**BASIC PACKET ANALYSIS**

**TABLE OF CONTENTS**

* **IP Header**
  + **Format & Notes**
  + **Fields**
  + **Dissection**
* **TCP Header**
  + **Format & Notes**
  + **Fields**
  + **Seq Number / Ack Number session graph**
  + **Dissection**
  + **Tcpdump Advanced TCP Filters**
* **UDP Header**
  + **Format & Notes**
  + **Fields**
  + **Dissection**
  + **Tcpdump Advanced UDP Filters**
* **ICMP Echo / Echo Reply Header**
  + **Format & Notes**
  + **Fields**
  + **Dissection**
  + **Tcpdump Advanced ICMP Echo Filters**
* **Loki Backdoor Case**
* **Hex to Decimal / Decimal to Hex Conversions**
* **Ethernet II Frame Format**
* **Basic Packet Analysis Test**

**IP HEADER**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|Version| IHL |Type of Service| Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identification |Flags| Fragment Offset |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Time to Live | Protocol | Header Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Destination Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

**First Notes about the IP Header:**

1. **Every row is 32 bits long (4 bytes)**
2. **Every IP Header field will be identified as a row division:**
   1. **Version**
   2. **IP Header Length (IHL)**
   3. **Type of Service (ToS)**
   4. **Total Length**
   5. **…**
3. **A row without divisions would be a 32 bit IP Header field**
4. **A row with three divisions like**

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Time to Live | Protocol | Header Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  
 **It has two 8 bit IP Header fields and one 16 bit one.**

1. **A row with four divisions like**

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|Version| IHL |Type of Service| Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

**It has two 4 bit IP Header fields, one 8 bit field and one 16 bit one.**

1. **The IP Header is usually 20 bytes long total**

**IP Header Fields:**

* **Version (4 bits. Current version of IP = 4)**
* **IP Header Length (4 bits. Describes how big the header is in 32-bit words. For instance, the minimum / practically standard value is 5 🡪 5\*32-bit words = 5\*4 bytes = 20 bytes, as the minimum size   
  of an IP header that contains all the correct fields (without IP options) is 20 bytes. This allows the receiver to know exactly where the payload data begins.)**
* **Type of Service also known as Differentiated Services Code Point (DSCP) (8 bits. Usually is set to 0, but may indicate particular Quality of Service needs from the network, the DSCP defines precedence on one of a set of class of service features like delay, throughput reliability, cost and normal service.)**
* **Total Length or Size of Datagram (16 bits. This is the combined length of the header and the data - in bytes. Since this field is 16 bits wide, the maximum length of an IP datagram is 65,535 bytes, though most are much smaller.)**
* **Identification (16 bits. Number which together with the source address uniquely identifies every packet – it is used during reassembly of fragmented datagrams.)**
* **Flags (3 bits. A sequence of three flags/bits used to control whether allow to fragment a packet (the don’t Fragment (DF flag), More Fragments (MF flag)) and to indicate the parts of a packet to the receiver (indicate whether the datagram is the last fragment).)**
* **Fragmentation Offset -expressed in 8-byte units- (13 bits. When a datagram is fragmented, it is necessary to reassemble the fragments in the correct order. The fragment offset numbers the fragments in such a way that they can be reassembled correctly at the destination IP Process. The very first fragment has offset zero.**
* **Time to Live (8 bits. Specifies how long the datagram is allowed to “live” on the network in terms of router hops and it is used to prevent accidental routing loops. Each router decrements the value of the TTL field (reduces it by one) prior to transmitting it. If the TTL field reaches zero the datagram is assumed to have taken too long a route so it is discarded by the router and an ICMP Time Exceeded Message is sent to the sender. Even though there are standards for request and response packets TTL values based on the OS, it is possible to change these values at the stack Level. This could help to fool OS fingerprints.)**
* **Protocol (8 bits. Indicates the type of transport protocol being carried e.g. 1 = ICMP; 2 = IGMP; 6 = TCP; 17 = UDP; 41 = IPv6 over IPv4)**
* **Header Checksum (16 bits. A complement checksum inserted by the sender and updated whenever the packet header is modified. It is commonly used to detect information processing errors introduced into the packet inside a router or bridge where the packet is not protected by a link layer cyclic redundancy check.  
    
