Global Information Assurance Certification Paper

HL7 Data Interfaces in Medical Environments: Attacking and Defending the Achille's Heel of Healthcare

GIAC (GCIA) Gold Certification

Author: Dallas Haselhorst
Email: dallas@treetopsecurity.com / Twitter: @oneoffdallas
Advisor: Sally Vandeven
Accepted: August 2017

Abstract

On any given day, a hospital operating room can be chaotic. The atmosphere can make one's head spin with split-second decisions. In the same hospital environment, medical data also whizzes around, albeit virtually. Beyond the headlines involving medical device insecurities and hospital breaches, healthcare communication standards are equally as insecure. This fundamental design flaw places patient data at risk in nearly every hospital worldwide. Without protections in place, a hospital visit today could become a patient's worst nightmare tomorrow. Could an attacker collect the data and sell it to the highest bidder for credit card or tax fraud? Or perhaps they have far more malicious plans such as causing bodily harm? Regardless of their intentions, healthcare data is under attack and it is highly vulnerable. This research focuses on attacking and defending HL7, the unencrypted and unverified data standard used in healthcare for nearly all system-to-system communications.

1. Introduction

Healthcare security is years behind all other critical industries – finances, retail, and utilities. However, healthcare contains some of the most beneficial data for attackers. Electronic health records (EHR) are loaded with protected health information (PHI) and personally identifiable information (PII) necessary for a broad range of illegal endeavors. As such, "Stolen EHR can be used to acquire prescription drugs, receive medical care, falsify insurance claims, file fraudulent tax returns, open credit accounts, obtain official government-issued documents such as passports, driver's licenses, and even create new identities" (Fuentes, 2017). The potential criminal activities are nearly endless using high-quality medical records.

When a credit card gets stolen, the card issuer sends a new card with few, if any, lasting consequences. In healthcare and associated data, a new social security number is not possible except in extreme situations. Those affected by a PHI breach receive no extra protection for their highly personal, exfiltrated healthcare data. While handled similarly post-breach, the uses for stolen healthcare data are far greater than standard financial data. As stated by the Health Care Industry Cybersecurity Task Force, "The identity protection is only a help for credit-based identity theft, it does not provide the patient with adequate protections based on the sensitivity, value, and permanence of their health care data, which is priceless… Someone could steal a teenager's medical history today, only for it to become valuable when the individual achieves a prominent role in public life." (Health Care Industry Cybersecurity Task Force, 2017). Fortunately, we may never see this Hollywoodesque, worst-case scenario. Regardless, one cannot argue the potential threats associated with a stolen EHR.

Despite its ever-increasing value, an overwhelming amount of healthcare data is mishandled every day, if not every second, due to a fundamental communications flaw. HL7 [or Health Level 7] is a relatively obscure standard that provides most of the system-to-system communications. It allows disparate systems in a hospital environment to speak a similar language. In many ways, HL7 is the glue that allows various hospital systems to interoperate. For that reason, it resides in nearly every corner of the healthcare industry.

Because HL7 works mostly behind-the-scenes, it often goes unnoticed when analyzing risks in healthcare IT.

As part of the HL7 standard, different messages and message types are sent for almost every event one might encounter during a hospital visit; each event triggers different messages and message types. When a patient moves to a different room, an HL7 ADT (admissions, discharge, and transfer) message goes from one system to another system that needs the updated information. When someone orders a test or drug, an ORM (order) message is sent and received. When sending test results or telemetry data, an ORU (observation result) message is expected to traverse the network. In most cases, a single HL7 message will be relayed to an interface engine to assist with the distribution of messages. The interface engine can transform the message and forward it to numerous outlying systems to help the data stay in sync (Figure 1).

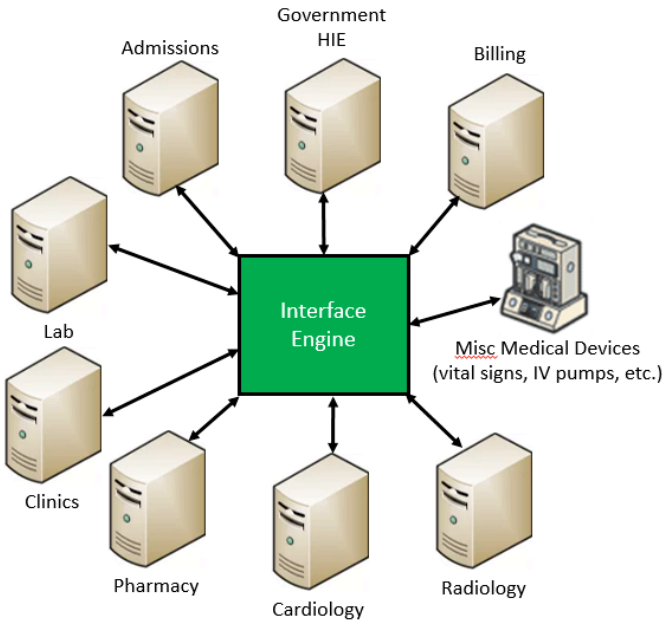

Figure 1: HL7 Interfaces Connecting Various Hospital Systems

As expected, HL7 messages can carry some of the most sensitive data in a hospital. In the ADT example message and corresponding table found in Figure 2 below, an HL7 message will include complete patient information including birthdays, current and past addresses, phone numbers, and social security numbers. It is also common to find other information such as the patient's relatives, email addresses, place of birth,

mother's maiden name, etc. This additional information is important to recognize because it is the type of information often used for secondary security questions when someone sets up an online account, e.g., a bank account, a new email account, etc. The valuable data found in EHRs is one of the many reasons healthcare is so attractive to cyber criminals.

```
MSH|^~\&|SENDING_APPLICATION|SENDING_FACILITY|RECEIVING_APPLICATION|RECEIVING_F
ACILITY|20170613083617||ADT^A01|911576160110613083617|P|2.3||||
EVN|A01|20170613083617|||
PID|1||135769||MOUSE^MICKEY^||19281118|M|||123 Main St.^^Lake Buena Vista^FL^
32830||(407)939-5555^^^ohtoodles@notdisney.com|||||1719|999999999|||
|MOUSETOWN||||||||||||||||
NK1|1|MOUSE^MINNIE|WIFE||||||NK
PV1|1|O|||||^^^^^^^^|^^^^^^^^
AL1|1||^Penicillin||Anaphylactic shock
AL1|2||^Cat dander|Skin rash
```

Identifier	Field Definition	Field Contents
	Patient Name	MOUSE^MICKEY
	Date/Time of Birth	19281118
	Patient Address	123 Main St.^^Lake Buena Vista^FL^32830
	Phone Number – Home	(407)939-5555
	Email Address	ohtoodles@notdisney.com
	SSN Number	999999999
	Birth Place	MOUSETOWN
	Next of Kin	MOUSE^MINNIE
	Allergies / Reactions	Penicillin / Anaphylactic Shock Cat Dander / Skin Rash

Figure 2: Sample HL7 ADT Message & Corresponding Field Definitions

D

Overall, HL7 helps healthcare function much more efficiently. It eliminates large amounts of data entry and it provides nearly instantaneous distribution of lab results, orders, and billing. Astonishingly, HL7 is most often passed system-to-system completely unencrypted. The organization that produces and guides the standard, HL7 International, has a brief discussion on encryption to show they were not oblivious to it. In their wiki, they state "In the Security TC we have assumed that encryption happens below the application layer, e.g., via IPSec or TLS, not within HL7 messages" (HL7 International, 2007). Essentially, despite the sensitivity of the data, HL7 does not require or even offer encryption, placing the sought-after patient information at risk.

