Michael Guidry

September 20, 2017

A realistic way to bypass signal intelligence on the Internet

Signal intelligence platforms use unconventional methods for extracting raw information from Internet backbone providers. It allows capturing all protocols, and sessions flowing through the providers. It is however at a disadvantage in certain ways, and it isn't something that's possible to fix. Abuse of these circumstances would allow you to perform network activity without being captured by mass surveillance frameworks. Automation of the techniques are quite easy, and would allow integration directly into any operating system.

Surveillance platforms obtain initial raw traffic by passively monitoring fiber optics cables connecting the worlds Internet providers [4]. The providers do not have to necessarily allow it either. The Snowden leaks give a lot of information regarding GCHQ, and NSA abusing this particular method for their intelligence platforms [1]. It captures all packets, and then begins to filter by protocols it considers of importance. The packets are reconstructed virtually using various parameters inside of the TCP/IP [9] headers. The final entire session is then ingested, and processed into a database for long term accessibility.

Operating systems, such as Windows, perform accurate processing of networking events. Each network connection updates a consistent context of that connection. It understands through all fragmented packets whether or not an event is taking place which requires healing by instructions within the protocol's technical specifications. If the instructions are performed correctly, then the issue is usually solved. It does have certain criteria which would close the connection permanently. Surveillance platforms capture so many connections simultaneously that it is impossible to perform the same due diligence. The surveillance technology itself will continue to contain these vulnerabilities for the foreseeable future. It is a byproduct of the method being used to obtain so much information to begin with.

The TCP/IP protocol has various fields of its core header which ensure packets do not get interpreted by software out of order. It allows mass surveillance platforms to reconstruct virtual sessions, and also lets routers resend packets if there was some sort of problems. The same acknowledge, and sequence portions of TCP/IP communication also allow the Quantum Insert attack [2]. It is very difficult to guess these values for a connection without the ability to capture the packets relating to it.

If you wanted to ensure these wiretap operations aren't able to capture your information correctly then there are also things you can do to bypass these platforms. The surveillance platforms expect everyone to use proper TCP/IP communication according to the protocols. Its necessary due to all major websites, and anything you would actually want to communicate with online would need to speak the same language. It is however possible to have a third party server on the internet helping you

manipulate the surveillance platforms in your favor. The software on your very machine would have to understand the packets arriving, and know whether one of these mechanism are taking place.

An example of one option would be to notify a third party server that is located somewhere in the world, preferably on the other side of the wiretap, about your upcoming connection. It would be a helper agent developed solely to help you bypass signal intelligence capturing systems. A single helper server could prevent millions of Internet users from being captured by these platforms.

TCP has a 3-way handshake [3] required for opening connections between two computers on the internet which is used for everything while you're browsing the web. The source computer, or web browser, will send a small amount of information called a synchronous value to the web server stating that it would like to open a connection. The web server uses the synchronous value inside of that packet to send back a packet containing an acknowledging number which states that it understands the request, and is preparing the connection. The web browser's operating system builds a packet using these values, and sends a new packet to the web server accomplishing the last step of the handshake. The connection is finally active, and established on both sides.

Once the connection is alive the acknowledgement, and synchronous values change during communication. The values are only supposed to be available to either of these two computers. If one of the computers receives a wrong acknowledge, or synchronous value then the possibility of a few situations are assumed such as packet loss. The protocol contains instructions for attempting to solve these problems. Surveillance platforms even use these values to reconstruct TCP/IP sessions for processing. The protocol specifications have existed for decades already. This is a wonderful thing since the entire Internet speaks the same language. It allows routers to function properly, and software to build other protocols on top of this system which are considered reliable. It allows some bad scenarios such as mass surveillance although the benefits outweigh the negative effects.

If you wanted to take a shot back at mass surveillance to manipulate their systems, stop your communications from being captured, or block attacks by Quantum Insert then you are in luck. It is possible to perform various actions regarding the protocol with these intensions in mind. I will only give examples regarding synchronous, and acknowledgement values. If you want to attempt to fully hack these mass surveillance platforms without having access to the source code then that depends on your own research.