  Since some header fields often change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed. This is a checksum on the header only, so the data may contain errors. If the header checksum fails, the internet datagram is discarded by the entity which detects the error.)**
* **Source Address (32 bits. The IP address of the sender of the packet)**
* **Destination Address (32 bits. The IP address of the intended destination)**
* **Options (Variable length. Not normally used, but when used, the IP header length will be greater than five 32-bit words to indicate the size of the options field. Various options can be included in the header by a particular vendor's implementation of IP, If options are included, the header must be padded with zeroes so that the header is a multiple of 32 bits, and matches the count of bytes or 32-bit words in the IP Header Length (IHL) field.)**

**IP HEADER FORMAT DISSECTION**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|Version| IHL |Type of Service| Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identification |Flags| Fragment Offset |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Time to Live | Protocol | Header Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Destination Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

**\*Ethernet Header \*TCP Header**

**0000 08 00 20 d1 76 19 00 07 ec b2 d0 0a 08 00 45 00 ...v.........E.**

**0010 00 28 53 96 40 00 71 06 a6 d1 50 75 0e de c0 a8 .(S.@.q...Pu....**

**0020 64 1c 0a 54 1b 58 d4 be 6b 24 a0 bf ca 34 50 10 d..T.X..k$...4P.**

**0030 3f f5 8f f3 00 00 ?...........**

**Dissection:**

IP Version **4**

IHL (32-bit words) **5**

Type of Service **00**

Total Length **00 28**

Identification  **53 96**

Flags **40 (Don’t Fragment)**

Fragment Offset **00**

Time to Live **71**

Protocol **06**

Header Checksum **a6 d1**

Source Address **50 75 0e de**Destination Address **c0 a8 64 1c**

**TCP HEADER**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Port | Destination Port |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Acknowledgment Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data | |U|A|P|R|S|F| |

| Offset| Reserved |R|C|S|S|Y|I| Window |

| | |G|K|H|T|N|N| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Checksum | Urgent Pointer |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

**First Notes about the TCP Header:**

1. **Every row is 32 bits long (4 bytes)**
2. **Every TCP Header field will be identified as a row division:**
   1. **Source Port**
   2. **Destination Port**
   3. **…**
3. **A row without divisions would be a 32 bit TCP Header field (i.e. Sequence Number)**
4. **A row with two divisions like**

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Port | Destination Port |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ **It has two 16 bit TCP Header fields.**

**5) A row with two divisions like** +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ **It has one 24 bit TCP Header field and one 8 bit one.**

**6) The TCP Header is usually 20 bytes long total**

**TCP Header Fields:**

* **Source Port (16 bits. The source port number)**
* **Destination Port (16 bits. The destination port number)**
* **Sequence Number - 32 bits. – It is the sequence number of the first data octet/byte in this segment (except when SYN is present). *If SYN is present, the sequence number is the initial sequence number (ISN) and the first data byte Seq would be ISN+1.***
* **Acknowledgment Number (ACK control bit set) – 32 bits. - This field is used to *acknowledge the receipt of accumulated segments* by the destination and it is directly proportional to the number of data bytes that are received. *It also indicates the value of the next sequence number the other end is expecting to use.*   
  Once a connection is established this control packet is always sent. If the sender does not receive an acknowledgement for a segment transmitted, it will time-out and retransmit.**
* **Sequence Number / Ack Number Explanation (session graph provided below)**