The most glaring issue with HL7 is due to its clear-text communications, but other security concerns exist as well. The HL7 standard also lacks authentication and by default, any system can communicate with an HL7 receiving port. For the trained security professional, native HL7 communications are similar to the security woes found in Telnet and FTP. Still, those insecure protocols still have authentication mechanisms built-in. Another potential oversight and significant factor to understand regarding how interfaces operate is the many-to-one communication relationship. Interfaces work much like a web server or any other client-server application, allowing multiple clients to connect to the same port on a server at any given time. Not unlike a web server, numerous unauthenticated clients can communicate with an open interface port.

Messages sent and received also have no method of verification. While HL7 does have acknowledgments, they are not required and the acknowledgments often only specify whether a message is received. An acknowledgment does nothing to identify whether the data is valid or reasonable. If any data verification checks exist, it is the responsibility of the receiving system. This methodology is inherently flawed as the receiving system lacks complete visibility and does not recognize if some messages are not received.

For a more in-depth discussion on the HL7 standard, please reference "HL7 Data Interfaces in Medical Environments: Understanding the Fundamental Flaw in Healthcare" at the website, https://www.linuxincluded.com/hl7-medical-fundamental-flaw/. That HL7 research paper covers the value of stolen medical data, types of HL7

messages, and even security concerns with the HL7 replacement, FHIR. It is also available for download in the SANS Reading Room.

2. Attacking HL7

2.1. Denial of Service (DoS)

While many criminals search for ways to monetize their endeavors on the black market or dark web, others thrive on chaos. Though service disruption attacks often appear to lack motive, this type of activity has surfaced in several cyberattacks over the years. Recently, it played a role in the Petya.2017 "fakesomware" campaign. The "not" ransomware was discovered by Comae Technologies and Kaspersky Lab to be a wiper because the attacker did not appear to have a way to recover the encrypted files (Suiche, 2017). It is possible the attack was a simple misconfiguration or a distraction. Whether an exact motive exists or is known, these types of attacks will persist and reaffirm the notion that "some men just want to watch the world burn."

HL7 interfaces are susceptible to denial of service attacks. In a standard HL7 interface, two systems communicate with one another on an ad-hoc basis. When there are no messages in the process of sending, the transmitting system will temporarily disconnect. As shown in Figure 3, the interface on the listening/receiving side will switch to "idle." The "idle" state indicates the system is not receiving or polling for new messages and it is ready for the next connection and subsequent message.

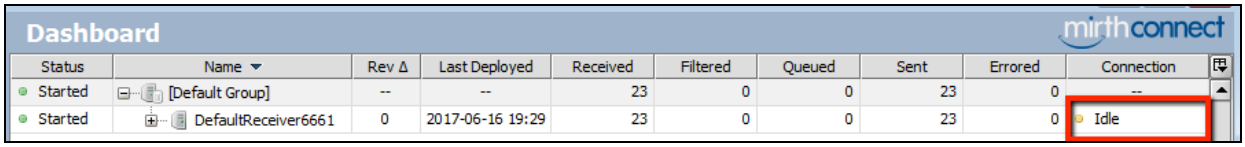

Figure 3: An Idle Connection in the Popular Interface Engine, Mirth

Figure 4 depicts a sample TCP listener configuration along with many of the default configuration options. As highlighted by the red box, the maximum number of listening connections is ten by default. Keep in mind that HL7 does not utilize authentication so any system can connect to the HL7 listener. Even a simple Telnet command will connect to an open interface port. Also, note that by default in this interface engine, the "receive timeout" is disabled and the "keep connection open" option

is set to "yes." Consequently, the connection will remain open until the initiator decides to close it regardless of whether traffic is sent or received. For a standard interface, this serves as neither an advantage or disadvantage and depends on the operational needs of the interface.

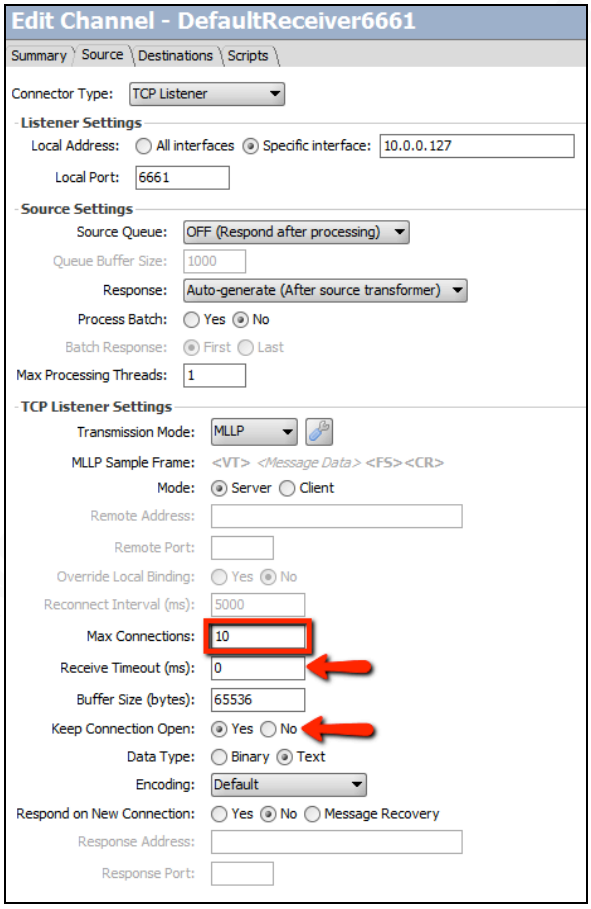

Figure 4: TCP Listener Configuration Options

An attacker could launch ten Telnet clients from a command line causing a denial of service. This action would prevent the receiving interface from processing HL7 messages sent by the actual sending interface system. A simple one-line 'for' loop from a Linux bash shell (Figure 5) would achieve the same result spawning ten connections almost instantaneously. The same attack could be performed natively with Linux TCP sockets from the command line rather than using Netcat as well. Regardless of the DoS method used, maxing the number of open connections results in blocking legitimate communications as shown in Figure 6.

```
for((conn=1;conn<=10;conn++)); do nc 10.0.0.127 6661 & done
```

Figure 5: DoS Attack Using Linux Command-Line & Netcat

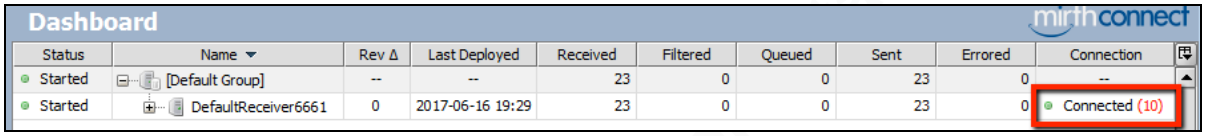

Figure 6: DoS Attack on an HL7 Interface

2.2. Unauthorized Senders

A basic HL7 interface consists of a single system sending messages and another system receiving messages. These systems most often communicate via MLLP (minimum lower layer protocol) encapsulated in TCP/IP packets. As shown in the DoS example, an attacker only needs the IP address and TCP port to cause havoc. With that same knowledge and a familiarity with the HL7 message format of the receiving system, an attacker could also cause damage by sending additional, unauthorized messages.

