A Survey on Packet Injection

Cole Faust, William Hight, and Alexander Henry

Abstract—Packet injection is disrupting, intercepting, or interfering with the network traffic between other computers. It can be used to provide a client with falsified information that is perceived as genuine, terminate a connection between two parties, and read data transmitted, among other broad applications. We explore attack vectors such as sending TCP reset packets to interrupt established connections, synchronizing sequence numbers to cloak the effects of injecting packets in out-of-band scenarios, utilizing security flaws in the ARP protocol to establish a man-in-the-middle, and exploiting quantum insert which can send false data without a trace. We will discuss real world scenarios of packet injection, methods of detection and defense, in addition to tools that can be used to execute and prevent such attacks.

I. Introduction

Packet injection is an incredibly broad topic with many applications. In this paper, we cover several specific instances of injection, and how it can be abused, detected, and defeated.

ARP spoofing, or cache poisoning, is a technique to have one computer take the IP address of another, leading to a potential man-in-the-middle attack. Once a man-in-the-middle has been established, the attacker has full control over the packets that are sent across the network. However with a new secure version of ARP, it might be possible to remove the threat of this attack.

Using an out-of-band method of injection called Quantum Injection, one could race the requested web server in delivering information to the victim. Whenever a victim makes a request, an attacker could detect this and send a response faster than the web server hosting the information, resulting in the web server's packets being dropped after the attacker's forged data has already been received.

One realm of packet injection revolves around controlling connections between communicating parties—more specifically, terminating types of connections deemed unwanted by an adversary. Exploiting the use of forged reset TCP packets (RST packets) , connections can be falsely terminated, leaving the communicating parties with an undesired and swift exit.

It is also possible to inject arbitrary data into an existing TCP connection without disrupting it. If you are able to re-synchronize the sequence and acknowledge numbers of the two ends of communication afterwards, the parties are able to continue legitimate communication, and the injected data can sneak in unnoticed.

Packet injection doesn't have to limit itself to one protocol or channel of communication. We'll show examples where a TCP request over the internet can lead to injected WiFi frames, which don't even necessarily have to have the same victim as the one making the initial request.

Finally, we'll look at several methods of detecting spoofed packets by determining if their source address is legitimate. These methods include inspecting the TTL and IPID of packets, and taking an active response to attempt to determine the true identity of the source address.

II. Background Knowledge

The most common protocol to send data over the network with is the Transmission Control Protocol (TCP). TCP is another layer after the IP layer, which has the primary goal of ensuring packets are not lost or out of order when sent over a network. TCP accomplishes this goal by including a sequence and acknowledge number in the header of every packet it sends. The sequence number (SEQ) is a count of how many bytes have been sent so far, and the acknowledge number is a count of how many bytes that have been received. If a client receives a packet with a sequence number past it's current acknowledge number, that means that it received a packet too early and it should hold onto it until the stuff in between comes. If a packet is received with a sequence number less than the receiver's acknowledge number, the receiver will drop it, because it does not make sense to receive a packet in such an order.

Other fields of a TCP packet, such as the Internet Protocol Identifier (IPID), Acknowledgement Number (ACK), and Time To Live (TTL) can be useful in identifying forged packets, by expectations of consistency by both valid sources, and specific packet injection software.

TCP has a special header bit called the reset bit (RST). If a client receives a packet with this bit set, it will drop the connection and drop all future packets coming on this connection.

A man in the middle attack is an attack where the perpetrator can inject themselves in the communication between two parties. Let there be a communication between Alice and Bob. Mallory wants to intercept this conversation, however, so as Alice initially greets Bob, Mallory steals this message and relays it to Bob. This message looks like it's coming from Alice, but in reality Mallory read it and sent it to Bob, so Bob will send back a greeting to Alice. This is then relayed again by Mallory, of which Alice thinks its origination is from Bob. This is the most simple form of a man in the middle attack.

Contrary to man in the middle, simply injecting packets without rerouting any other traffic is called an out-of-band or man on the side injection.

