Network Inspection of Duplicate Packets

GIAC (GCIA) Gold Certification

Author: Randy Devlin, rdevlin@mastersprogram.sans.edu
Advisor: Adam Kliarsky

Accepted: November 5, 2016

Abstract

Network Intrusion Analysis enables a security analyst to review network traffic for protocol conformity and anomalous behavior. The analyst's goal is to detect network intrusion activity in near-real time. The detection provides details as to who the attackers are, the attack type, and potential remediation responses. Is it possible that a network security stack could render the analyst "blind" to detecting intrusions? This paper will review architecture, traffic flow, and inspection processes. Architecture review validates proper sensor placement for inspection. Traffic flow analyzes sources and destinations, approved applications, and known traffic patterns. Inspection process evaluates protocols and packet specific details. The combination of these activities can reveal scenarios that potentially result in limitations of network security inspection and analysis.

1. Introduction

Network inspection is a vital component for intrusion analysts to review and detect threatening or anomalous network traffic. A strong foundation for network intrusion analysis requires properly placed inspection points and documentation of expected and approved traffic that should exist.

Inspection points provide visibility to critical "intersections" of the network. As Luciana Obregon discussed in the Infrastructure Security Architecture for Effective Security Monitoring Gold Paper, segmenting the network into logical zones, create decisive "intersections" ideal for inspection points (Obregon, 2015). The recommendation for segmentation is further emphasized by Nimmy Reichenberg, in tip four "*Create and segment the DMZ*" (Reichenberg, 2013).

Understanding network flows and at which point the inspection takes place is key for proper analysis. During analysis, inaccurate expectations can lead analysts to be subjective, resulting in improper investigations. This can subsequently lead to incorrect conclusions. The issue of expectation (or assumption) is explicitly pointed out in the second 'commandment' of the *10 Commandments of Intrusion Analysis*; "Unless you created the packet yourself, there are no absolutes" (Sanders, 2011).

The similarities between a digital incident analysis approach and a law enforcement physical scene approach share the same concept as to how to carry out the investigation. Regarding the physical approach, the principle for the Preliminary Documentation and Evaluation of the Scene states: "allows for the determination of the type of incident to be investigated and the level of investigation to be conducted" (DOJ, 2000, pg 19). This holds true for the digital incident analysis approach as well.

2. Intrusion Analysis Insight

Insight into architecture, flow, and applications/data are foundational for analysts to begin an analysis. Deficiencies here can quickly skew the analysis potentially creating what's known as 'intrusion blindness'[1] or 'investigation fatigue'[2].

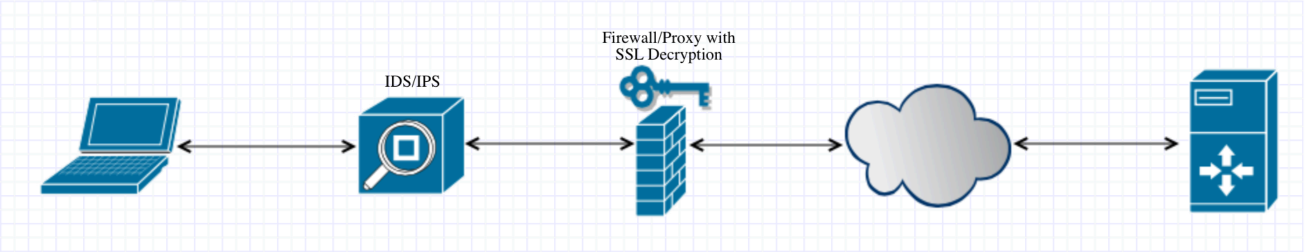
2.1 Architecture

Architecture is responsible for where and how inspection takes place. Segmentation points create the first ideal locations for inspection points. Basic segmentation involves three segmented networks, Internal, External, DMZ. This segmentation generates three natural inspection points:

1. Between Internal and External
2. Between Internal and DMZ
3. Between External and DMZ

Once the inspection points are defined, the method of inspection is next to be addressed. There are three common methods for how network inspection can occur; Inline, via a Switched Port Analyzer (SPAN) port, or using a Network Test Access Point (TAP).

Inline inspection occurs through a sensor with separate interfaces between the two systems communicating. The inspection platform receives the packets, performs inspection, if nothing malicious is detected, the inspection platform transmits the packets to the destination. One of the primary benefits of the inline method is the ability to take prevention actions to traffic deemed malicious. The drawback will be latency introduced by the performance for inspection.



[1] Intrusion blindness is caused by the lack of visibility to network traffic at specific points in the network

[2] Investigation fatigue is a security analyst's tiredness from reviewing data not directly relevant to an incident.

Figure 1. Inline

SPAN architecture leverages the switching platform's ability to duplicate the traffic and send a copy of the traffic to the inspection platform. Duplication does not impact the normal packet processing. If the switching platform begins to experience resource availability issues, SPAN traffic is one of the first resources to no longer process. The inspection platform will not receive any packets when the SPAN resources stop processing. This scenario creates gaps in visibility which can defeat the intention of the design.

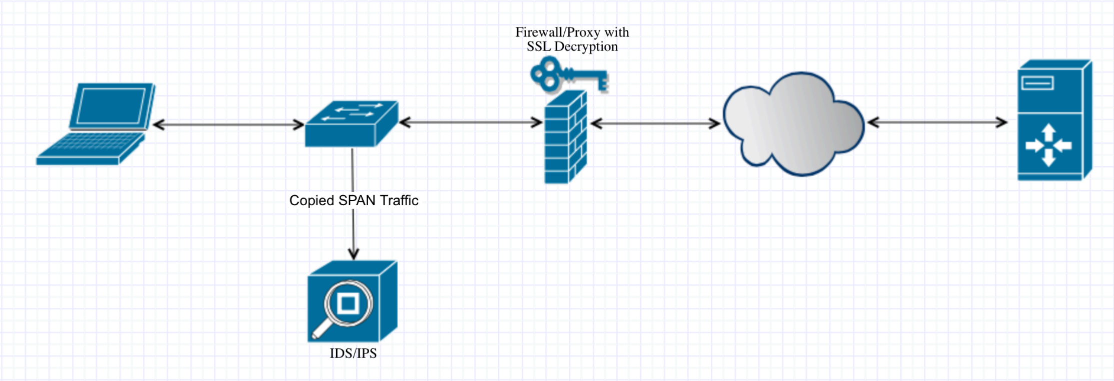


Figure 2. SPAN

TAP architecture appears similar to SPAN. TAP copies the traffic forward to the inspection platform. The main difference is the TAP is a dedicated appliance (hardware or software) introduced inline with the sole purpose of duplicating traffic and sending it forward. TAP architecture was designed to alleviate the deficiencies of the SPAN architecture (O'Neill, 2007).

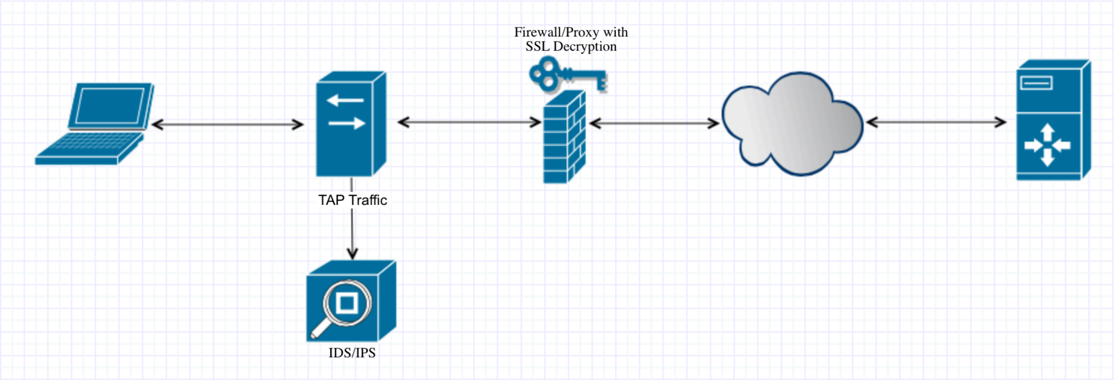


Figure 3. TAP

2.2 Netflow

To convict anomalies, the analyst must understand the flow data collected at the inspection points. Flow data can include context such as:

1. Source IP address
2. Destination IP address
3. Source Port
4. Destination Port
5. Protocol
6. Amount of data transferred
7. Connection length
8. Time of connection

The capture of this data helps two aspects of network intrusion analysis. First, during incident analysis, the flow data becomes a record to add context around the incident for forensic archiving. The usage during incident analysis is reactive. Comparing current flow data to historical flow data can proactively alert for anomalous behavior (Gennuso, 2012, page 9-10). Understanding what flows are normal assists with inspection rule development and intrusion analysis (proactive and reactive).