In a standard interface configuration, nothing prevents an attacker from sending messages to a receiving application. Errant messages could cause numerous problems and confusion, significantly impacting patient care. Given HL7 is nothing more than delimited text, a simple text editor can make any change imaginable. As seen in Figure 7, with a few keystrokes on the HL7 allergy segment, a severe allergic reaction to penicillin is modified to "no known allergies" instead. The ADT message also changes to an A08 (patient information update) message from an A01 (patient admission).

```
1  MSH|^~\&|SENDING_APPLICATION|SENDING_FAC
2  EVN|A01|20170613083617|||
3  PID|1||135769||MOUSE^MICKEY^||19281118|M
4  NK1|1|MOUSE^MINNIE|WIFE||||||NK
5  PV1|1|O|||||^^^^^^^^|^^^^^^^^
6  AL1|1||^Penicillin||Anaphylactic shock
7  AL1|2||^Cat dander|Skin rash
```

```
1  MSH|^~\&|SENDING_APPLICATION|SENDING_FACI
2  EVN|A08|20170613083617|||
3  PID|1||135769||MOUSE^MICKEY^||19281118|M|
4  NK1|1|MOUSE^MINNIE|WIFE||||||NK
5  PV1|1|O|||||^^^^^^^^|^^^^^^^^
6  AL1|1||^No Known Allergies||
7
```

Figure 7: HL7 Message Changes in Text Editor
(Original Message on the Left)

Once the modifications are complete, the attacker requires a way to send the fraudulent HL7 message. Fortunately for the attacker, there are a handful of freeware

tools to perform this action. These software packages also properly accept and respond to HL7 ACK (acknowledgment) messages with no additional configuration necessary. This functionality is no fault of the tools themselves; this maleficence is leveraging capabilities required for testing HL7 interfaces. A packet crafting/manipulation application such as Scapy could also act as a messaging tool for an attacker with knowledge of HL7. In Figure 8, the HL7 messaging application sends a modified message to the specified IP address and port.

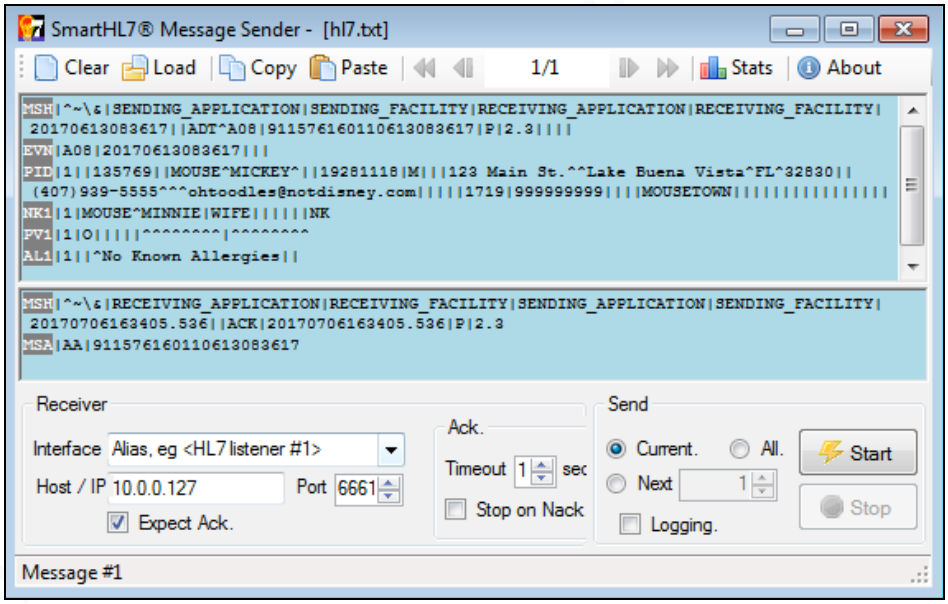

Figure 8: Modified HL7 Message Sent from a New Host

Even though the HL7 message originates from a new IP address, the message processes as normal and the receiver sends an ACK back to the originator. Using chaos as the sole motive, an attacker could send discharge messages (ADT-A03) for every patient in a hospital. The mass chaos could act as a distraction for another, more profitable criminal endeavor. Erroneous update messages (ADT-A08) could be sent to remove drug allergy information. A fraudulent order (ORM message) for the known drug allergy might follow the ADT-A08 message. It is not difficult to understand the potentially fatal outcome of such an attack.

2.3. Man-in-the-Middle (MITM) & ARP Spoofing

A discussion on the insecurities of clear-text protocols and communication is not complete without an understanding of man-in-the-middle (MITM) attacks. HL7 is highly

susceptible to MITM because the standard does not address the two means to combat it – authentication (prevention) and tampering detection. Without either of these controls, an attacker can easily intercept the communications between two systems. The captured data could be sold immediately on the dark web or it could be stored for future mischief as discussed by the Health Care Cybersecurity Task Force. MITM also provides ample opportunities for an attacker to make changes to HL7 messages in real-time.

While numerous MITM techniques exist, the most common and likeliest candidate for a local network attack is ARP (address resolution protocol) spoofing, also known as ARP poisoning. This attack would work on the sending system, the receiving system, or the interface engine. As with any MITM technique, the goal of ARP spoofing is for the attacker to intercept traffic between endpoints. The endpoints can be between any two networked devices – computer to server, computer to firewall, server to server, etc. By sending gratuitous ARP packets to a targeted system, an attacker can re-associate their own MAC address with the legitimate IP addresses of another system. The traffic then routes to the attacker instead of the secondary system. If the attacking system forwards the traffic, neither system is aware their packets are being intercepted and instead "believe" they are communicating directly with one another. Figure 9 shows an ARP spoofing attack. This graphic also accurately illustrates the IP and MAC addresses used in the technical examples throughout this research.