The ARP protocol consists of two main different messages, an ARP request and ARP reply. An ARP request asks "Who has this IP?" This is broadcasted across the entire network, requesting the MAC address of the computer with the IP. An ARP reply is when the one

computer with that IP address replies with its MAC address, all other computers on the network ignore the request. It is also possible to request the identity of a MAC address of an IP address using RARP, but this is not used often.

To gather information about the devices on the network, a host or router would broadcast an ARP request, and the appropriate machine would send an ARP reply.

A WiFi MAC frame is OSI Layer 2 equivalent for the 802.11 protocol. It provides the basic datagrams for packets sent over WiFi. In addition to routing information that we're familiar with Ethernet, it also includes extra fields to cover authentication, broadcast the network name (SSID), and other control messages.

III. Literature Review

Franco Callegati, Walter Cerroni, and Marco Ramilli discuss the dangers of assuming that when a browser's lock icon appears indicating a secure connection, your information is secure. From their abstract, "we show the danger of that assumption by demonstrating how attackers can successfully intercept the data transfer and corrupt the safety of the communication" [1]. Using ARP cache poisoning, the attacker can forward requests, intercept replies, create a self-signed certificate to replace the server's, send it to the client, and snoop on the encrypted channel between the client and the server with the false certificate, all while remaining undetected.

D. Bruschi, A. Ornaghi, and E. Rosti from the Univeristy defli Studi di Milano, Italy present "a secure version of ARP that provides protection against ARP poisoning" called S-ARP [2]. Every host on the LAN has a public / private key pair that is certified by a local trusted host on the LAN. ARP messages are digitally signed by the sender, so it effectively prevents spoofed ARP replies. They revealed that it is feasible to implement with low performance costs.

Erik Hjelmvik is a network forensics software developer who writes several articles on his website about network security incidents and attack vectors on networks. He wrote an in-depth analysis on what a man-on-the-side attack is, the "Quantum Insert" variant of it [4], and how this was used by the Chinese government to DDoS GitHub.com [3].

In the paper, Detecting Forged TCP Reset Packets, a team from the International Computer Science Institute (ICSI), Nicholas Weaver, Robin Sommer, and Vern Paxson, developed a toolbox to help classify, and counter the usage of TCP RST packets. With analysis of real web traffic and possible instances of forged RST packets, they showed that the tool is effective at both detecting the forged packets, and also classifying the type of tool / software used to deploy said packets. Instances of forged RST packets appear within large ISPs, such as Comcast, and and also censoring governments, such as the Great Firewall of China, among other malicious applications [5].

Researchers Oliver Zheng, Jason Poon, and Konstantin Beznosov at the University of British Columbia were able to use exploit the protocol of certain applications in order to inject packets into TCP streams without relying on a man in the middle, or disrupting the TCP stream by creating a mismatch in sequence and acknowledge numbers. This is one of the few examples of an out-of-band attack that does not interrupt the entire stream [6].

Pieter Robyns, Peter Quax, and Wim Lamotte from Hasselt University were able to exploit WiFi frame aggregation in order to inject WiFi MAC frames from an unrelated TCP stream. This allows us to make attacks over a stream that may not be related to the true victim [7].

Steven Templeton and Karl Levitt from UC Davis discussed how to detect whether a particular source IP was legitimate or nor, using a variety of methods [8].

IV. Discussion

A. ARP Exploits

ARP cache poisoning is a LAN method to establish a man-in-the-middle (MITM) attack. The network host caches mappings of IP to MAC addresses using the Address Resolution Protocol. The problem with this is that network hosts automatically accept any ARP reply by any machine in a DHCP setup, regardless of if a request was made by the network host [1]. The attacker can spam ARP replies on the LAN to the network host to effectively make all incoming traffic that would originally go to the victim's device, go to the attacker's device. Similarly, the attacker can "poison" the victim's ARP cache that holds all the entries to route all outgoing traffic to the attacker's device by sending ARP replies to the target victim. The attacker has just routed all traffic to his own device, establishing himself as the man in the middle. As discussed with MITM, an attacker would be able to intercept and read all information passing through the new spoofed gateway, in addition to being able to drop any packets from either side of the connection [1]. Most implementations of ARP have no foolproof mechanism to secure the protocol, so this is a very effective method to attack a LAN.