The attributes of flow data are extracted from layer 3 (Network) and layer 4 (Transport). The data can be similarly compared to a telephone call's detail. The telephone call documents who made the call (Source IP Address), to whom the call was made (Destination IP address, the duration of the call (Connection length), and what time the call was placed (Time of connection). What is missing is the content of the call.

2.3 Applications and Data

Content inspection focuses on application and data. Application and data inspection require the inspection platform to be able to interpret layers 5-7 (Session/Presentation/Application). These specific layers provide the content that was unavailable in the example from section 2.2.

Application inspection attempts to determine what application is responsible for the communication. How can the analyst determine if the Skype for Business application is responsible for the communication when netflow identifies layer 4 is TCP and destination port 80 and not http? These are the same flow characteristics as a browser connection to http://google.com. This answer is dependent upon whether the inspection continues beyond layer 4.

The Skype vs http scenario is easier to determine since the applications are unencrypted. Encrypted payloads are quickly becoming the standard for network traffic (Moscaritolo, 2016). As more internet content leverages secure sockets layer/transport layer security (SSL/TLS), the more difficult application inspection becomes.

Next-generation intrusion inspection platforms continue to inspect the session layer header of the packet for known application headers. Frameworks such as Protocol Identification Analyzers (PIA) provide the capability to determine what application is responsible for the communication (Dreger, Feldmann, Mai, Paxson, Sommer, section 4.1). The application headers become a signature to determine what application is involved in the communication. The signature is capable of determining behaviorally how these applications communicate in comparison to other known applications.

A great deal of attention is required during the architecture design if the final intention is to provide visibility to encrypted traffic. In order to inspect encrypted payloads, the architecture must be designed to provide the significant amount of resources required to decrypt and re-encrypt (if necessary). In addition, the sensitivity of the data to be decrypted must be managed. Several legal regulations exist for decrypting specific traffic types; for instance banking and healthcare.

The analyst's ability to inspect and interpret content enables full intrusion inspection capability. Architectural design intent is to prevent 'blind spots' in the network intrusion inspection capability.

3. Duplicate Packet Analysis and Inspections

Duplicate packet analysis and inspection refers to a single packet evaluated multiple times. Wireshark states this is not an uncommon occurrence/practice if the intention is to inspect packet context and packet content separately (Wireshark, 2008). A properly designed architecture is capable of accomplishing with limited issues. Issues arise when the analysis of a packet (or stream) is reviewed by an analyst without architectural understanding or when the inspection is accomplished by the same sensor multiple times. Three scenarios below demonstrate the architectures, outputs, and the possible issues experienced.

3.1 Wireshark Analysis

The architecture in Figure 4 allows the switch and firewall to send the packet data at the ingress interfaces. These packets are then duplicated via SPAN and sent to a laptop running the Wireshark application. The SPAN traffic sent from the switch is encrypted and is presented on the left-hand side of Figure 5. The SSL decrypted and port mirrored traffic is sent from the firewall and is presented on the right of Figure 5.

Each capture in Figure 5 was done individually to show how side by side packet analysis can be done between encrypted and decrypted traffic. Notice the filters are searching for TCP Streams. TCP Streams are a logical grouping for the TCP connection. Within the packets on client or server side, there is not a value equating for a TCP Stream. The TCP Stream is applied by Wireshark to be able to track the session from beginning to end.

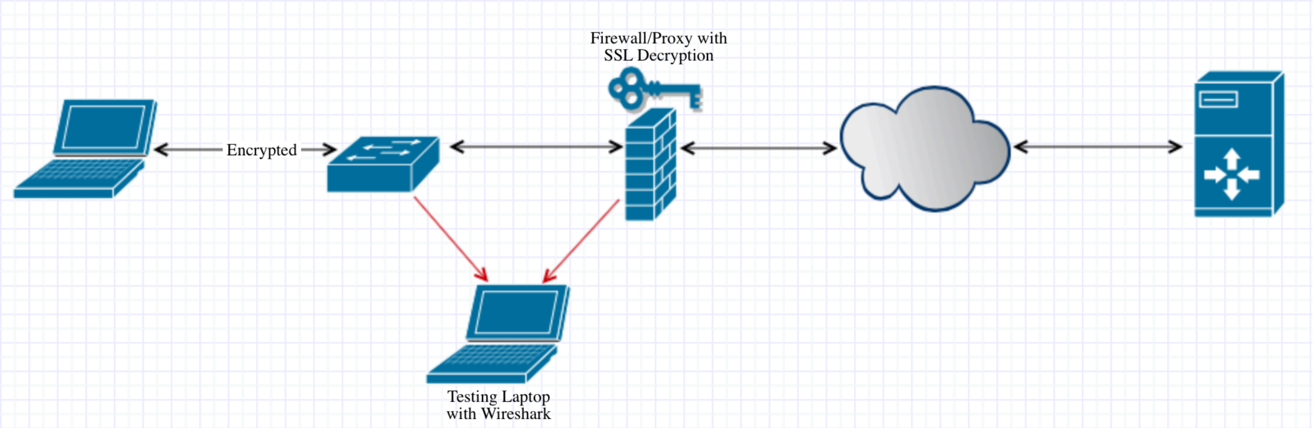
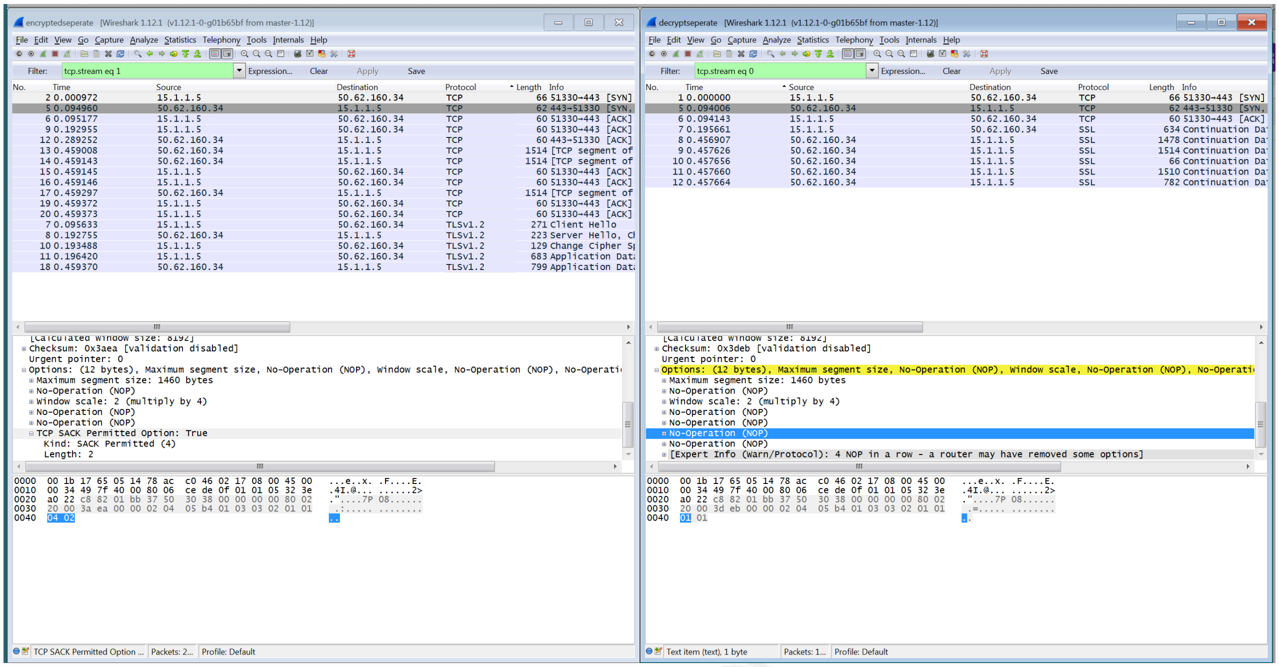