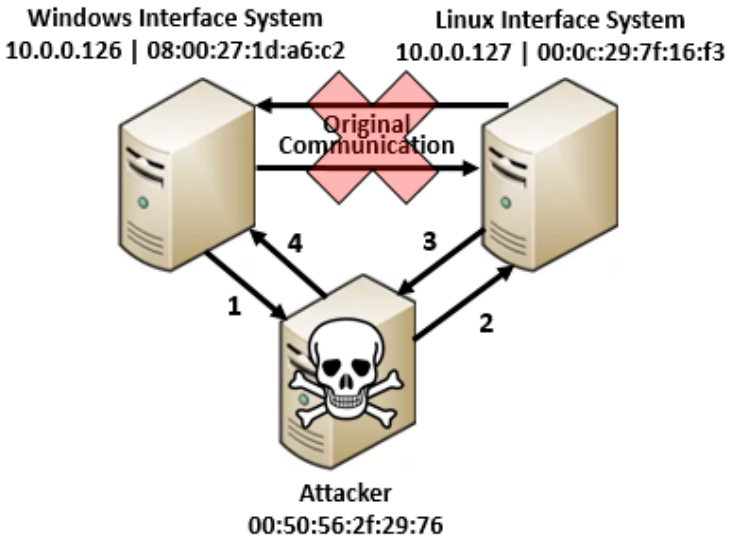

Figure 9: ARP Spoofing Example With IP & MAC Addresses

ARP spoofing is easy with the help of penetration testing distributions such as Kali Linux. In fact, there are multiple tools installed by default to assist in performing ARP spoofing. If an attacker is familiar with the command line, the "echo 1" command in Figure 10 enables IP forwarding. The second line in Figure 10 is the arpspoof command, which specifies the two target IP addresses. In the example, the "-t" specifies the target IP address and the "-r" is used to enable bi-directional poisoning (host and target spoofing in one command). The arpspoof command will continue sending packets periodically until the operator terminates the process. A gratuitous ARP every few seconds is more than sufficient to maintain a constant MITM presence since most systems have a minimum ARP cache timeout of a few minutes. Depending on the manufacturer, some routers default to several hours for ARP cache timeouts. Figure 11 shows the ARP cache from the affected systems pre- and post-ARP poisoning. Once the ARP poisoning attack completes, the associated MAC address for the neighboring IP address is now that of the attacker (00:50:56:2F:29:76).

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# arpspoof -r 10.0.0.126 -t 10.0.0.127
0:50:56:2f:29:76 0:c:29:7f:16:f3 0806 42: arp reply 10.0.0.126 is-at
0:50:56:2f:29:76
0:50:56:2f:29:76 8:0:27:1d:a6:c2 0806 42: arp reply 10.0.0.127 is-at
0:50:56:2f:29:76
…
```

Figure 10: IP Traffic Forwarding & arpspoof Command

```
Windows system ARP entries - pre-ARP poisoning
C:\>arp -a
  Internet Address      Physical Address      Type
  10.0.0.127            00-0c-29-7f-16-f3     dynamic
Windows system ARP entries - post-ARP poisoning
C:\>arp -a
  Internet Address      Physical Address      Type
  10.0.0.127            00-50-56-2f-29-76     dynamic
Linux system ARP entries - pre-ARP poisoning
# arp -a
? (10.0.0.126) at 08:00:27:1d:a6:c2 [ether]
Linux system ARP entries - post-ARP poisoning
```

```
# arp -a
? (10.0.0.126) at 00:50:56:2f:29:76 [ether]
```

Figure 11: MAC Addresses Pre- and Post-ARP Poisoning

Once all traffic between the two systems is being intercepted and forwarded by the attacking system, the attacker can run another command to collect the data. A common networking tool, tcpdump, can be used for this part of the process. In the tcpdump example found in Figure 12, the "host" and "port" are used as capture filters to narrow down the network packets captured as both the IP address and HL7 TCP port are known. The "-w" parameter will store captured traffic to a filename of choice (hl7_capture.pcap). Since tcpdump follows the well-known packet capture standard, other tools such as tshark or Wireshark (Figure 13) can view the data as well.