There are a number of methods to defend against ARP cache poisoning. Hard-coding the ARP cache ensures that ARP replies that update the statically assigned IP address will be ignored [2]. The downside is that IP addresses then cannot be re-used and the cache has to be manually updated on all devices in the network. Monitoring the ARP traffic allows one to identify suspicious activity and be aware if the apparent network host is trustworthy or not. These are the only effective methods to counter the attack, however methods that mitigate MITM attacks or their damage are also effective. Using public key infrastructure such as SSL / TLS, the victim client and a server can exchange certificates that are issued and verified by a certificate authority. As long as the original key to authenticate the CA wasn't transmitted during the MITM attack, then the certificates can be used to authenticate the messages sent by either party.

B. Secure ARP

Securing the Address Resolution Protocol is a feasible endeavor using public key infrastructure [2]. Instead of trying to mitigate damage on normal ARP, S-ARP is a protocol that modifies existing ARP to be secure. S-ARP only provides message authentication, not encryption, as MAC addresses in the network should be public. Every host in the network has a public/private key pair, and a certificate that binds the host identify to its public key [2]. The certificate also contains the host IP address and the MAC address of a trusted local host acting as a sort of certificate authority. This host, otherwise known as the Authoritative Key Distributor (AKD), acts as the key repository. Every host on the network sends its signed certificate to the AKD, which stores the IP address and public key of the host in its local database [2]. Instead of keeping a revocation list, the AKD distributes a clock that all hosts on the network must synchronize to for the purposes of defeating replay attacks.

All S-ARP replies sent by hosts on the network are digitally signed by the sender using their private keys. The receiver verifies this signature with the host public key. If the receiver doesn't have the public key, it requests it from the AKD. AKD then sends the corresponding host's public key and the clock time to synchronize to in a digitally signed message by the AKD. When the receiver host gets the message from the AKD, it checks the new time against the timestamp on the sender host's message. If it's too old, it discards it. This prevents replay attacks as an attacker could just use an old S-ARP reply to spoof its MAC address on the network. If the timestamp and the signature checks out, the reply is considered valid and the receiver hosts caches the new information from the AKD and sender.

One downside, however, is the single point of failure that is the AKD [2]. Despite this, the performance of the protocol is good enough to be implemented on low end systems as well. With proper message authentication, S-ARP takes aim to rid ARP cache poisoning forever.

C. Man-on-the-Side

On March 26, 2015, GitHub experienced a very large DDoS attack. This distributed denial of service involved unsuspecting people who had nothing to do with GitHub, and the original source of the attack was unknown. On the surface it looked like a lot of IP addresses were attempting to access github.com/greatfire and github.com/cn-nytimes, however the people behind the IP addresses had no knowledge of this [3]. It was found that China was using their network infrastructure to use packet injection to DDoS GitHub. A user browsing internet from inside or outside China could visit a website that loads JavaScript from inside of China, such as Baidu Analytics, a script that is used to track visitor statistics similarly to Google Analytics. The request for such a script was detected by China's network infrastructure, and a spoofed response back with a malicious script was sent instead. The script itself made a request for the GitHub pages aforementioned, and since Baidu Analytics is used on a grand scale, the DDoS was very effective.

The key to this attack is that this script was injected without establishing a MITM attack. The attack vector used here is a man-on-the-side, or out-of-band attack. The packets that were sent back looked like they came from Baidu, however while the original request to Baidu succeeded and a response was sent back, the response from China's network got to the web browser first, resulting in Baidu's response packet with the expected script to be dropped [3]. While this incident is a result of the Great Firewall of China, it doesn't necessarily require such a network.