**CLIENT SERVER**

**SYN -> Seq = ISN = 0, Ack = 0 (no data received yet)**

**SYN/ACK -> Seq = ISN = 0, Ack = 1**

**ACK -> Seq = 1 (ISN + 1), Ack = 1 <- First data byte**

**----------------------- Connection Established ------------------------**

**DATA = 27 (Ack, Psh), Seq = 1, Ack = 1**

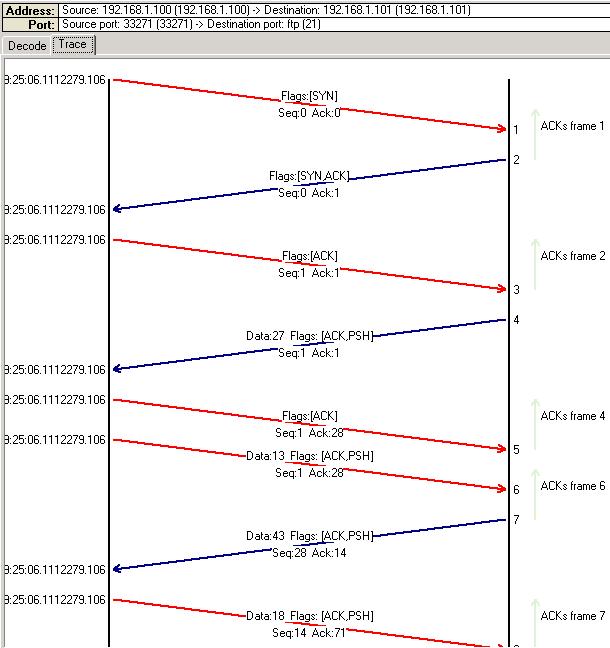
**ACK –> Seq = 1, Ack = 28 (1 + 27 ACKd seg.)**

**DATA = 13 (Ack, Psh), Seq = 1, Ack = 28**

**DATA = 43 (Ack, Psh), Seq = 28, Ack = 14 (1 + 13 ACKd seg.)**

**DATA = 18 (Ack, Psh), Seq = 14, Ack = 71 (1 + 27 + 43)**

**Sequence Number / Ack Number Session Graph   
(partial FTP Session)**

****

* **Data Offset (4 bits. Describes how big the header is in 32-bit words. For instance, the minimum / practically standard value is   
  5 🡪 5\*32-bit words = 5\*4 bytes = 20 bytes, as the minimum size   
  of a TCP header that contains all the correct fields (without TCP options) is 20 bytes. This allows the receiver to know exactly where the payload data begins.)**
* **Reserved (3 bits. Reserved for future use. Must be zero.)**
* **Control Bits (6 bits. TCP Flags)**
  + **U (URG) Urgent pointer. Used to identify incoming data segments as 'urgent' and give them priority so they are sent directly to the stack and processed immediately (i.e. abortion of the listing of a large file using Control-C on a Telnet session (out-of-band signaling)).**
  + **A (ACK) Acknowledgment. It is used to acknowledge the successful receipt of segments.**
  + **P (PSH) Push function. Like the Urgent flag, this tells the stack that the data has to be pushed up with certain priority at both ends. To assure that data submitted to a TCP is actually transmitted, the sending user indicates that it should be pushed through to the receiving user. A push causes the TCPs to promptly forward and deliver data up to that point to the receiver. When a receiving TCP sees the PUSH flag, it must not wait for more data from the sending TCP and pass the data to the receiving process. The PSH flag is used quite frequently at the end (last segment) of a data transfer. An example of using the PSH flag would be real player, where data must be sent and processed (by the receiver) immediately to ensure a smooth stream without any cut offs.**
  + **R (RST) Used to immediately terminate (reset) an existing connection. It is usually seen as a response to a non-available service connection request (i.e. a closed port result during a portscan).**
  + **S (SYN) Synchronize sequence numbers. This flag is initially sent every time a connection is established (3-way Handshake).**
  + **F (FIN) No more data from sender. Used during the teardown of an existing connection. FIN flags always appear on the last packets exchanged during a connection and because TCP is full duplex, FIN flags have to be sent from both sides of the connection to close it gracefully.**
* **Window (16 bits.** **The Window size is considered to be one of the most important flags within the TCP header. This field is used by the receiver to indicate to the sender the amount of data (in bytes) that it is able to accept. The Window size field is the key to efficient data transfers and flow control.)**
* **Checksum (16 bits. The TCP Checksum field was created to ensure that the data contained in the TCP segment reaches the correct destination and is error-free. It is calculated based on a pseudo header of information from the IP header (SRC IP, DST IP, Zeros, Protocol and TCP Segment Length (Header + Data in bytes)), the TCP header and the Data. The pseudo header is not transmitted to the receiver, it is simply involved in the checksum calculation and it will be recreated by the receiver in order to recalculate the TCP header checksum and verify the result. This pseudo header ensures the segment has arrived at the correct receiver.)**
* **Urgent Pointer (16 bits. As we previously mentioned, the URG flag in TCP allows us to mark a segment of data as “urgent”. Using that control bit together, the Urgent Pointer field basically specifies where exactly the urgent data ends. This way the application will know the right location of the urgent data.)**
* **Options (Variable. TCP Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. This means that if we use one TCP Option that is 4 bits in length, there must be another 4 bits of padding in order to comply with the TCP RFC. TCP options commonly available are Maximum Segment Size (MSS), Window Scaling, Selective Acknowledgements (SACK), Timestamps and Nop.)**
* **Padding (Variable. The TCP header padding is used to ensure that the TCP header length is a multiple of 32 bits. The padding is composed of zeros.)**