```
# tcpdump host 10.0.0.126 and port 6661 -w hl7_capture.pcap
```

Figure 12: tcpdump Capturing Interface Traffic

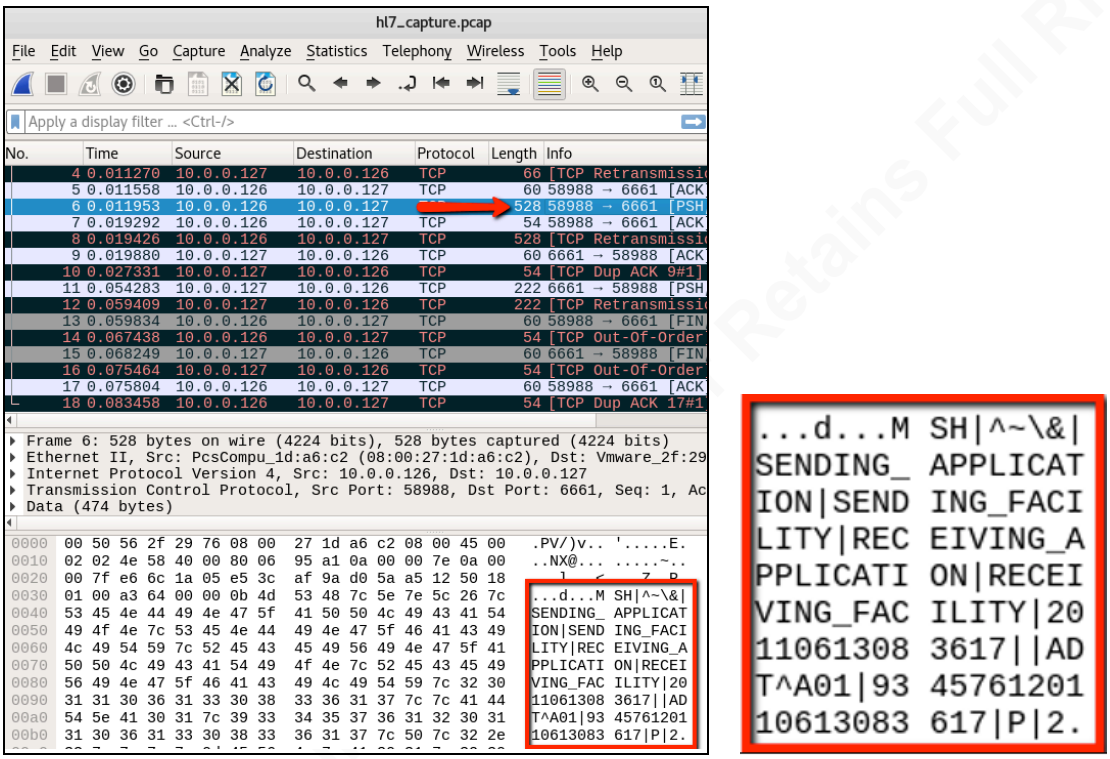

Figure 13: Using Wireshark to Read an HL7 Message Captured During an ARP Spoofing Attack

Other tools can perform these same tasks in a graphical environment. Ettercap GUI provides many of the capabilities discussed above in a single, easy-to-use application. It also simplifies the ARP spoofing portion of the attack point-and-click (Figure 14) while providing benefits such as connection data (Figure 15). In the connection data window, Ettercap improves the legibility of the HL7 message exchange by identifying the original message [on the left] and its corresponding ACK [on the right] (Figure 16). This feature of Ettercap shows relationship context between the two transmissions similar to the "follow stream" functionality in Wireshark.

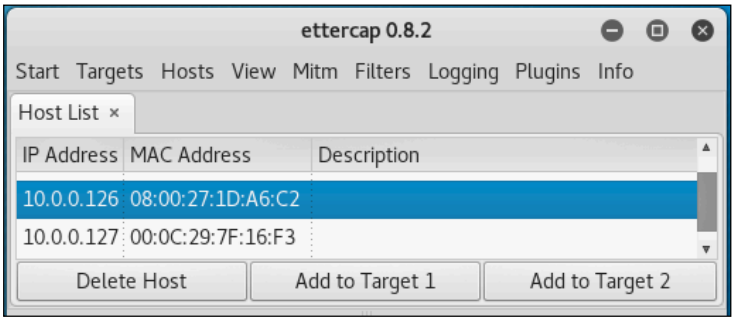

Figure 14: Ettercap - Adding Targets

Dallas Haselhorst

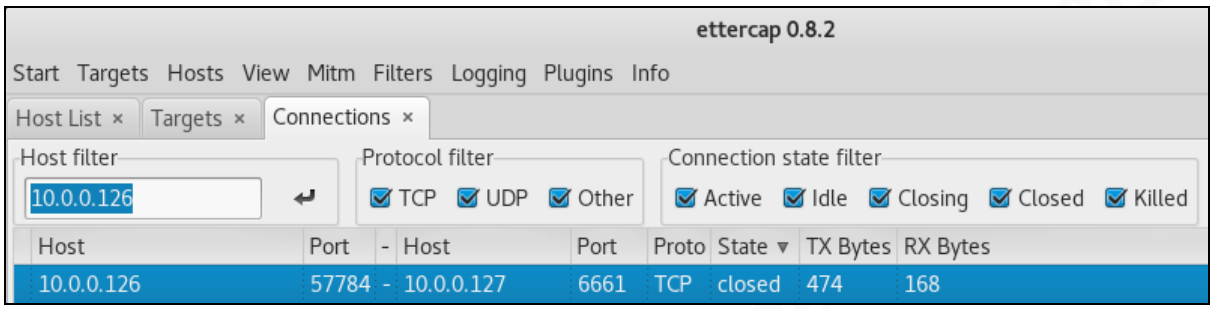

Figure 15: Ettercap - Connection Information

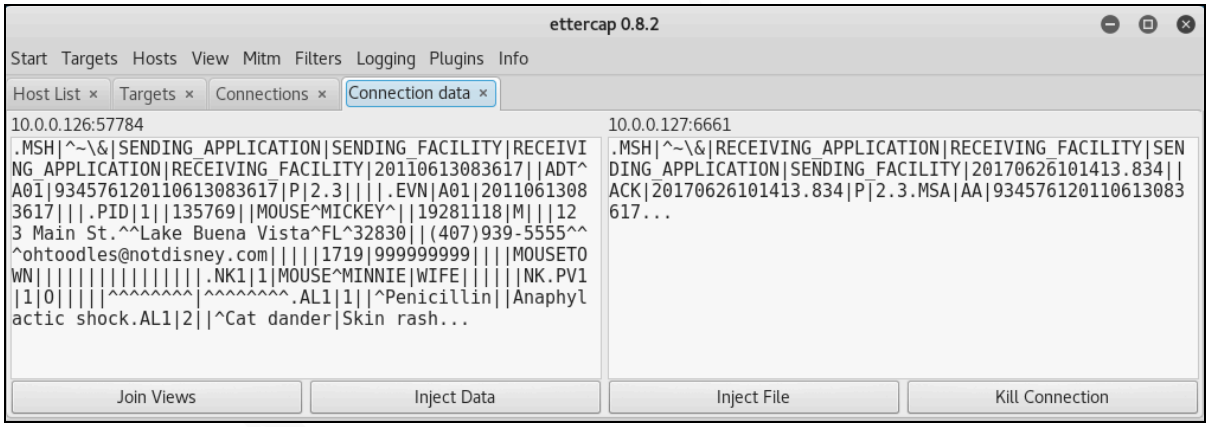

Figure 16: Ettercap - HL7 Message (Left) & ACK (Right)

An attacker familiar with the environment or a receiving system/interface may already understand the various nuances of the HL7 interface implementation. If not, general information to guide an attacker's research is often readily available on vendor websites under "HL7 interface specifications." If an attacker does not understand the intricacies of the receiving system or have access to vendor specifications, he or she could setup a MITM attack. Using MITM, the attacker could capture the messages, make modifications, and send errant messages as shown previously. Altered messages would ensure a much higher level of success than attempting to mint new messages from scratch.

Once the attacker establishes a MITM foothold, real-time data manipulation is also possible. Numerous tools exist for this type of packet altering, but the previously mentioned Ettercap is well-suited for this task too. A basic Ettercap filter such as the one in Figure 17 can change the text within HL7 messages on-the-fly. In the example, a mild case of hives linked to sushi replaces the life-threatening penicillin allergy causing

anaphylactic shock. This change is limited to any HL7 message containing the word MOUSE, which happens to be the last name of the patient. Using the Ettercap filter, the before and after HL7 messages with highlighted differences are in Figure 18.