A man-on-the-side injection attack such as this is sometimes called a "Quantum Insert" attack [4]. For example, whenever a GET request for a website is sent through the network from the victim, the attacker can spot this and send a response to this request that is forged to look like it came from the server originally hosting the content. This forged packet needs to arrive before the web server's packet. The forged packet contains a FIN flag to terminate further communication with the web server and results in all other packets being dropped, so the web server's packet is dropped on arrival. Thus completes the out-of-band injection attack.

Detecting Quantum Insert is tricky. One indicator that a packet was injected due to this attack vector is the time-to-live (TTL) field in the header. The injected packet's TTL field will typically be much different than any other packet sent by the server, due to the fact that the attacker will be closer in proximity than the server [4]. Another possible indicator is that both the forged packet and the real one will likely have the same sequence numbers. In general, detecting Quantum Insert attacks is difficult.

D. Reset Packets

TCP reset packets are a way for the TCP protocol to terminate a connection in erroneous situations, where the connection cannot be sustained. Either triggered by an unreachable socket or aborting a connection, once an RST packet has been sent, no more packets should be sent in that TCP session. Additional RST packets may be sent to reiterate the message, if the recipient of the RST keeps sending packets, or had packets in transit while the initial RST was sent.

Forging RST packets is dangerous for the very reasons described above. With the power to terminate a TCP session, a client would logically think it is an issue with their communicating counterpart (for instance, a server). In this case, a client would receive a forged RST packet, assume that the connection has been terminated, and no longer accept any packets from the previously established TCP session with the server. A forged RST packet could similarly be injected during a TCP handshake, denying the connection nearly instantly. With the assumption that the server is inaccessible, the client is left with no service, which on a large scale can prove to be catastrophic.

To combat forged RST packets, it is imperative to first be able to detect them—genuine RST packets serve a fundamental purpose for TCP connections. Valid RST packets should be accepted to terminate the connection when necessary, but accepting a forged RST packet is a serious disruption of service for users that degrades accessibility, so classifying an RST packet as forged should not be taken lightly.

The most rudimentary, and telling method to detect RST packet injection requires packet sniffers monitoring the traffic between two communicating points. If there are inconsistencies between the RST packets sent and received from both ends, there is evidence of a forged RST packet. This method, although simple, is not practical for use as an active countermeasure due to its requirement of access to both communicating machines [5].

There are new tools to help detect RST packet injection from a single device, developed by security experts and researchers. One of the tool kits was developed by an ICSI team. By gathering data at several Universities, they were able to develop and test tools to help identify, classify, and actively ignore forged RST packets. Using several different properties of a standard, and genuine RST packet, it is assumed that individual RST injectors are generally consistent relative to themselves, and can differ wildly from valid RST packets [5].

It is important to understand what types of fields there are in a TCP packet, and which fields tell us the most about the packets origin, and its authenticity. Of the many parts of TCP packets, the ICSI team focus on the SEQ, ACK, IPID, and TTL aspects. The SEQ field is used to determine if the packet is in the proper order, and with RFC standards, there is a margin of error on sequence numbers to accept valid, delayed packets among other edge cases. The SEQ field should not be lower than a previously sent packet, and such properties already protect users from blind RST injection. While the latter three fields are not required for RST packets, it is assumed that they remain somewhat consistent when being sent from a single source, benign or malicious. While this cannot help us detect if a RST packet is forged, this expected consistency can help categorize different types of RST packet injectors, while the sequence number can be used to identify it as forged.

Of all of the TCP header and payload fields, the sequence number is the hardest to mimic for a forged RST packet—making it the most important to analyze in the search for forged RST packets. With an adversary injecting packets that have a higher SEQ value than the triggering packet[1], the adversary is in competition with two race conditions that can help detect the fraudulent from the genuine.

The first detector, named RST_SEQ_DATA in their toolbox, utilizes the race condition innate to most injection scenarios. This race condition is caused by latency between when the adversary decides to send a forged RST packet, and when it is actually sent. In this short time period, packets can continue to be sent from the sender to the receiver, meaning that the injected RST packet will have an incorrect or inconsistent sequence number [5].