Firewall/Proxy with SSL Decryption
Encrypted
Testing Laptop with Wireshark

Figure 4. Wireshark Analysis

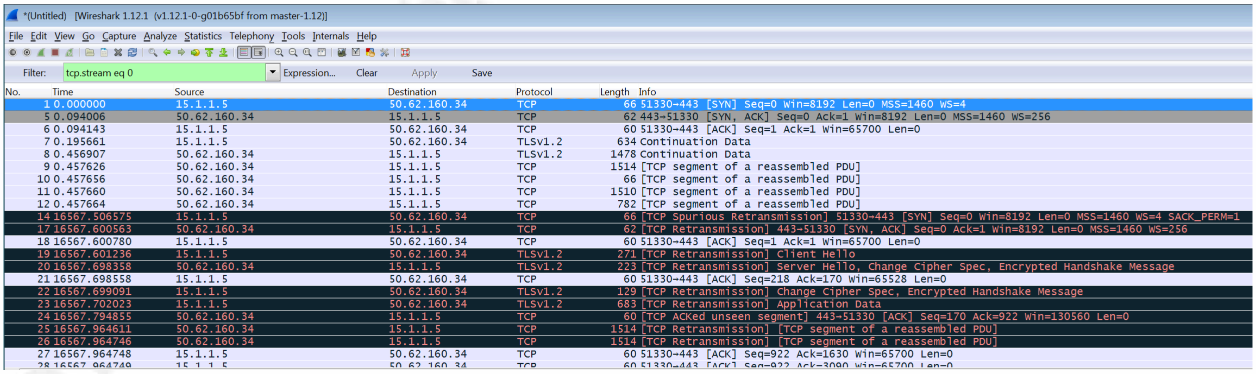

encryptedseperate [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
TCP SACK Permitted Option ... Packets: 2... Profile: Default
decryptseperate [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
Text item (text), 1 byte Packets: 1... Profile: Default

```
Filter: tcp.stream eq 1

No.  Time      Source        Destination   Protocol  Length Info
  2 0.000972  15.1.1.5      50.62.160.34  TCP          66 51330→443 [SYN]
  5 0.094960  50.62.160.34  15.1.1.5      TCP          62 443→51330 [SYN,
  6 0.095177  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK]
  9 0.192955  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK]
 12 0.289252  50.62.160.34  15.1.1.5      TCP          60 443→51330 [ACK]
 13 0.459008  50.62.160.34  15.1.1.5      TCP        1514 [TCP segment of
 14 0.459143  50.62.160.34  15.1.1.5      TCP        1514 [TCP segment of
 15 0.459145  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK]
 16 0.459146  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK]
 17 0.459297  50.62.160.34  15.1.1.5      TCP        1514 [TCP segment of
 19 0.459372  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK]
 20 0.459373  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK]
  7 0.095633  15.1.1.5      50.62.160.34  TLSv1.2     271 Client Hello
  8 0.192755  50.62.160.34  15.1.1.5      TLSv1.2     223 Server Hello, Ch
 10 0.193488  15.1.1.5      50.62.160.34  TLSv1.2     129 Change Cipher Sp
 11 0.196420  15.1.1.5      50.62.160.34  TLSv1.2     683 Application Data
 18 0.459370  50.62.160.34  15.1.1.5      TLSv1.2     799 Application Data

  [Calculated window size: 8192]
⊞ Checksum: 0x3aea [validation disabled]
  Urgent pointer: 0
⊟ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operati
  ⊞ Maximum segment size: 1460 bytes
  ⊞ No-Operation (NOP)
  ⊞ Window scale: 2 (multiply by 4)
  ⊞ No-Operation (NOP)
  ⊞ No-Operation (NOP)
  ⊟ TCP SACK Permitted Option: True
      Kind: SACK Permitted (4)
      Length: 2

0000  00 1b 17 65 05 14 78 ac  c0 46 02 17 08 00 45 00   ...e..x. .F....E.
0010  00 34 49 7f 40 00 80 06  ce de 0f 01 01 05 32 3e   .4I.@... ......2>
0020  a0 22 c8 82 01 bb 37 50  30 38 00 00 00 00 80 02   ."....7P 08......
0030  20 00 3a ea 00 00 02 04  05 b4 01 03 03 02 01 01    .:..... ........
0040  04 02                                              ..
```

```
Filter: tcp.stream eq 0

No.  Time      Source        Destination   Protocol  Length Info
  1 0.000000  15.1.1.5      50.62.160.34  TCP          66 51330→443 [SYN]
  5 0.094006  50.62.160.34  15.1.1.5      TCP          62 443→51330 [SYN,
  6 0.094143  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK]
  7 0.195661  15.1.1.5      50.62.160.34  SSL         634 Continuation Da
  8 0.456907  50.62.160.34  15.1.1.5      SSL        1478 Continuation Da
  9 0.457626  50.62.160.34  15.1.1.5      SSL        1514 Continuation Da
 10 0.457656  50.62.160.34  15.1.1.5      SSL          66 Continuation Da
 11 0.457660  50.62.160.34  15.1.1.5      SSL        1510 Continuation Da
 12 0.457664  50.62.160.34  15.1.1.5      SSL         782 Continuation Da

  [Calculated window size: 8192]
⊞ Checksum: 0x3deb [validation disabled]
  Urgent pointer: 0
⊟ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operati
  ⊞ Maximum segment size: 1460 bytes
  ⊞ No-Operation (NOP)
  ⊞ Window scale: 2 (multiply by 4)
  ⊞ No-Operation (NOP)
  ⊞ No-Operation (NOP)
  ⊞ No-Operation (NOP)
  ⊞ No-Operation (NOP)
  ⊞ [Expert Info (Warn/Protocol): 4 NOP in a row - a router may have removed some options]

0000  00 1b 17 65 05 14 78 ac  c0 46 02 17 08 00 45 00   ...e..x. .F....E.
0010  00 34 49 7f 40 00 80 06  ce de 0f 01 01 05 32 3e   .4I.@... ......2>
0020  a0 22 c8 82 01 bb 37 50  30 38 00 00 00 00 80 02   ."....7P 08......
0030  20 00 3d eb 00 00 02 04  05 b4 01 03 03 02 01 01    .=..... ........
0040  01 01                                              ..
```

Figure 5. Wireshark Output

More commonly, the analysis is accomplished simultaneously as the packet traverses the inspection points. When reviewed in Wireshark there becomes a significant difference of the output. Figure 6 appears to have more complexity for the exact same traffic.