**TCP HEADER FORMAT DISSECTION**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Port | Destination Port |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Acknowledgment Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data | |U|A|P|R|S|F| |

| Offset| Reserved |R|C|S|S|Y|I| Window |

| | |G|K|H|T|N|N| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Checksum | Urgent Pointer |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

**\*Ethernet Header** **\*IP Header**

**0000 08 00 20 d1 76 19 00 07 ec b2 d0 0a 08 00 45 00 ...v.........E.**

**0010 00 28 53 96 40 00 71 06 a6 d1 50 75 0e de c0 a8 .(S.@.q...Pu....**

**0020 64 1c 0a 54 1b 58 d4 be 6b 24 a0 bf ca 34 50 10 d..T.X..k$...4P.**

**0030 3f f5 8f f3 00 00 ?...........**

**Dissection:**

Source Port **0a 54**

Destination Port **1b 58**

Sequence Number **d4 be 6b 24**

Acknowledgment Number **a0 bf ca 34**

Data Offset **5**

ACK Flag set **10**

Window **3f f5**

Checksum **8f f3**

**TCPDUMP ADVANCED FILTERS FOR TCP HEADER FIELDS**

**----------------------------------------------**

**These filters are based on three parameters:**

**\* The offset where the field is located inside the TCP Header in bytes**

**\* The number of bytes the field uses on the TCP Header**

**\* The value assigned to the field**

**I.e.   
tcp[offset:bytes] = value**

**tcp[0:2] = TCP Source Port**

**tcp[2:2] = TCP Destination Port**

**tcp[4:4] = TCP Sequence Number**

**tcp[8:4] = TCP Acknowledgement Number**

**TCP Flags Filters (TCP Header offset 13)**

**\_ \_ \_ \_ \_ \_**

**U A P R S F  
32 16 8 4 2 1**

**tcp[13] = 0 No flags set**

**tcp[13] = 2 S 🡪 x0002**

**tcp[13] = 16 A 🡪 x0010**

**tcp[13] = 18 SA 🡪 x0012**

**tcp[13] = 24 PA 🡪 x0018**

**tcp[13] = 40 UP 🡪 x0028**

**tcp[14:2] = TCP Window**

**tcp[16:2] = TCP Checksum**

**tcp[18:2] = TCP Urgent Pointer**

**tcp[20:3] = TCP Options**

**UDP HEADER**

0 7 8 15 16 23 24 31

+--------+--------+--------+--------+

| Source | Destination |

| Port | Port |

+--------+--------+--------+--------+

| | |

| Length | Checksum |

+--------+--------+--------+--------+

|

| data octets ...

+------------------------------- ...

**First Notes about the UDP Header:**

1. **Every row is 32 bits long (4 bytes)**
2. **Every UDP Header field will be identified as a row division:**
   1. **Source Port**
   2. **Destination Port**
   3. **…**
3. **A row with two divisions like**

+--------+--------+--------+--------+

| Source | Destination |

| Port | Port |

+--------+--------+--------+--------+

**It has two 16 bit UDP Header fields.**

**4) The UDP Header is 8 bytes long total**

**UDP Header Fields:**

* **Source Port (16 bits. It is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed.)**
* **Destination Port (16 bits. Related to a particular internet destination address and service.)**
* **UDP Length (16 bits. It is the length in octets of this datagram including the header and the data. That means the minimum value of the length is eight.)**
* **UDP Checksum (16 bits. Like the TCP Checksum, this value is calculated based on a pseudo header of information from the IP header (SRC IP, DST IP, Protocol and UDP Length (Header + Data in bytes)), the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.)**

**UDP HEADER FORMAT DISSECTION**

0 7 8 15 16 23 24 31

+--------+--------+--------+--------+

| Source | Destination |

| Port | Port |

+--------+--------+--------+--------+

| | |

| Length | Checksum |

+--------+--------+--------+--------+

|

| data octets ...