All of these examples require a third party helper server on the Internet. It shouldn't be inside of your local network, or data center because those packets would never travel through the surveillance wiretap. It will only work if the surveillance wiretap parses all packets from this third party server as well as your outgoing packets to the Internet. The third party server could easily handle millions of separate computers helping each bypass signal intelligence on a mass scale. It would be smart to have a network of a few dozen worldwide which would work together with small programs installed on end user computers. Another option is for Internet Service Providers to install these tactics directly into their data centers, which could perform these actions automatically for every connection that they wish to secure by means of filters although the exact steps would be different. The wiretaps themselves are

vulnerable solely because they cannot accurately interpret every packet correctly as an operating system must. The operating system could never handle the amount of traffic that the wiretaps can though. Each have their own pros, and cons. If you want secure communications online then you must capitalize on this scenario.

This external helper service would allow you to force surveillance platforms to drop your packets. You could replace certain packets with random garbage data which would manipulate what is being captured. The surveillance platforms cannot process packets that it does not understand. It has the same issues as the rest of the internet whenever it comes to spoofed packets [5]. A spoofed packet is one which is claiming to be from one source, although actually from an entirely different source. It is how Quantum Insert performs its injection hack to redirect web browsers to FOXACID [6] servers for further exploitation. You could essentially bypass all Quantum Insert attacks using this method as well. It however comes with a small downside in my opinion.

The downside relating to the third party service is that it needs all information regarding web servers that the browser is connecting to. This information is critical to the operation of manipulating the surveillance platforms to bypass their systems. The ways of how this information is communicated is up to the software developer. It could be a simple stateless UDP packet, or it could have a continuous connection with cryptographic operations on top for security. It really works the same either way. I do not care if I am letting a server know which connections I am making so that it can protect me from systems that are not in my best interest.

The web browsing computer must have the ability to ignore certain packets, and process others. It would require a firewall which would understand that these processes are taking place with intention of removing the packets which are sent by the third party server. If those packets were not removed then all of the connections would fail because the protocol wouldn't match correctly. This is exactly why it will manipulate surveillance platforms.

The idea is that the third party server will send you spoofed packets of which the surveillance wiretaps believe is actually from the server that you are attempting to connect to. It means that as long as your web browsing computer doesn't react to that packet, and instead waits for the actual packet from that web server then your Internet would function properly. It would require the ability to have a firewall on the computer that is connecting to the web server. The firewall must recognize connections that it wishes to secure from these capturing systems. It would have to notify the third party server with the information about its upcoming connection. The third party server would generate a spoofed packet supposedly coming from the web server you're attempting to connect to. Several types of packets would do the trick, although a reset (RST) packet is the most convenient. The packet would reach your computer before the web server could send its own packet. The software on the web browser's computer would wait until it receives the requested packet from the helper server that it will ignore. The web server's actual authentic packet would arrive, and it would continue on to the next stage of the TCP/IP handshake. It's a simple explanation of an effective way to bypass signal intelligence platforms currently being stationed around the world by all governments.

The network engineers out there should take a deep breathe. How can you be sure that the incoming packet which is spoofed from the web server is actually from the web server, or from this third party server? You can have the firewall associated with the operating system queue the packet for 10, or even 400 milliseconds. You could even have it wait until the third party server acknowledges that it's sending the packet. It would ensure the entire operation takes place in the order that is required to successfully bypass these platforms. Responding to the fake packet with this firewall application would allow you to further manipulate the surveillance platform. It would allow further manipulations to take place such as inserting random garbage data into the stream at various moments if you don't want to be completely ignored on every connection.

Your system doesn't have to perform this operation for every single connection. A browser helper object, or plugin for Chrome would allow you to determine exactly when, and if you want to perform this task. It wouldn't cause your download speed to crawl even with large media files. The entire operation only needs to take place once per TCP/IP socket to have these systems forget your connection. You could handle it as an operating system driver in Windows using the Windows Filtering Platform [7], or you could perform the tasks on a Linux squid server [8]. The options are endless for exactly how to implement the mechanisms. Quantum Insert wouldn't even be effective because essentially it is attempting to hack a connection that it doesn't even follow.

References:

1. https://nsa.gov1.info/surveillance/
2. http://resources.infosecinstitute.com/analyzing-quantum-insert-attacks/#gref
3. https://support.microsoft.com/en-us/help/172983/explanation-of-the-three-way-handshake-via-tcp-ip
4. http://www.janitha.com/articles/passive-splice-network-tap/
5. https://en.wikipedia.org/wiki/IP_address_spoofing
6. https://techblahblah.com/2013/11/13/what-is-foxacid/
7. https://en.wikipedia.org/wiki/Windows_Filtering_Platform
8. https://en.wikipedia.org/wiki/Squid_(software)
9. https://en.wikipedia.org/wiki/Transmission_Control_Protocol