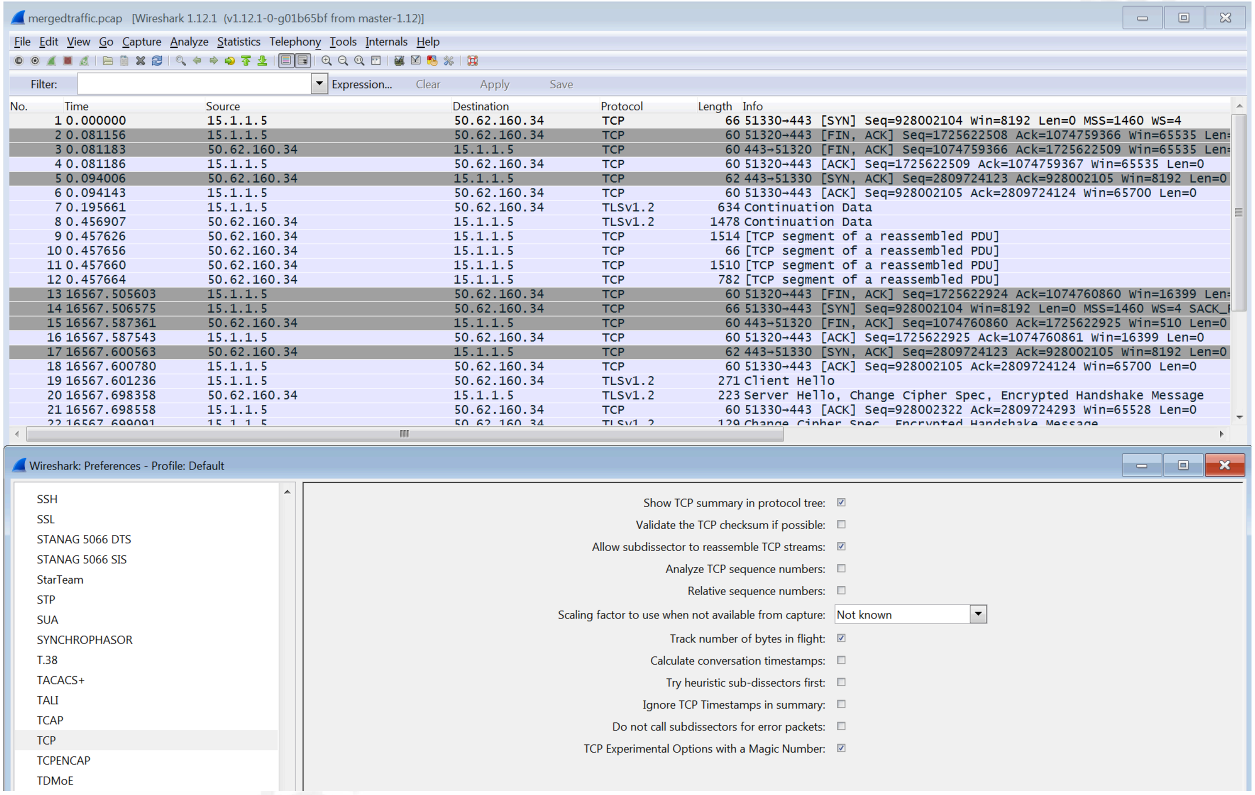

*(Untitled) [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
Filter: tcp.stream eq 0

No.  Time          Source        Destination   Protocol  Length Info
  1 0.000000      15.1.1.5      50.62.160.34  TCP          66 51330→443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4
  5 0.094006      50.62.160.34  15.1.1.5      TCP          62 443→51330 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256
  6 0.094143      15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  7 0.195661      15.1.1.5      50.62.160.34  TLSv1.2     634 Continuation Data
  8 0.456907      50.62.160.34  15.1.1.5      TLSv1.2    1478 Continuation Data
  9 0.457626      50.62.160.34  15.1.1.5      TCP        1514 [TCP segment of a reassembled PDU]
 10 0.457656      50.62.160.34  15.1.1.5      TCP          66 [TCP segment of a reassembled PDU]
 11 0.457660      50.62.160.34  15.1.1.5      TCP        1510 [TCP segment of a reassembled PDU]
 12 0.457664      50.62.160.34  15.1.1.5      TCP         782 [TCP segment of a reassembled PDU]
 14 16567.506575  15.1.1.5      50.62.160.34  TCP          66 [TCP Spurious Retransmission] 51330→443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 17 16567.600563  50.62.160.34  15.1.1.5      TCP          62 [TCP Retransmission] 443→51330 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256
 18 16567.600780  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
 19 16567.601236  15.1.1.5      50.62.160.34  TLSv1.2     271 [TCP Retransmission] Client Hello
 20 16567.698358  50.62.160.34  15.1.1.5      TLSv1.2     223 [TCP Retransmission] Server Hello, Change Cipher Spec, Encrypted Handshake Message
 21 16567.698558  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK] Seq=218 Ack=170 Win=65528 Len=0
 22 16567.699091  15.1.1.5      50.62.160.34  TLSv1.2     129 [TCP Retransmission] Change Cipher Spec, Encrypted Handshake Message
 23 16567.702023  15.1.1.5      50.62.160.34  TLSv1.2     683 [TCP Retransmission] Application Data
 24 16567.794855  50.62.160.34  15.1.1.5      TCP          60 [TCP ACKed unseen segment] 443→51330 [ACK] Seq=170 Ack=922 Win=130560 Len=0
 25 16567.964611  50.62.160.34  15.1.1.5      TCP        1514 [TCP Retransmission] [TCP segment of a reassembled PDU]
 26 16567.964746  50.62.160.34  15.1.1.5      TCP        1514 [TCP Retransmission] [TCP segment of a reassembled PDU]
 27 16567.964748  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK] Seq=922 Ack=1630 Win=65700 Len=0
 28 16567.964749  15.1.1.5      50.62.160.34  TCP          60 51330→443 [ACK] Seq=922 Ack=3090 Win=65700 Len=0
```

Figure 6. Wireshark Merged TCP Streams

The merged TCP Streams of Wireshark has made the assumption that the encrypted packets from the switch are retransmissions of the decrypted packets from the firewall. Wireshark is not aware that there are two separate inspection points feeding this packet capture. An analyst reviewing the merged packet capture could make incorrect assumptions during in investigation if there is an incorrect understanding of architecture. An analyst could spend wasted time troubleshooting the significantly high volume of packet retransmission issue, which are normally associated with network or application congestion issues.

To prevent the analyst from confusion Wireshark has the capability to remove the TCP Streams that Wireshark applies. To remove the stream analysis, disable the “Analyze TCP sequence numbers” and “Relative sequence numbers” from the Preferences section for TCP.


mergedtraffic.pcap [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
Clear
Apply
Save