Another race condition is present in an injection scenario due to packets already sent, or queued to be sent from the sender to receiver. Named DATA_SEQ_RST, this detector is effective at identifying this second race condition, to determine if a RST packet has been forged. Assuming a forged RST packet is sent, and is accepted by the RST_SEQ_DATA test, there will still be packets in transit, and in queue, from the sender to receiver. While the forged RST packet makes the sender think the connection has been terminated, this detector can still determine that the sender is still sending other packets, indicating it didn't send a RST packet after all, greatly increasing the chance that the RST was forged [5].

The other tests developed were much less important in detecting actual injection, but serve the purpose of classifying which type of injectors are used. By analyzing the values in ACK, TTL, IPID (and even in some cases, payload) after a RST packet was determined to be forged, the team was successful in identifying multiple different fingerprints for injectors, most notably the ones used by the Chinese Government and the United States' large Internet Service Provider, Comcast, two incredibly formidable instances of forged RST packets [5].

Around 2007, Comcast customers began noticing and reporting BitTorrent connection issues. Some users found that when attempting to seed a public domain file (not protected by copyright), an unexpected TCP RST Packet was received, effectively terminating the transfer, with neither communicating party consenting. These issues ended up being found across many different types of peer-to-peer sharing sites. The EFF tested these claims by running their own tests with Comcast customers on BitTorrent, with the aforementioned (but not practical) method of having a packet sniffer on both communicating ends, and comparing the sent and received packets. Coincidentally, the Associated Press was concurrently running similar experiments, with results that reaffirm the EFF's findings [5]. Those findings were that a third party was indeed injecting RST packets, with such injections only occurring with Comcast customers, indicating that Comcast was the guilty party. Comcast attempted to justify its use of RST packets as a mere delay in connection, due to congested networks, to both customers and the FCC. The EFF found that their explanation was inconsistent with their usage, as the injection of RST TCP packets affected packets on connections seeding any file, not discriminating based on size, but rather connection type . In normal, likely, everyday situations, the "Comcast customers will not experience the interference as a 'delay'; their software simply won't work" [9]. Popular for distributing pirated films, BitTorrent has also begun to distribute licensed films, among many other types of media. So in this case with Comcast, not only is net-neutrality at risk, but serious conflicts of interest arise. For instance, if

[1]The triggering packet creates an event which begins the injection process (i.e. once the attacker has decided to send forged packets to a target) [5]

Comcast blocks legally distributed films, they could stifle innovative competition in the market, while promoting and expanding their own cable / film services [9]. Although Comcast was ordered by the FCC to halt such practices in August of 2008, this was still a very awakening event regarding an ISP's power over the connections they serve, alongside the power of injecting forged RST packets [10]. While there are clearly some serious threats that arise from ISPs having this injection ability, the tests ran by the ICSI team also yielded cases where these injection techniques were aimed to shut down malicious connections, in order to protect costumers—a constructive and moral way to apply this dangerously powerful method [5].

Confirming the usefulness of the detection toolbox that the ICSI team designed and built, a fingerprint was successfully generated for Sandvine, the software Comcast uses to inject RST packets. This fingerprint could effectively could be used to ignore these forged RST packets, preserving the connection, with widespread usage [5]. However, the most promising tool that developers have to protect users from such attacks lies with cryptography, which can help confirm the identity of a packet sender, and obfuscate the protocol being used [9]. China, in an effort to block undesired connections, used a variety of tools that modified the IPID by constant addition, alongside simple incrementation of SEQ values by 1460, in a simple, pseudo-brute-force method. China injects RST packets to connections that it finds inappropriate, implemented with the fitting name of the Great Firewall of China. While there are methods to get around this firewall, it serves it's purpose by greatly increasing difficulty to access information the Chinese Government does not agree with, a serious threat to an open, data-driven world [5].