```
# only apply filter to TCP data on port 6661 (receiving port)
if (ip.proto == TCP && tcp.dst == 6661) {
        # only modify HL7 messages with MOUSE in them
        if (search(DATA.data, "MOUSE")) {
        replace("Penicillin","Sushi"); # replace allergy
        replace("Anaphylactic shock","Hives"); # replace allergy reaction
        msg("Modified HL7 message!\n"); # send message to user
        }
}
```

Figure 17: Ettercap Filter to Replace HL7 Text (With Comments)

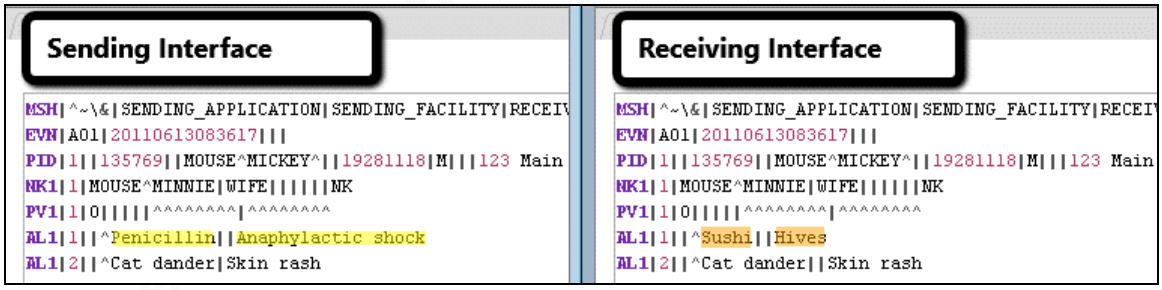

Figure 18: HL7 Message Modified in Transit Using Ettercap Filter

Aside from modifying allergies and sending potentially fatal drug orders, there are numerous other disturbing possibilities. There is the possibility of altering a patient's weight via an ADT message in advance of determining a weight-based drug prescription dosage. Other unfortunate outcomes might involve a crash cart showing up in the wrong room because of swapped patient telemetry data. Or perhaps flawed test results for a cancer patient come back clear. Who would be at fault? Someone would need to shoulder the blame and it is doubtful whether anyone would properly identify the errant messages. Consequently, the doctor might be sued for malpractice because his or her flawed diagnosis caused the patient mental anguish. The "creativity" of an attacker is the only limiting factor associated with the potential dangers of HL7 message tampering.

3. Defending HL7

As important as it is to discuss why an attacker might target EHRs and even the attack vectors they might use, it would be a disservice to ignore ways to protect interfaces from these attacks. Defending against these attacks requires some analysis of the sensitive medical data itself. Understanding the "what, where, and how" go a long way to mitigating risk with healthcare data (Tarala & Tarala, 2015). While the what is generally understood, where it is located or stored and how to secure it can get murky. This uncertainty is profoundly accurate for HL7. IT departments might realize attackers are targeting medical data so servers and healthcare applications themselves receive additional attention. However, very few recognize the hidden, criminal possibilities for stolen PHI and PII contained in HL7 messages. Even fewer healthcare organizations have sufficient defense measures configured for their otherwise vulnerable interfaces.

If an interface analyst and hospital management were interested in a more secure alternative to HL7, there is a possibility the vendor does not support it. The other concern is whether the new solution provides legitimate security improvements. Large HL7 interfaces were frequently multi-month projects requiring extensive configuration. Aside from their complicated setup and testing, interfaces often involve assistance from numerous departments as well as vendors. To further compound the issue, there is a high probability the original designers are no longer with the organization. Given these concerns, it is doubtful an organization would re-architect an interface solution from the ground-up solely for the sake of security. But what can be done to improve the security of existing HL7 interfaces without altering the complex, underlying components of the configuration? There are numerous ways to work around the insecurities of HL7 with advantages and disadvantages to each method.

3.1. Manual Validation

Though often forgotten in a high-tech world, manual validation can provide some level of defense. In its most basic form, manual validation is the act of checking whether the numbers on system one match those found on system two. It is an important discussion as it tests data validity outside of the insecure HL7 process. On a billing interface, the check might verify if the totals match. This form of checks and balances

would uncover deleted charges with relative ease. Similarly, someone could confirm the number of admissions, discharges, and transfer events match on both sides. While manual validation is a staple for billing processes, it unfortunately does not work for any other interface type. An interface engine will often discard certain message types because a receiving system cannot handle them. Thus, while system one would send an HL7 message, the interface engine may remove it entirely so system two never sees the message. Aside from limited usage based on interface types, manual validation does not scale well. Finally, errant or modified messages are still possible because manual validation cannot verify if the HL7 messages or totals are valid.

3.2. Host-based Firewall

A host-based firewall could provide some level of protection for HL7 interfaces. Most notably, it could prevent unauthorized systems from communicating with interfaces and sending HL7 messages. It would also prevent random systems from performing DoS attacks via the maximum port connections technique. Unfortunately, a host-based firewall provides zero protection for the otherwise clear-text data. A host-based firewall also does not protect against ARP spoofing attacks. Blocking all ARP traffic via a firewall rule would be ill-advised since ARP helps devices communicate. IP-specific firewall rules would also fail to protect since the MAC address of the attacking system would re-associate with the IP address rule; the firewall rule would allow traffic to pass because it would still match on the IP address despite the incorrect, new MAC address. One of the few benefits of this approach is that the need for vendor involvement might not be necessary, but it falls short otherwise.

3.3. Network Segmentation

Network segmentation can limit the visibility of the interfaces (and interface engine) from less sensitive networks. Segmenting user-facing systems from systems handling sensitive data can prove extremely advantageous since user systems are often the biggest threat due to phishing campaigns (Zurier, 2016). For this reason alone, network segmentation is an excellent defense strategy for any sensitive data and HL7 is no different. The primary issue with this approach lies in the role of an interface engine. Even in a well-segmented environment, an interface engine by nature must communicate

with numerous systems residing on different networks. Also, many interfaces were configured before widespread use of security best practices such as network segmentation. Network segmentation is difficult if it requires an IP scheme change for a receiving interface and all the systems connecting to it. While network segmentation will inevitably improve security, it will not mitigate the risk entirely. If an attacker could gain access to a network switch, the HL7 data would still transmit as clear-text.

3.4. ARP Spoofing Defenses

Network administrators are quick to dismiss static ARP entries as having no security benefits. While static ARP entries do provide speed improvements and can cut down on network noise, they can also improve security. A static ARP entry creates a permanent MAC and IP address association in the ARP cache so it cannot get changed with gratuitous ARPs. On Windows, a static ARP entry receives a "static" designation in place of "dynamic" under the type column as shown in Figure 19. On Linux, a "PERM" tag identifies a static ARP entry. Regardless of the underlying operating system, static ARP entries successfully prevent ARP spoofing attacks as expected.