```
No.  Time          Source        Destination   Protocol  Length Info
  1 0.000000       15.1.1.5      50.62.160.34  TCP         66 51330→443 [SYN] Seq=928002104 Win=8192 Len=0 MSS=1460 WS=4
  2 0.081156       15.1.1.5      50.62.160.34  TCP         60 51320→443 [FIN, ACK] Seq=1725622508 Ack=1074759366 Win=65535 Len=
  3 0.081183       50.62.160.34  15.1.1.5      TCP         60 443→51320 [FIN, ACK] Seq=1074759366 Ack=1725622509 Win=65535 Len=
  4 0.081186       15.1.1.5      50.62.160.34  TCP         60 51320→443 [ACK] Seq=1725622509 Ack=1074759367 Win=65535 Len=0
  5 0.094006       50.62.160.34  15.1.1.5      TCP         62 443→51330 [SYN, ACK] Seq=2809724123 Ack=928002105 Win=8192 Len=0
  6 0.094143       15.1.1.5      50.62.160.34  TCP         60 51330→443 [ACK] Seq=928002105 Ack=2809724124 Win=65700 Len=0
  7 0.195661       15.1.1.5      50.62.160.34  TLSv1.2    634 Continuation Data
  8 0.456907       50.62.160.34  15.1.1.5      TLSv1.2   1478 Continuation Data
  9 0.457626       50.62.160.34  15.1.1.5      TCP       1514 [TCP segment of a reassembled PDU]
 10 0.457656       50.62.160.34  15.1.1.5      TCP         66 [TCP segment of a reassembled PDU]
 11 0.457660       50.62.160.34  15.1.1.5      TCP       1510 [TCP segment of a reassembled PDU]
 12 0.457664       50.62.160.34  15.1.1.5      TCP        782 [TCP segment of a reassembled PDU]
 13 16567.505603   15.1.1.5      50.62.160.34  TCP         60 51320→443 [FIN, ACK] Seq=1725622924 Ack=1074760860 Win=16399 Len=
 14 16567.506575   15.1.1.5      50.62.160.34  TCP         66 51330→443 [SYN] Seq=928002104 Win=8192 Len=0 MSS=1460 WS=4 SACK_
 15 16567.587361   50.62.160.34  15.1.1.5      TCP         60 443→51320 [FIN, ACK] Seq=1074760860 Ack=1725622925 Win=510 Len=0
 16 16567.587543   15.1.1.5      50.62.160.34  TCP         60 51320→443 [ACK] Seq=1725622925 Ack=1074760861 Win=16399 Len=0
 17 16567.600563   50.62.160.34  15.1.1.5      TCP         62 443→51330 [SYN, ACK] Seq=2809724123 Ack=928002105 Win=8192 Len=0
 18 16567.600780   15.1.1.5      50.62.160.34  TCP         60 51330→443 [ACK] Seq=928002105 Ack=2809724124 Win=65700 Len=0
 19 16567.601236   15.1.1.5      50.62.160.34  TLSv1.2    271 Client Hello
 20 16567.698358   50.62.160.34  15.1.1.5      TLSv1.2    223 Server Hello, Change Cipher Spec, Encrypted Handshake Message
 21 16567.698558   15.1.1.5      50.62.160.34  TCP         60 51330→443 [ACK] Seq=928002322 Ack=2809724293 Win=65528 Len=0
 22 16567.699091   15.1.1.5      50.62.160.34  TLSv1.2    129 Change Cipher Spec, Encrypted Handshake Message
```

Show TCP summary in protocol tree: ☑
Validate the TCP checksum if possible: □
Allow subdissector to reassemble TCP streams: ☑
Analyze TCP sequence numbers: □
Relative sequence numbers: □
Scaling factor to use when not available from capture: Not known
Track number of bytes in flight: ☑
Calculate conversation timestamps: □
Try heuristic sub-dissectors first: □
Ignore TCP Timestamps in summary: □
Do not call subdissectors for error packets: □
TCP Experimental Options with a Magic Number: ☑

Figure 7. Wireshark Preferences

3.2 Snort Inspection

Wireshark is an analysis tool only. Snort is an open source network intrusion detection/prevention tool with the ability to inspect from a security perspective. Traffic inspection can take place architecturally inline or from a SPAN or TAP. From the inspection process, perspective security rules can be applied to inspect packets and/or streams. Packets are inspected individually and streams leverage preprocessors to maintain track of the session in order to detect anomalous or malicious behavior throughout.

The architecture in Figure 7 employs both the inline inspection and the SPAN inspection.

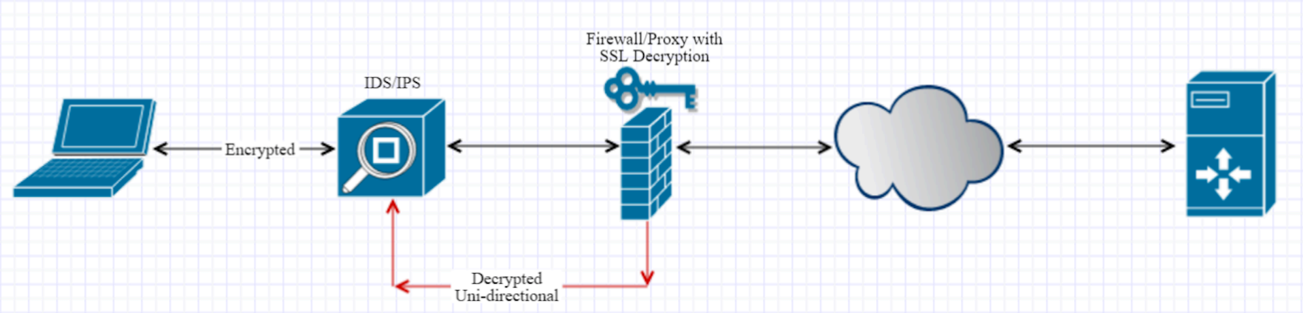
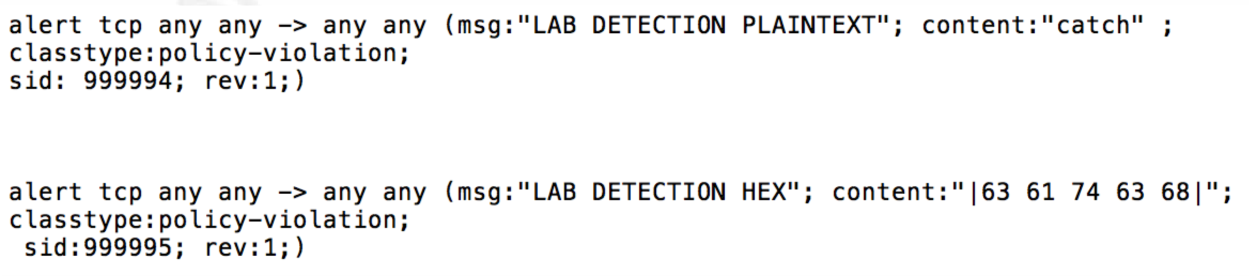


Figure 8. Single Sensor Inline inspection

Testing was accomplished on packets only without any preprocessors enabled. The first inspection point has encrypted payload inline. The second inspection takes place with decrypted payload in SPAN mode. From the laptop, a web browser connection was established to DataLeakTest.com [3]. The DataLeakTest.com website has the ability to test http and https POST commands. Traditionally this is used to verify data loss prevention tools are working properly, but in this scenario, it provides a simple and repeatable method to test Snort rules.

The rules are configured to alert if either the plaintext word “catch” is in the payload or if the equivalent HEX appears. Figure 8 displays the configured rules. These basic rules read in plain English as follows:

“Alert if there is tcp traffic seen from any host on any port to any other host on any port containing “catch” (or HEX values) in the payload. If alerting use the following message (msg).”



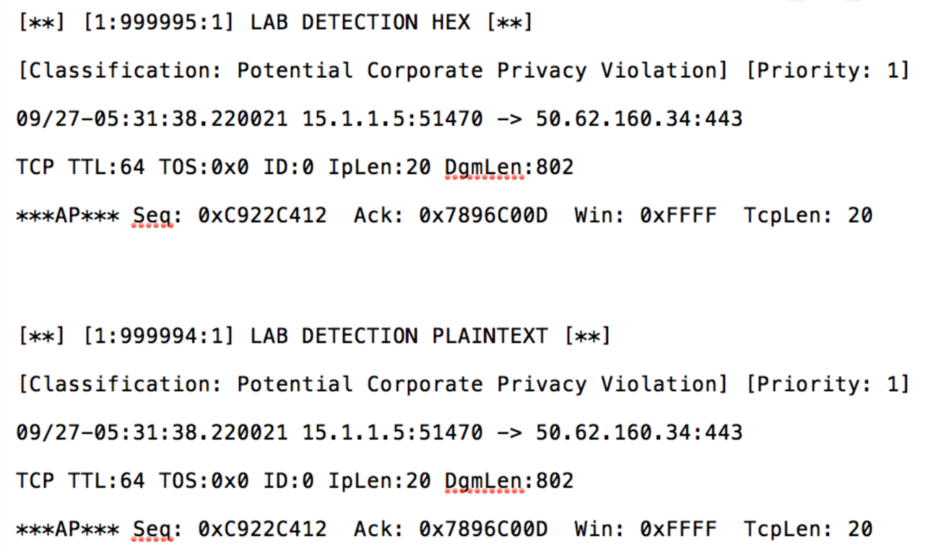
```
alert tcp any any -> any any (msg:"LAB DETECTION PLAINTEXT"; content:"catch" ;
classtype:policy-violation;
sid: 999994; rev:1;)


alert tcp any any -> any any (msg:"LAB DETECTION HEX"; content:"|63 61 74 63 68|";
classtype:policy-violation;
 sid:999995; rev:1;)
```

Figure 9. Snort Rules

[3] http://dataleaktest.com/

During this test, the 'HTTPS' mode was utilized on DataLeakTest.com. The https post was the word "catch", which in the packet would also contain the corresponding HEX values. Both rules fired but only a single time, Figure 9. This demonstrates two points:

1. Given the architecture in Figure 7, the first inline inspection point is unable to evaluate payload due to the encryption.
2. The second SPAN inspection occurs and evaluated all rules and did not stop after the first match. Was able to inspect the decrypted payload and generate the alerts.



