```
echo "() {" | nc 192.168.1.71 80
```




```
nc –l 192.168.1.71 80
```

Rule:
alert tcp any any -> any any (msg:"OS-OTHER Bash CGI environment variable injection attempt";
flow:stateless; content:"() {"; classtype:attempted-admin;)

Log:
[Priority: 1] {TCP} 192.168.1.70:50995 -> 192.168.1.71:80
11/09-12:31:22.150984 [**] [1:9000992:1] OS-OTHER Bash CGI environment variable injection attempt
[**] [Classification: Attempted Administrator Privilege Gain

Conclusions

- Continuous empirical testing of network security controls has historically been challenging
- Preponderance of endpoint focused security approaches ignore the behavior of network devices
- Continuous validation is more feasible than ever, given tools, virtual infrastructure and low barrier deployment vehicles
- Continuous monitoring *does* catch misconfigurations, new vulnerabilities and provides high confidence

Firebind

Network Visibility Platform that continuously and empirically monitors the network security, availability, and performance posture across in-house, cloud, and mobile infrastructure.

For more information or a demonstration please contact me!

Jay Houghton
CTO at Firebind
jay@firebind.com
www.firebind.com