E. Non-disruptive TCP Injections

It is possible to inject packets into an existing TCP connection without disrupting it, however problems arise. The packets you inject will cause the acknowledge number of the victim to rise, disrupting any further legitimate communication with the real host, since it will have a lower sequence number. One way of resolving the sequence / acknowledge desynchronization between two clients is to inject more packets that cause the clients to respond. When a client responds it will increase its own sequence and acknowledge numbers, and by increasing them on different amounts on two clients you can cause their counts to become synchronized again. This requires the response to be bigger than the packet that caused the response, because we're trying to make up for added acknowledge. If the response was smaller than the message that started it, it would create a larger gap between the sequence and acknowledge numbers, which is the opposite of what we want. To find the number of packets to send to each client, you can create a system of equations comparing one server's sequence number to the others' acknowledge number for one equation, and the other way around for the other equation. Since the injected packet only changed one client's acknowledge number, only one equation will have a constant offset added to it to represent this difference [6].

This method has a few limitations however. For one, no other legitimate communication could occur when resynchronizing the clients or else they'll create more of an offset. This could be accounted for by sniffing out the legitimate packets and changing how many false packets you send, but it requires more work. Also, this attack is not effective against a Linux client, as Linux goes against the RFC standards and accepts packets even when they have an acknowledge number greater than what was expected. This causes that client's acknowledge number to rise, which makes it impossible to catch up. Windows does not do this however, and is still vulnerable to the attack.

F. WiFi Frame Injections

Not all injection attacks have to remain in one protocol. Another type of packet injection attack involves injecting WiFi (802.11n) frames to a network over the internet. Some background knowledge to be familiar with, is that 802.11n's lowest two layers of abstraction are the PYS layer, which contains information about the wireless transmission and other radio properties, and the MAC layer, which has the basic logical commands that WiFi needs to be able to do. One interesting property that is new to 802.11n, (as opposed to the prior a, b, and g) is that MAC frames can be aggregated into one packet. There are two ways to do this, called A-MSDU and A-MPDU, which are acronyms for the aggregation of MAC service data units and MAC protocol data units, two ways to aggregate MAC frames together for performance reasons. A-MPDU has the property where if one of the subframes in the aggregate is corrupted, the receiver will continue looking for the rest of the data for more MAC headers. Since the MAC header was corrupted, it doesn't know how far to jump ahead to get past the corrupted one, and instead only checks every 4 bytes for an A-MDPU delimiter. If the corrupted frame were to have an A-MDPU delimiter in its payload, the receiver would start processing that as if it were a frame that was actually part of the aggregate. It is practical to trigger an aggregation of frames and for one of them to be corrupted. Several tests shown that it can happen within a few hundred megabytes of transmitted data [7].

This allows us to inject our own WiFi frame. Now, as for what we can do with that, our options are somewhat limited because most MAC frames require a source and destination MAC address, which an attacker over the internet wouldn't know. There are a few things we can do however, such as injecting a bogus SSID frame, which doesn't require you to know a MAC address. The transmitting address can be anything, because you should be able to receive SSID's from a network you're not part of yet, and the receiving address is just the broadcast address, ff:ff:ff:ff:ff:ff. If you can guess or are otherwise informed of the router's MAC address, you can make ping echo requests to the broadcast address, but with a certain

IP. One example where this is useful is scanning to see what IP's have computers associated with them on an internal network.

This attack is pretty exotic, but there's some pretty trivial ways to combat it. Simply encrypting your wireless connection will scramble the injected MDPU frames, and prevent this attack. Some other ways include disable A-MDPU aggregation or dropping corrupted A-MDPU frames, but you'll lose out on the benefits they provide.

G. Detecting Packets with Spoofed Sources

In certain packet injection attacks, such as many denial of service attacks, the attacker may construct a packet with a source IP address different than his own. The goal of this could be to make a victim generate traffic towards another victim, or hide the attacker's identity. There are several ways we can detect and prevent this, both simply by inspecting the packet itself, and by taking additional action to try and validate the packet [8].