+------------------------------- ...

**\*Ethernet Header** **\*IP Header**

**0000 00 07 ec b2 d0 0a 08 00 20 d1 76 19 08 00 45 00 .........v...E.**

**0010 00 70 b9 a2 40 00 ff 11 61 2d c0 a8 64 1c c0 a8 .p..@...a..d...**

**0020 64 9e 8d ec 02 02 00 5c ca d7 3c 32 38 3e 4e 6f d......\..<28>No**

**0030 76 20 32 39 20 30 39 3a 35 36 3a 31 38 20 69 6e v 29 09:56:18in**

**0040 65 74 64 5b 31 36 37 5d 3a 20 5b 49 44 20 38 35 etd[167]: [ID85**

**0050 38 30 31 31 20 64 61 65 6d 6f 6e 2e 77 61 72 6e 8011daemon.warn**

**0060 69 6e 67 5d 20 2f 75 73 72 2f 73 62 69 6e 2f 73 ing]/usr/sbin/s**

**0070 61 64 6d 69 6e 64 3a 20 4b 69 6c 6c 65 64 admind: Killed**

**Dissection:**

Source Port **8d ec**

Destination Port **02 02**

Length **00 5c**

Checksum **ca d7**

Syslog Header **3c 32 38 3e**

**TCPDUMP ADVANCED FILTERS FOR UDP HEADER FIELDS**

**----------------------------------------------**

**udp[0:2] = UDP Source Port**

**udp[2:2] = UDP Destination Port**

**udp[4:2] = UDP Message Length**

**udp[6:2] = UDP Checksum**

**ICMP ECHO / ECHO REPLY HEADER**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Type | Code | Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identifier | Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data ...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

**First Notes about the ICMP Echo Header:**

1. **Every row is 32 bits long (4 bytes)**
2. **Every ICMP Header field will be identified as a row division:**
   1. **Type**
   2. **Code**
   3. **Checksum**
   4. **…**
3. **A row with three divisions like**

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Type | Code | Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

**It has two 8 bit ICMP Header fields and one 16 bit one.**

1. **The ICMP Echo Header is 8 bytes long total**

**ICMP Echo Header Fields:**

* **Type (8 bits. 8 for echo message; 0 for echo reply message.)**
* **Code (8 bits. 0)**
* **Checksum (16 bits. ICMP Header Checksum. Recomputed at each end)**
* **Identifier (16 bits. Can be used to help match echo requests to the associated replies. It may be set to zero.)**
* **Sequence Number (16 bits. Can be used to help match echo requests to the associated replies. It may be set to zero.)**

**ICMP ECHO / ECHO REPLY HEADER FORMAT DISSECTION**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Type | Code | Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identifier | Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data ...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

**\*Ethernet Header** **\*IP Header**

**0000 08 00 20 d1 76 19 00 07 ec b2 d0 0a 08 00 45 00 ...v.........E.**

**0010 00 54 e5 ce 00 00 35 01 3a ad 40 0e 75 0a c0 a8 .T....5.:.@.u...**

**0020 64 1c 08 00 70 37 f8 4c b7 87 08 09 0a 0b 0c 0d d...p7.L........**

**0030 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d ................**

**0040 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d ..!"#$%&'()\*+,-**

**0050 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d ./0123456789:;<=**

**0060 3e 3f >?**

**Dissection:**

Type **08 <- Echo**

Code **00**

Checksum **70 37**

Identifier **f8 4c**

Sequence Number **b7 87**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Type | Code | Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identifier | Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data ...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