```
[**] [1:999995:1] LAB DETECTION HEX [**]
[Classification: Potential Corporate Privacy Violation] [Priority: 1]
09/27-05:31:38.220021 15.1.1.5:51470 -> 50.62.160.34:443
TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:802
***AP*** Seq: 0xC922C412  Ack: 0x7896C00D  Win: 0xFFFF  TcpLen: 20


[**] [1:999994:1] LAB DETECTION PLAINTEXT [**]
[Classification: Potential Corporate Privacy Violation] [Priority: 1]
09/27-05:31:38.220021 15.1.1.5:51470 -> 50.62.160.34:443
TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:802
***AP*** Seq: 0xC922C412  Ack: 0x7896C00D  Win: 0xFFFF  TcpLen: 20
```

Figure 10. Snort Alerts

These points are important for the analyst to be aware of during alert investigation. Understanding of architecture and expected data flows are critical to reducing investigation times. There is not enough content in the rules or alerts alone to indicate where in the network the alert was triggered.

To increase analyst awareness and context for the alerts, policies should utilize the flexibility of the message (msg:) section. The use of commas, semicolons, and vertical bars are the only limitation for the message section. An example to enhance would be to make sid:999995 msg: "LAB DETECTION HEX – LOCATION=INLINE – SENSOR ID=X" and sid:999994 msg: "LAB DETECTION PLAINTEXT – LOCATION=SPAN – SENSOR ID=X". Sensor identification (SENSOR ID=) would indicate which sensor is generating the alert in the network.

3.3 Next Generation Security Platform Inspection

The final testing occurs with two inline inspection points on the same unified threat management (UTM) platform. The architecture in Figure 10 depicts the flow from the laptop to an interface on the UTM, then out an interface back into a third interface; from that third interface to a forth externally facing interface.

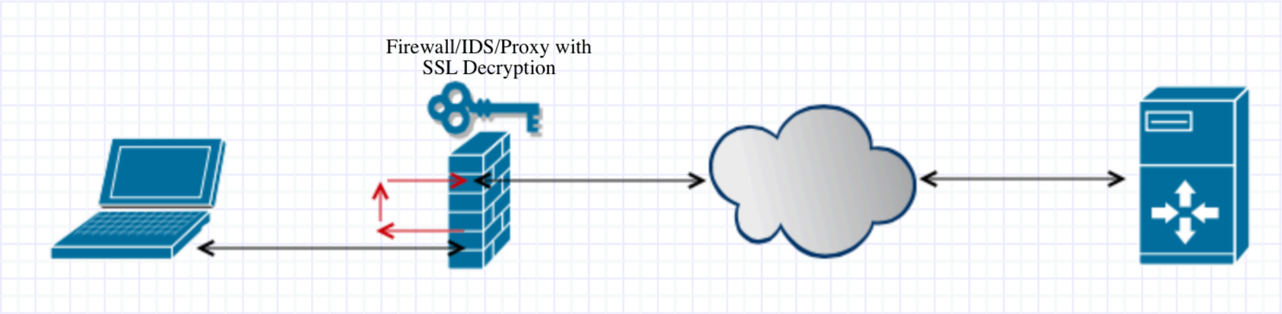
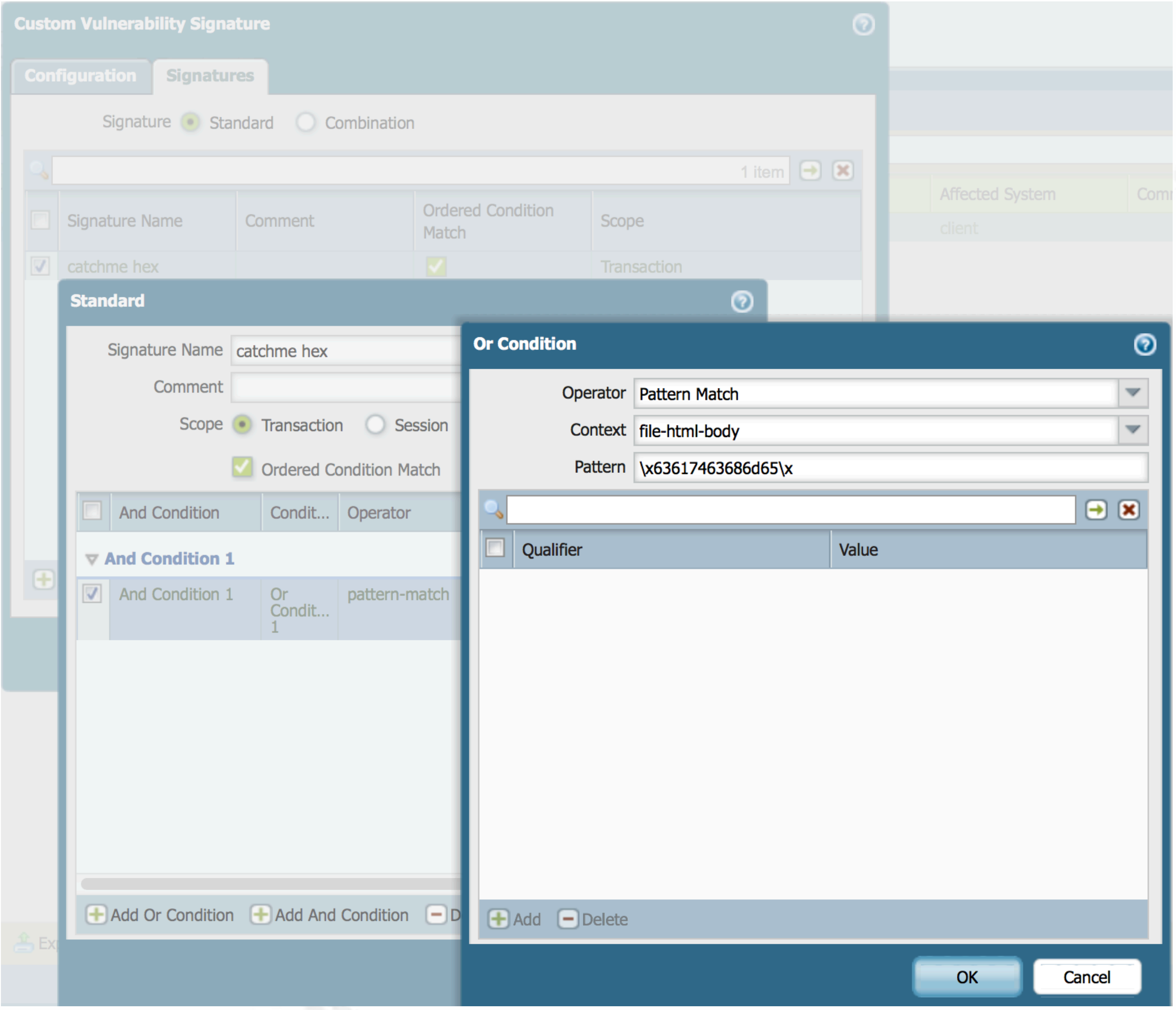