```
Create a static ARP entry in Windows (from an administrator prompt)
C:\>netsh interface ipv4 add neighbor "Local Area Connection"
10.0.0.127 00-0c-29-7f-16-f3 store=persistent
Windows system ARP entries - post-static ARP entry
C:\>arp -a
  Internet Address      Physical Address      Type
  10.0.0.127            00-0c-29-7f-16-f3     static
Create a static ARP entry in Linux (as root or using sudo)
# arp -s 10.0.0.126 08:00:27:1d:a6:c2
Linux system ARP entries - post-static ARP entry
# arp -a
? (10.0.0.126) at 08:00:27:1d:a6:c2 [ether] PERM
```

Figure 19: Creating & Verifying Static ARP Entries

If an administrator cannot modify the OS, networking equipment such as a switch or router could also perform this defense-based role. Similar to the shortcomings of network segmentation, static ARP entries do nothing to protect or encrypt the data. An attacker could still collect HL7 information by gaining access to an intermediate networking device. Another possible approach to protecting a network against ARP

spoofing is through *detection* rather than *prevention*. Tools such as arpwatch can monitor and alert on MAC and IP address pairing changes, which would alert when malicious ARP traffic is detected.

3.5. Virtual Private Networks (VPN)

If HL7 messages receive protection, a VPN is the most common means. Based on first-hand observation alone, interfaces only utilize VPN solutions when the data is sent over the Internet or shared with a different location, i.e., not for internal communications. Even then, a site-to-site VPN provides encryption between the tunnel endpoints, but it does nothing to protect the data *befor*e reaching the point of encryption. One cannot overlook the security of the HL7 data pre-encryption when analyzing the risks associated with handling insecure data. The combination of proper network segmentation and a site-to-site IPSEC tunnel can offer adequate defenses, although a thorough understanding of the environment is necessary to confirm limited exposure.

Most organizations choose to deploy router and firewall-based site-to-site tunnels even though host-to-host VPN options are available. A host-based tunnel and proper configuration would eliminate any concerns regarding data in transit. However, host-based VPN tunnels can be difficult to configure. A host-to-host VPN solution in any form would potentially require some level of assistance from the medical software vendor. If the software vendor also manages the OS where the software resides, their involvement is almost guaranteed. Aside from configuration and IP address changes at the OS-level, both ends of the VPN would need configuration along with changes on the interface itself.

3.6. SSH Tunneling

VPN tunnels and SSH tunneling are the only two defense methodologies discussed here that protect HL7 data by encrypting the data. Due to added benefit of encryption alone, VPNs and SSH tunneling are the most thorough options to protect existing HL7 interfaces. An SSH tunnel is often easier to configure than a VPN and the software is commonly available for free. Amazingly, utilizing SSH tunneling is not frequently discussed as an option to protect HL7 data. SSH tunneling is little more than a passing mention in knowledge base articles (iNTERFACEWARE, 2013) or tucked away

in vendor documentation as an alternative to VPN tunnels (athenahealth, 2016). This lack of attention given to SSH tunneling might result from the overall lack of interest in securing HL7.

Figure 20 depicts how an HL7 connection over SSH works (lower boxes) in comparison to a standard HL7 connection (upper boxes). Instead of the direct connection via port 6661 found in the standard connection, the SSH tunnel sends all traffic destined for port 6661 to port 22. System 1 encrypts the data before it leaves the host and System 2 decrypts it upon arrival. System 2 sends the decrypted data internally to port 6661 to complete the transmission. With SSH tunneling, unencrypted HL7 data is never visible outside of the two endpoints.

A real advantage to using SSH tunneling is the easy implementation. SSH tunneling requires very few adjustments on either end and does not require changes to the interface messaging or HL7 formatting, making it very attractive to secure existing interfaces. Leaving most of the configuration untouched is significant because the initial setup of HL7 interfaces is notoriously challenging. If both ends are Unix or Linux variants, there is a high likelihood that SSH may already exist on the operating system and no additional software is necessary.

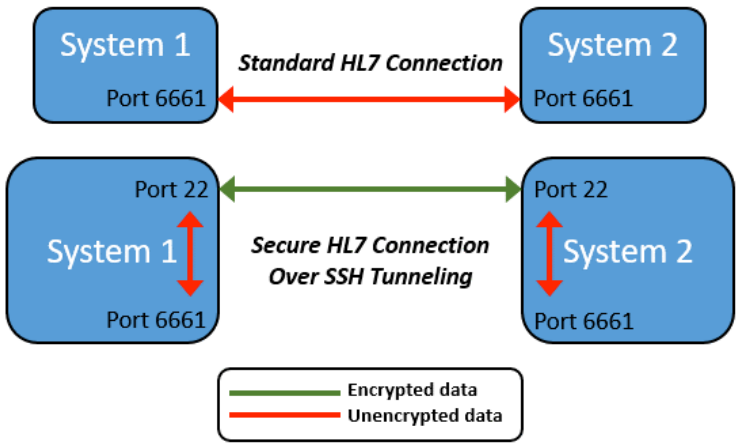

Figure 20: HL7 Over SSH Tunnel Comparison

If either endpoint is using Windows, there are several free options available. After many stalled attempts over the years, Microsoft has finally added SSH into the Windows operating system. To install SSH on Windows, one can also use the Win32-OpenSSH PowerShell package. Follow the instructions at the first link below to install the package. If OpenSSH from Microsoft is a little too bleeding edge, the developers of OpenSSH have a non-PowerShell setup (second link). Although it would be excessive for SSH

tunneling features alone, Cygwin also provides an SSH server and client as a part of its feature-rich, Unix-like environment for Windows.

https://github.com/PowerShell/Win32-OpenSSH/wiki/Install-Win32-OpenSSH

http://www.mls-software.com/opensshd.html

Once the software installation is complete, the remaining setup for an SSH tunnel conversion consists of three phases – 1) configuring password-less SSH access, 2) configuring the SSH tunnel, and 3) modifying the interface engine settings. The steps to create a tunnel on Windows and Linux are very similar. While some commands may differ, the thought process, overall configuration, and outcome are the same.

The creation of public keys often occurs during the initial SSH setup and if not, the 'ssh-keygen' command can accomplish this task as shown in Figure 21. Once the key generation is complete, copy the public key (.pub) over to the .ssh/authorized_keys file on the other system. On Windows or Linux, copying/pasting the public key can be performed manually.

```
Generate an SSH public/private key pair
administrator@BASE-PC C:\Program Files\OpenSSH>ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key
(C:\Users\administrator/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
C:\Users\administrator/.ssh/id_ed25519.
Your public key has been saved in
C:\Users\administrator/.ssh/id_ed25519.pub.

Copying over the public key file - Method #1
ssh-copy-id <user>@<hostname>
Copying over the public key file - Method #2
cat ~/.ssh/<file>.pub | ssh <user>@<hostname> 'cat >>
.ssh/authorized_keys'
```

Figure 21: Generating SSH Keys & Configuring Password-less SSH

On Linux, one can use the 'ssh-copy-id' method found in Figure 21 if the system has it. If not, the second method uses standard tools available on nearly every Linux installation.

Once complete, a simple 'ssh <user>@<host>' command will test the configuration. If the key copying goes as planned, the user will now have access to the second system and bypass the login prompt entirely. If this is a long-term solution, hardening the SSH configuration (sshd_config) is strongly recommended. Hardening SSH would include steps such as disabling password authentication and disabling version 1 of the SSH protocol.

Once the password-less SSH is tested and working, the creation of the SSH tunnel is next. The one-line command for the tunnel is in Figure 22 below. The "-f" backgrounds the SSH session and the "-L" forwards the localhost port. The IP address is that of the remote system while 6661 is both the local and remote port. The "-N" means do not run a command on the remote system and only forward ports. In a more permanent arrangement, a tool such as autossh or a shell script could automatically restart the SSH connection in the event it dies. During testing, the "-f" should be substituted with a "-v" to see the verbose output from the tunnel. With verbosity enabled, each communication including "test connections" will generate logs.