**\*Ethernet Header** **\*IP Header**

**0000 00 07 ec b2 d0 0a 08 00 20 d1 76 19 08 00 45 00 .........v...E.**

**0010 00 54 71 49 40 00 ff 01 a5 31 c0 a8 64 1c 40 0e .TqI@....1..d.@.**

**0020 75 0a 00 00 78 37 f8 4c b7 87 08 09 0a 0b 0c 0d u...x7.L........**

**0030 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d ................**

**0040 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d ..!"#$%&'()\*+,-**

**0050 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d ./0123456789:;<=**

**0060 3e 3f >?**

**Dissection:**

Type **00 <- Echo Reply**

Code **00**

Checksum **78 37**

Identifier **f8 4c**

Sequence Number **b7 87**

**TCPDUMP ADVANCED FILTERS FOR ICMP HEADER FIELDS**

**-----------------------------------------------**

**icmp[0]= icmp type**

**icmp[1] = icmp code**

**icmp[4:2] = identifier**

**icmp[6:2] = sequence number**

## Loki Backdoor Case Study

### Source of Trace

On the 7th July 2001, a Loki attack was revealed by an ISS Real Secure IDS system. The information as supplied by this system was:

Date 07/07/2001

Time 14:18:51

Detect name Loki

Source Address B.B1.142.62

Destination Address B.B2.159.218

tcpdump –r 14.15 –nX ’host B.B1.142.62’

14:18:51.507735 B.B1.142.62 > B.B2.159.218: icmp: request

0x0000 4500 02d8 3d17 0000 7b01 924b BBBB 8e3e E...=...{..KBB.>

0x0010 BBBB 9fda 0800 6a80 0100 **f001** a147 1842 BB....j......G.B

0x0020 2c49 3a4d 233f 5f51 337c cd4a b156 1e5d ,I:M#?\_Q3|.J.V.]

0x0030 fc57 9e57 f877 5e29 3e4d 1c2b f71f 3315 .W.W.w^)>M.+..3.

0x0040 877c 266a 620c 9322 c86e 7f0d 4458 6730 .|&jb..".n..DXg0

..................

0x02b0 fb2f c00b f172 130f 0042 f05d e94b cf5c ./...r...B.].K.\

0x02c0 1979 f658 2076 8c17 fc7a 4522 8e7d 7266 .y.X.v...zE".}rf

0x02d0 fb30 7221 c400 df2d cf6d e81e ce24 b943 .0r!...-.m...$.C

0x02e0 473f 78a5 G?x.

### Detect was generated by

The initial detect was generated by ISS Real Secure [[ISS](#ISS)], with correlating information from Tcpdump [[TCPDump](#TCPDump)].

### Probability the source address was spoofed

This packet is very unlikely to have been spoofed as both the source and destination addresses are internal to the corporate network.

### Description of attack

Loki is a backdoor trojan horse program that was published in Phrack 51, Article 6 [[Phra51§6](#Phra51)]. It uses ICMP echo request|reply packets as a carrier for the data payload, which is used to remotely control the infected computer. The ICMP header contains a sequence number (offset 0x001A in the Tcpdump output in section §1.1.1 above), which is set to 0x**f001** (or 0x01f0 in little-endian notation) for a Loki attack. It is therefore to be expected that, assuming a regular distribution of sequence numbers, two in every 65,536 ICMP packets will contain one of these sequence numbers, and therefore trigger a Loki false positive.

The event here was accepted as a false positive. ***The full dump of ICMP packets from Tcpdump showed four ping requests and replies, each containing a payload of 700 random bytes. These ping requests had consecutive sequence numbers (0xefe1, 0xeff1, 0xf001, 0xf011), and only one packet contained the ‘trigger’ sequence number.***

***In addition, the machine B.B1.142.62 was found to be a network monitoring machine, using large ICMP packets to test network resilience. If the target machine B.B2.159.218 had, indeed, been compromised then a larger number of ICMP packets would have been expected and they would all have had the same sequence number.***

### Attack mechanism

There is no attack in this incidence, but the ping probe was determined to be a heartbeat monitor coming from HP Openview [[HewlOV](#HewlOV)], a network monitoring tool.

### Correlations

The Real Secure alert was correlated to the Tcpdump output. No further correlations are indicated or required.

### Evidence of active targeting

There is no evidence of active targeting and this alert was accepted as a false positive.