Figure 11. Next Generation Security Platform Inspection

The custom vulnerability signature strings for this specific UTM are required to inspect payload content with 7-byte minimums. For this reason, the string “catch” from the Snort scenario must contain two additional characters. Figure 11 reveals the string to be inspected for is “catchme” converted to HEX.



Signature ● Standard ○ Combination

□	Signature Name	Comment	Ordered Condition Match	Scope
☑	catchme hex		✓	Transaction

Signature Name: catchme hex

Comment:

Scope: ● Transaction ○ Session

☑ Ordered Condition Match

□	And Condition	Condit...	Operator
	▽ And Condition 1		
☑	And Condition 1	Or Condit... 1	pattern-match

Operator: Pattern Match

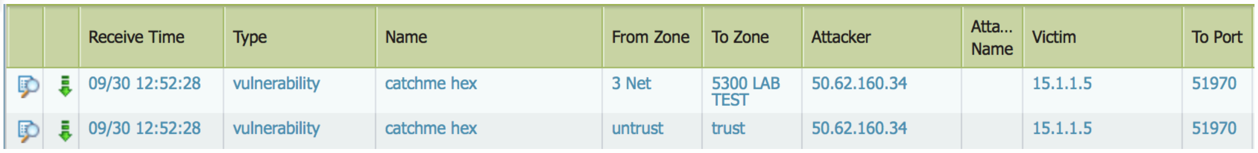
Context: file-html-body

Pattern: \x63617463686d65\x

□	Qualifier	Value

Figure 12. Custom Vulnerability Signature

To validate the signature works, an 'HTTP' session with DataLeakTest.com was used. Both sets of policies were able to inspect the unencrypted traffic and detect the "catchme" signature within the payload. The details in Figure 12 show two separate alerts, with different "From Zone" and "To Zone" values showing the custom vulnerability having been detected.

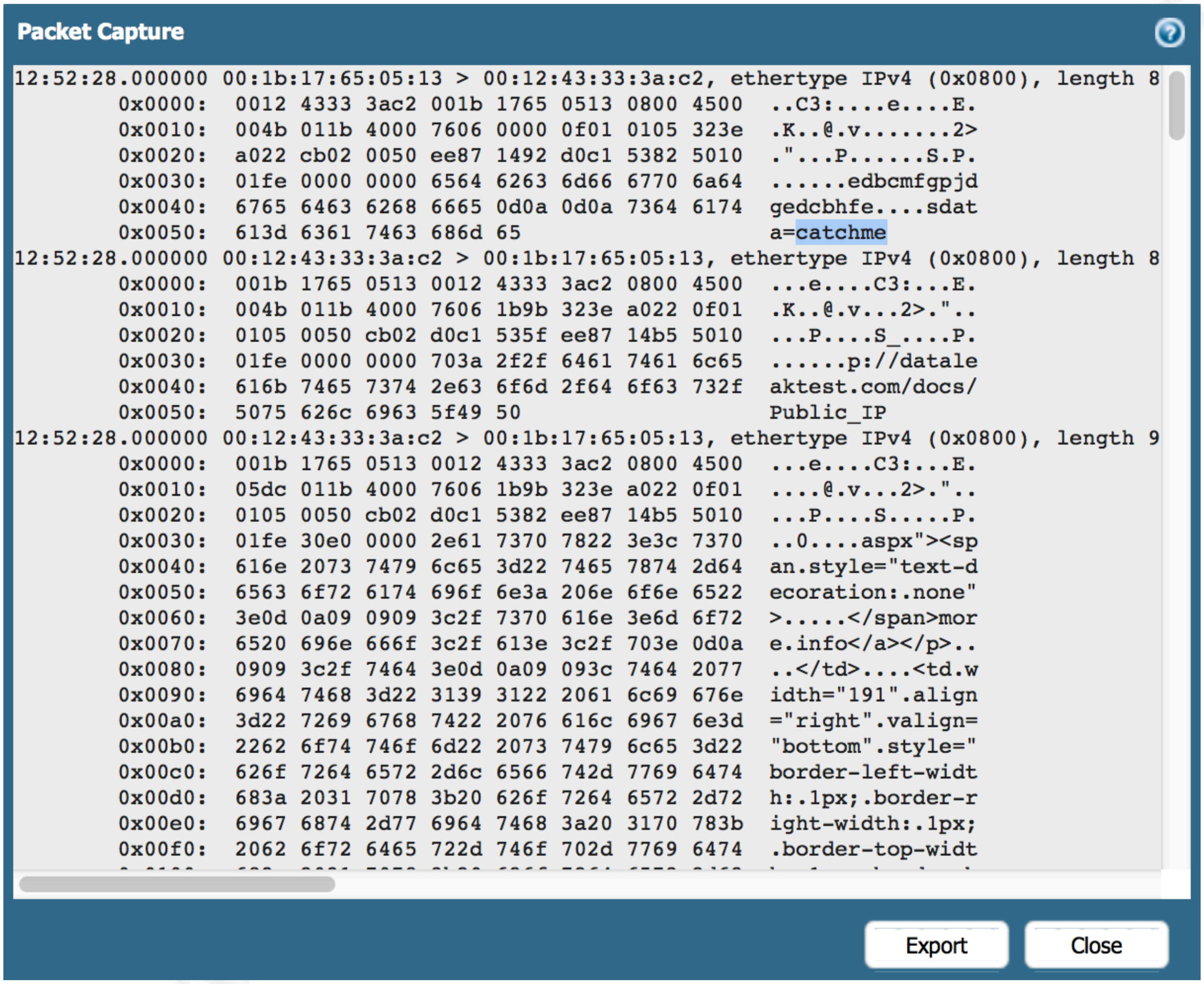


		Receive Time	Type	Name	From Zone	To Zone	Attacker	Atta... Name	Victim	To Port
		09/30 12:52:28	vulnerability	catchme hex	3 Net	5300 LAB TEST	50.62.160.34		15.1.1.5	51970
		09/30 12:52:28	vulnerability	catchme hex	untrust	trust	50.62.160.34		15.1.1.5	51970

Figure 13. Custom Vulnerability Signature Validation

Randy Devlin, rdevlin@mastersprogram.sans.edu

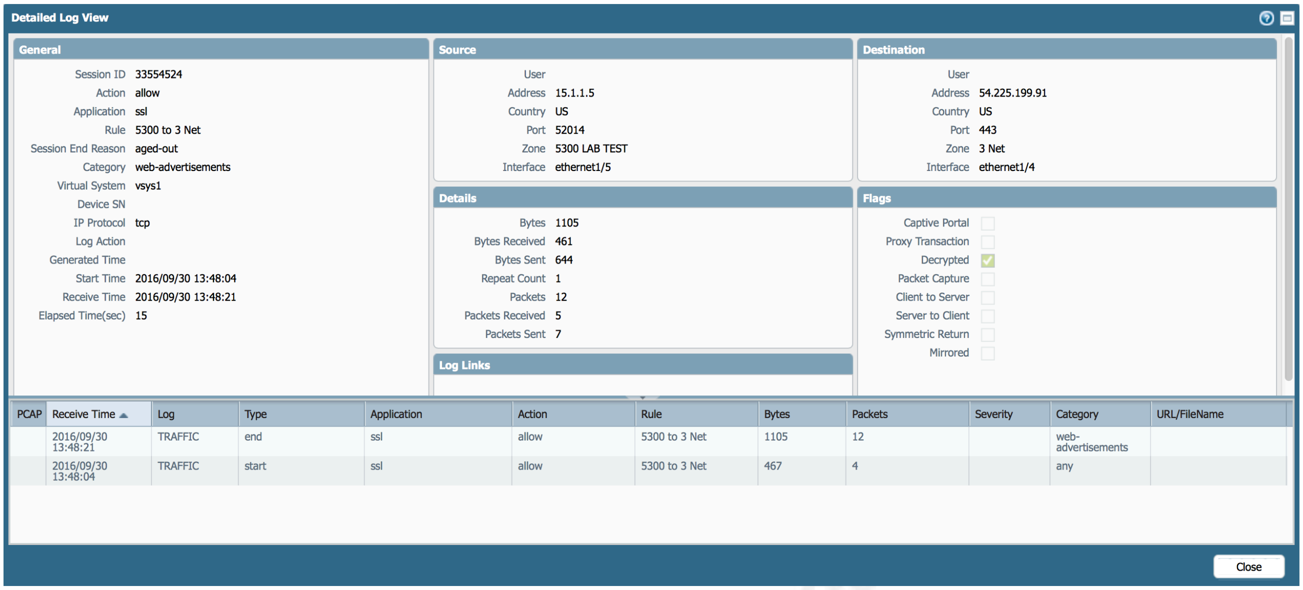
The UTM alerting focuses on session based detections. Deeper analysis can be viewed by clicking the green down arrows. This indicates there is an associated packet capture for the triggered alert with the raw packet available, Figure 13.