One immediate red flag that a packet is spoofed is if it's source IP address is for an internal IP but it comes from over the internet. This can happen in the case that an attacker wants to indirectly affect a machine on a network, similar to how the host scan in the WiFi attack section worked. Software on the router can be programmed to detect and drop these packets, as the router can clearly tell what's coming from outside. The reverse can also happen, where an attacker uses a different external source IP than it really is. The router can detect these internal packets and drop them as well. These two types of filtering are called ingress (packets coming in) and egress (packets leaving) filtering, and can be useful as a first line of defense.

If a packet is coming from the internet and has an external IP, we can still verify that it actually came from that IP with a few tricks. One way is to look at the TTL of that packet and see if it is what we'd expect from that IP. We can send a ping to the source IP and see if the reply has the same TTL as the potentially spoofed packet. If it doesn't, it's likely been spoofed, as the attacker must be located a different number of hops away from us. One thing to keep in mind though is that different protocols (namely ICMP for the ping and TCP) have different starting TTL values, and there are a few options for starting values, so you'll have to check all of them. This TTL check is difficult for the attacker to spoof because he can't easily figure out how many hops are between his victim and the spoofed source address. The expected TTLs for a host can be cached and only updated when a suspicious packet is found, so as not to incur much more network overhead.

The TTL method only works when there are some hops the attacker has to go through. Another method is examining the IPID field, which is an identifier on each IP packet. This identifier generally increments by one for every new non-fragmented packet. So if we were to send a packet to the source shortly after receiving a questionable packet, the source should respond with ideally a packet with an IPID of one more than what we saw before.

V. Conclusion

Packet injection is a useful vector of attack, as it's easy to get started sending forged packets. Almost every modern computer relies on network connections with others, and would suffer greatly from forged data being injected into its communication. Because of the fundamental role packets play with the communication between devices, applications of forged packet injection affect many network functions. As with all packet injection, packet sniffing is instrumental to determining what to forge and inject. With the specific method of resynchronizing the TCP stack, the injection can go undetected but still alter the traffic, whereas forged RST packets, adversaries are given power to shut down undesired connections. Quantum Insert allows undetected falsification of data, while ARP spoofing gives attackers full access to a system network. As a whole, all forms of packet injection act to compromise confidentiality, integrity, and accessibility of the impacted systems with frightening ease.

References

[1] F. Callegati, W. Cerroni Man-in-the-Middle Attack to the HTTPS Protocol
http://ieeexplore.ieee.org/document/4768661/

[2] D. Bruschi, A. Ornaghi, E. Rosti S-ARP: a secure address resolution protocol
http://ieeexplore.ieee.org/document/1254311/

[3] E. Hjelmvik, China's Man-on-the-Side Attack on GitHub
http://netres.ec/?b=153DB4E

[4] E. Hjelmvik, Covert Man-on-the-Side Attacks
http://netres.ec/?b=1598A63

[5] N. Weaver, R. Sommer, and V. Paxson, Detecting Forged TCP Reset Packets
http://www.icir.org/vern/papers/reset-injection.ndss09.pdf

[6] O. Zheng, J. Poon, and K. Beznosov, Application-Based TCP Hijacking
https://pdfs.semanticscholar.org/95a4/8f36f04882cd5c2ad9869a3f13e8387009f5.pdf

[7] P. Robyns, P. Quax, and W. Lamotte, Injection Attacks on 802.11n MAC Frame Aggregation
https://github.com/rpp0/aggr-inject/blob/master/paper/ampdu_inj_wisec2015.pdf

[8] S. Templeton, K. Levitt, Detecting Spoofed Packets
http://ieeexplore.ieee.org/document/1194882

[9] P. Eckersley, F. von Lohmann, and S. Schoen, Packet Forgery By ISPs: A Report On The Comcast Affair
https://www.eff.org/files/eff_comcast_report.pdf

[10] R. Kenny, Commission Orders Comcast to End Discriminatory Network Management Practices
https://apps.fcc.gov/edocs_public/attachmatch/DOC-284286A1.pdf