## Hex to Decimal / Decimal to Hex Conversions

**Hex to decimal examples**

4096 256 16 1

163 162 161 160

1234 hex to decimal

*From right to left:*

160 X 4 + 161 X 3 + 162 X 2 + 163 X 1 = 4 + 48 + 512 + 4096 = **4660** **decimal**

FC96 hex to decimal

*From right to left:*

160 X 6 + 161 X 9 + 162 X 12 + 163 X 15 = 6 + 144 + 3072 + 61440 = **64662 decimal**

**Decimal to Hex examples**

*Using the Remainder Method: use remainders of repeated integer divides by 16 until quotient is zero. Convert remainders to hex (0-16 to 0-F) and line up in reverse order of rendering.*

13117 decimal to hex

13117 / 16 = 819 with remainder of 13 or D (least significant digit)

819 / 16 = 51 with remainder of 3

51 / 16 = 3 with remainder of 3

3 / 16 = 0 with remainder of 3 (most significant digit)

Result 🡪 **333D hexadecimal**

## Ethernet II Frame Format

+-----------+----------+----------+-----------+----------+

|Destination|Source MAC|Frame type|Data | FCS |

|MAC Address|Address |(IP, ARP) |(46 to | Checksum |

|(6 bytes) |(6 bytes) |(2 bytes) |1500 bytes)| (4 bytes)|

+-----------+----------+----------+-----------+----------+

**Ethernet II Frame Fields:**

* **Destination MAC Address**
* **Source MAC Address**
* **EtherType** 
  + **x0800 = IP**
  + **x0806 = ARP**
  + **x8100 = 802.1q or VLAN tagged**
  + **x8137 = Novell NetWare IPX/SPX**
* **Data 🡪 IP Packet Data**
* **FCS (Optional Trailer). The Frame Check Sequence is a cyclic redundancy check used by the sending and receiving stations to verify a successful transmission. The FCS is calculated on the contents of the Destination MAC address, Source MAC address, Type, and Data.**

**MAC Address (unique NIC identifier)**

* **6 bytes (48 bits) long. I.e. 00–07–ec-b2-d0-0a**
* **The first half is assigned to the NIC manufacturer. It is also called the organizational identifier or OI. If we look for   
  ”0007EC” on standards.ieee.org/regauth/oui/index.shtml, we are going to find that the NIC was manufactured by:**

**Cisco Systems Inc.**

**170 West Tasman Dr.**

**San Jose CA 95134**

**UNITED STATES**

* **The second half “b2-d0-0a” is a unique serial number assigned by the manufacturer.**