```
12:52:28.000000 00:1b:17:65:05:13 > 00:12:43:33:3a:c2, ethertype IPv4 (0x0800), length 8
        0x0000:  0012 4333 3ac2 001b 1765 0513 0800 4500  ..C3:....e....E.
        0x0010:  004b 011b 4000 7606 0000 0f01 0105 323e  .K..@.v.......2>
        0x0020:  a022 cb02 0050 ee87 1492 d0c1 5382 5010  ."...P......S.P.
        0x0030:  01fe 0000 0000 6564 6263 6d66 6770 6a64  ......edbcmfgpjd
        0x0040:  6765 6463 6268 6665 0d0a 0d0a 7364 6174  gedcbhfe....sdat
        0x0050:  613d 6361 7463 686d 65                   a=catchme
12:52:28.000000 00:12:43:33:3a:c2 > 00:1b:17:65:05:13, ethertype IPv4 (0x0800), length 8
        0x0000:  001b 1765 0513 0012 4333 3ac2 0800 4500  ...e....C3:...E.
        0x0010:  004b 011b 4000 7606 1b9b 323e a022 0f01  .K..@.v...2>."..
        0x0020:  0105 0050 cb02 d0c1 535f ee87 14b5 5010  ...P....S_....P.
        0x0030:  01fe 0000 0000 703a 2f2f 6461 7461 6c65  ......p://datale
        0x0040:  616b 7465 7374 2e63 6f6d 2f64 6f63 732f  aktest.com/docs/
        0x0050:  5075 626c 6963 5f49 50                   Public_IP
12:52:28.000000 00:12:43:33:3a:c2 > 00:1b:17:65:05:13, ethertype IPv4 (0x0800), length 9
        0x0000:  001b 1765 0513 0012 4333 3ac2 0800 4500  ...e....C3:...E.
        0x0010:  05dc 011b 4000 7606 1b9b 323e a022 0f01  ....@.v...2>."..
        0x0020:  0105 0050 cb02 d0c1 5382 ee87 14b5 5010  ...P....S.....P.
        0x0030:  01fe 30e0 0000 2e61 7370 7822 3e3c 7370  ..0....aspx"><sp
        0x0040:  616e 2073 7479 6c65 3d22 7465 7874 2d64  an.style="text-d
        0x0050:  6563 6f72 6174 696f 6e3a 206e 6f6e 6522  ecoration:.none"
        0x0060:  3e0d 0a09 0909 3c2f 7370 616e 3e6d 6f72  >.....</span>mor
        0x0070:  6520 696e 666f 3c2f 613e 3c2f 703e 0d0a  e.info</a></p>..
        0x0080:  0909 3c2f 7464 3e0d 0a09 093c 7464 2077  ..</td>....<td.w
        0x0090:  6964 7468 3d22 3139 3122 2061 6c69 676e  idth="191".align
        0x00a0:  3d22 7269 6768 7422 2076 616c 6967 6e3d  ="right".valign=
        0x00b0:  2262 6f74 746f 6d22 2073 7479 6c65 3d22  "bottom".style="
        0x00c0:  626f 7264 6572 2d6c 6566 742d 7769 6474  border-left-widt
        0x00d0:  683a 2031 7078 3b20 626f 7264 6572 2d72  h:.1px;.border-r
        0x00e0:  6967 6874 2d77 6964 7468 3a20 3170 783b  ight-width:.1px;
        0x00f0:  2062 6f72 6465 722d 746f 702d 7769 6474  .border-top-widt
```

Figure 14. Custom Vulnerability Signature Validation

In testing the visibility of https traffic, using DataLeakTest.com, SSL https POST was enabled, and the same security policies with the "catchme" signature are enabled. A decryption profile on the UTM was set up solely for the second pass through the UTM. In this scenario, the traffic was detected by the UTM, but the signature never triggered an alert. The detailed log view indicates the traffic was decrypted in the Flags pane. Figure 14 illustrates the log.



General

Session ID	33554524
Action	allow
Application	ssl
Rule	5300 to 3 Net
Session End Reason	aged-out
Category	web-advertisements
Virtual System	vsys1
Device SN	
IP Protocol	tcp
Log Action	
Generated Time	
Start Time	2016/09/30 13:48:04
Receive Time	2016/09/30 13:48:21
Elapsed Time(sec)	15

Source

User	
Address	15.1.1.5
Country	US
Port	52014
Zone	5300 LAB TEST
Interface	ethernet1/5

Destination

User	
Address	54.225.199.91
Country	US
Port	443
Zone	3 Net
Interface	ethernet1/4

Details

Bytes	1105
Bytes Received	461
Bytes Sent	644
Repeat Count	1
Packets	12
Packets Received	5
Packets Sent	7

Flags

Captive Portal	□
Proxy Transaction	□
Decrypted	☑
Packet Capture	□
Client to Server	□
Server to Client	□
Symmetric Return	□
Mirrored	□

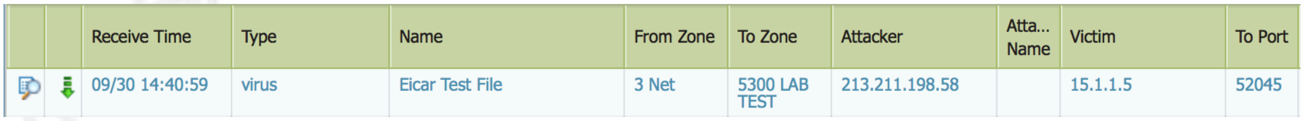
Log Links

PCAP	Receive Time	Log	Type	Application	Action	Rule	Bytes	Packets	Severity	Category	URL/FileName
	2016/09/30 13:48:21	TRAFFIC	end	ssl	allow	5300 to 3 Net	1105	12		web-advertisements	
	2016/09/30 13:48:04	TRAFFIC	start	ssl	allow	5300 to 3 Net	467	4		any	

Figure 15. UTM HTTPS Detailed Log View

There is an issue with the lack of the vulnerability signature not alerting once inspection was accomplished post decryption. A secondary test was completed with the European Expert Group for IT-Security website [4]. Under the download Anti-Malware Test section there is the ability to select http or https downloads. The download of the EICAR.txt file was selected for SSL enabled protocol https download. The download was successful, and the UTM was able to decrypt and detect the malicious test file in the payload, Figure 15.

The successful EICAR test and the http test confirm the decryption is properly configured and the vulnerability signature is accurate. The lack of signature alerting versus test file alerting is under review by the manufacturer.



		Receive Time	Type	Name	From Zone	To Zone	Attacker	Atta... Name	Victim	To Port
		09/30 14:40:59	virus	Eicar Test File	3 Net	5300 LAB TEST	213.211.198.58		15.1.1.5	52045

Figure 16. UTM HTTPS EICAR Download Alert

[4] http://www.eicar.org/85-0-Download.html

4. Conclusion

Review of the three testing scenarios confirms the importance for an analyst to understand architecture, desired network flows, and expected or acceptable data and application communication. Prior to initiating the technical review of the incident evidence, a non-functional review is required. Starting with the technical review may lead to similar issues presented above and further limiting the effectiveness of the investigation. Analysis with the technical first approach and the data captured in the Figures can lead to incorrect assumptions pertaining to the duplication, the relevance of the event, or the lack of visibility to the event.

References

Department of Justice. (2000, January). Crime Scene Investigation: A Guide for Law Enforcement

https://archives.fbi.gov/archives/about-us/lab/forensic-science-communications/fsc/april2000/twgcsi.pdf

Dreger, Feldmann, Mai, Paxson, Sommer. Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection

http://www.icir.org/robin/papers/usenix06/> section 4.1

Gennuso, K. (2012). Shedding Light on Security Incidents Using Network Flows

https://www.sans.org/reading-room/whitepapers/networkdevs/shedding-light-security-incidents-network-flows-33935

Moscaritolo, A. (2016, March 16) 77 Percent of Google Internet Traffic Now Encrypted

http://www.pcmag.com/news/342935/77-percent-of-google-internet-traffic-now-encrypted

Obregon, L. (2015, December 2). Infrastructure Security Architecture for Effective Security Monitoring

https://www.sans.org/reading-room/whitepapers/bestprac/infrastructure-security-architecture-effective-security-monitoring-36512

O'Neill, T. (2007, August 23). SPAN Port or TAP? CSO Beware

http://www.lovemytool.com/blog/2007/08/span-ports-or-t.html

Reichenberg, N. (2013, September 26). Four Tips for Designing a Secure Network Perimeter

http://www.securityweek.com/four-tips-designing-secure-network-perimeter

Sanders, C. (2011, January 17). The 10 Commandments of Intrusion Analysis

http://chrissanders.org/2011/01/the-10-commandments-of-intrusion-analysis/

Wireshark. (2008, April 4). Duplicate Packets

https://wiki.wireshark.org/DuplicatePackets