```
C:\>ssh.exe -f root@10.0.0.127 -L 6661:10.0.0.127:6661 -N
```

Figure 22: Creating the SSH Tunnel

Different ports could be used for the newly formed SSH interface if the interface administrator chose to do so. This approach might prove helpful during the testing stage, but changes to the ports would also require modifications to the corresponding ports on the sending and receiving interfaces. Conversely, using the previously configured port numbers would only require a single change to the sending interface configuration. Instead of the data sending directly to the receiving system IP address of 10.0.0.127, the HL7 traffic redirects to the localhost as shown in Figure 23. The SSH tunnel then handles the system-to-system communication. On the receiving side, no additional modifications are required in the interface application as SSH handles the traffic redirection back to 6661. The lack of changes to the receiving interface is tremendously helpful if a vendor controls the configuration.

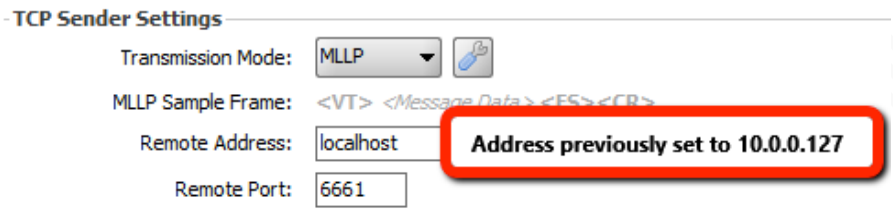

Figure 23: TCP Sender Modifications for SSH Tunneling

SSH tunneling protects extremely well against the earlier MITM ARP spoof attack. As previously discussed, HL7 data sends as clear-text by default. The Wireshark example below in Figure 24 is from an HL7 interface over an SSH tunnel. Although the SSH tunnel did not prevent the ARP spoof attack from occurring, it neutralizes the effectiveness of the attack. The HL7 data is unreadable due to encryption and as important, the data remains confidential and unaltered.

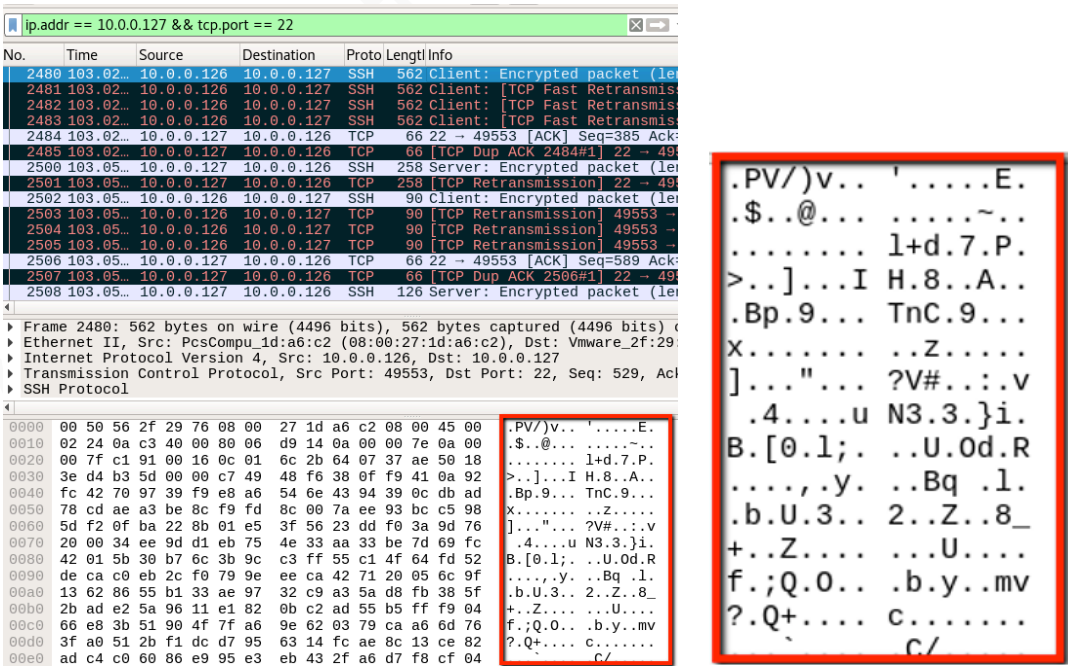

Figure 24: Encrypted HL7 Message Captured During ARP Spoofing Attack

4. Conclusion

As healthcare and security professionals deepen our understanding of how healthcare data can be misused, protecting data at rest or in transit should be a priority. There is no doubt healthcare is receiving plenty of attention from cybercriminals. Though targeted attacks are unlikely to subside, medical data is far too valuable for the current

level of protection. Likewise, it is important to understand where the data lies and where it travels so we can protect it. When recognizing where significant gaps exist in protecting medical data, HL7 should be near the top of the list.

Perhaps the relationship between security, HL7, and its impact on healthcare is best summarized by Scott Rohleder, the CIO at HaysMed, part of The University of Kansas Health System:

> When a person enters a healthcare facility for care, there is a certain level of implicit trust that the information in their medical record is precise. One of the last things on their minds is that their healthcare-related data might be at risk due to insecure communication protocols. However, with the current implemented base of HL7 interfaces that may not be the case… Since all organizations utilize this method to conduct business, it is very important that everyone understands the risks (personal communication, May 5, 2017).

Information security and IT professionals alike have worked tirelessly over the years toward removing insecure communications such as Telnet and FTP. Meanwhile, an obscure standard known as HL7 is used in nearly every hospital worldwide and it possesses many of the same vulnerabilities we have worked diligently to eliminate. The situation is more dangerous considering HL7 messages carry some of the most sensitive data found in a hospital. Yes, the attacks described above require local network access, but that has proved to be a minor barrier in the past decade or more. The ease of gaining a local foothold on a network halfway around the world and pivoting toward more valuable data such as HL7 falls far short of complex for motivated attackers.

There are strengths and weaknesses to each HL7 defense method described. When implemented without additional compensating controls, they all improve security to some degree and eliminate potential attack vectors. However, none of the defenses are perfect by themselves. Instead, a combination of protections will ultimately yield a proper defense-in-depth methodology. All the defense mechanisms work for existing HL7 interfaces, i.e., there is no need to re-design or re-architect an existing HL7 interface. Healthcare needs to do more to protect data and that change needs to happen now. Defenders cannot ignore the extraordinary amount of damage an attacker can cause by

collecting HL7 data or using it for more nefarious purposes. Sensitive HL7 and patient medical data cannot continue sitting in the open, waiting for the next major breach to occur. Our data is too valuable.

References

athenahealth, Inc. (January 2016) *Connectivity Methods Overview*. Retrieved from https://www.athenahealth.com/~/media/athenaweb/files/developer-portal/connectivity_methods_overview.docx?la=en

Fuentes, Mayra Rosario. (2017). *Cybercrime and other Threats faced by the Healthcare Industry.* Retrieved from https://www.trendmicro.com/content/dam/trendmicro/global/en/security-intelligence/research/reports/wp-cybercrime-&-other-threats-faced-by-the-healthcare-industry.pdf

Health Care Industry Cybersecurity Task Force. (June 2017). *Report on Improving Cybersecurity in the Health Care Industry.* Retrieved from https://www.phe.gov/Preparedness/planning/CyberTF/Documents/report2017.pdf

HL7 International. (2007, August 31). Implementation FAQ: Encryption and Security. Retrieved January 26, 2017, from http://wiki.hl7.org/index.php?title=Implementation_FAQ:Encryption_and_Security

iNTERFACEWARE. (2013, December 5). Secure Protocols for HL7. Retrieved February 13, 2017, from http://help.interfaceware.com/kb/164

Medical Informatics Engineering. (2016, July 6). Sample HL7 Messages. Retrieved June 1, 2017, from http://www.mieweb.com/wiki/Sample_HL7_Messages

Suiche, Matt. (June 2017). Petya.2017 is a Wiper, Not Ransomware. Retrieved July 1, 2017, from https://securelist.com/expetrpetyanotpetya-is-a-wiper-not-ransomware/78902/

Tarala, James & Tarala, Kelli. (April 2015). *The What, Where and How of Protecting Healthcare Data.* Retrieved from https://www.sans.org/reading-room/whitepapers/dlp/what-protecting-healthcare-data-35887

Zurier, Steve. (December 2016). 91% Of Cyberattacks Start With A Phishing Email. Retrieved May 1, 2017, from http://www.darkreading.com/endpoint/91--of-cyberattacks-start-with-a-phishing-email/d/d-id/1327704