**Ethernet Frame Capture Dissection**

**0000 00 07 ec b2 d0 0a 08 00 20 d1 76 19 08 00 45 00 .........v...E.**

**0010 00 54 71 49 40 00 ff 01 a5 31 c0 a8 64 1c 40 0e .TqI@....1..d.@.**

**0020 75 0a 00 00 78 37 f8 4c b7 87 08 09 0a 0b 0c 0d u...x7.L........**

**00 07 ec b2 d0 0a Destination MAC address**

**08 00 20 d1 76 19 Source MAC address**

**08 00 Protocol being carried 🡪 IP**

**Basic Packet AnalysisTest**

0000 08 00 20 d1 76 19 00 07 ec b2 d0 0a 08 00 45 00

0010 00 55 6b 23 40 00 2c 06 9a db 3d db 5a b4 c0 a8

0020 64 1c dd 86 17 e0 7f 91 5f 2e ba 3d 11 2a 80 18

0030 16 d0 a4 df 00 00 01 01 08 0a 02 e4 34 22 06 c9

0040 7a a2 30 30 30 30 30 30 30 32 30 34 30 30 30 64

0050 30 30 30 31 20 20 34 20 00 72 6f 6f 74 00 00 31

0060 30 00 00

1. What is the "DST" Ethernet   
 address? \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_

2. What is the "SRC" Ethernet

Address? \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_

3. What is the type of protocol the Ethernet Frame is carrying? \_\_\_\_  
   
 What hex digits represent this protocol? \_\_ \_\_ \_\_ \_\_

4. What hex digits are in the first byte of the IP header? \_\_ \_\_

5. What hex digits are in the last byte of the IP header? \_\_ \_\_

6. What is the IP version number? \_\_

7. In hex, what is the "DST" IP address. \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_

8. In hex, what is the "SRC" IP address. \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_

9. How many bytes long is the IP header. \_\_\_\_\_

10. How many bytes long is the Total Packet Length \_\_\_\_\_

11. How many bytes long is the IP data   
 (without counting the IP header). \_\_\_\_\_

12. What hex digits are in the first two bytes of the TCP header?   
 \_\_ \_\_ \_\_ \_\_

13. How many bytes long is the TCP header (consider no  
 TCP options). \_\_\_\_\_

14. How many bytes long is the TCP data if the TCP Header Size  
 is increased to 32 (using 12 bytes of TCP Options). \_\_\_\_\_

15. In hex, what is the size of the TCP window? \_\_ \_\_ \_\_ \_\_

In decimal, what would be its value? \_\_\_\_\_

17. In hex, what is the TCP SRC port. \_\_ \_\_ \_\_ \_\_

In decimal, what would be its value? \_\_\_\_\_

18. What hex Ethernet code what assigned to Cisco?

(The company that made the SRC Device’s NIC). \_\_ \_\_ \_\_ \_\_ \_\_ \_\_

IP Packet - An IP header contains the following fields in the following order.

1. Version number 8. Fragment offset

2. Header length 9. Time to live

3. Type of service 10. Protocol

4. Total length 11. Header checksum

5. Identifier 12. Source address

6. More flag 13. Destination address

7. Don't fragment flag 14. Options, if any

What is the length, in bits, of each of the following fields in an IP packet?

1. Version number \_\_\_\_ 8. Fragment offset \_\_\_\_

2. Header length \_\_\_\_ 9. Time to live \_\_\_\_

3. Type of service \_\_\_\_ 10. Protocol \_\_\_\_

4. Total length \_\_\_\_ 11. Header checksum \_\_\_\_

5. Identifier \_\_\_\_ 12. Source address \_\_\_\_

The TCP header consists of the following fields in the following order.

1. source port 8. PUSH flag

2. destination port 9. RST flag

3. sequence number 10. SYN flag

4. acknowledgement number 11. FIN flag

5. data offset 12. window

6. Urgent flag 13. checksum

7. ACK flag 14. urgent pointer

What is the length, in bits, of each of the following fields in a TCP packet?

1. the source port \_\_\_\_ 5. the data offset \_\_\_\_

2. the destination port \_\_\_\_ 12. the window \_\_\_\_

3. the sequence number \_\_\_\_ 13. the checksum \_\_\_\_

4. the ack number \_\_\_\_ 14. the urgent pointer \_\_\_\_

\* Hint: data offset indicates the offset where the data begins and it is equivalent to the TCP Header size.

**References:**

RFC 793 - Transmission Control Protocol

<http://www.faqs.org/rfcs/rfc793.html>

TCP, Transmission Control Protocol

http://www.networksorcery.com/enp/protocol/tcp.htm

RFC 768 - User Datagram Protocol

http://www.faqs.org/rfcs/rfc768.html

UDP, User Datagram Protocol

<http://www.networksorcery.com/enp/protocol/udp.htm>

RFC 792 - Internet Control Message Protocol  
<http://www.faqs.org/rfcs/rfc792.html>

ICMP type 8, Echo request message

<http://www.networksorcery.com/enp/protocol/icmp/msg8.htm>

Tech Info - IP Message Formats

<http://www.zytrax.com/tech/protocols/tcp.html#tcp>

Analyzing The TCP Header  
http://www.firewall.cxi/index.php

Hex Conversions

http://www.htservices.com/Reference/HexadecimalConversions/

<http://www.stcsc.edu/OCL/hex_to_decimal_conversion.htm>

GCIA Practical Assignment

Robert Turner Practical

Ethernet Frame Format

<http://www.geocities.com/SiliconValley/Haven/4824/ethernet.html>

<http://www.zyxel.com/support/supportnote/ies1000/app/8021q.htm>

<http://sd.wareonearth.com/~phil/net/overhead/>