COMP 8505 Final Project Tech Report

Alex Zielinski – A00803488

03/12/2018

How it Works

My backdoor program makes use of the library libpcap for reading and sending of packets. This was done so that my program can pick up packets before they hit the firewall. This means that my program will still work even if the default rule for the INPUT chain is iptables is set to DROP. The covert communication between the server and client machines is done using TCP and only holds one byte of data per packet (so if 8 bytes of data need to be sent, then 8 packets will be sent). Users also have the option to set a minimum and maximum delay time (in seconds) as to spread the sending of packets to a random timed interval.

Two header fields are used to covertly hold data. The TOS field within the IP header is used to authenticate and categorize packets. The urgent pointer field within the TCP header is used to hold 1 byte of data and is being covertly transmitted. When data is sent the source and sending ports are always randomized and the SYN flag is set.

To start off with, I make use of the TOS field within the IP header to authenticate packets. If the TOS field in the IP header is set to the character ‘c’ then this means that it is a packet from the attacker machine containing command data. If the client receives a packet and the TOS field is set to the character ‘o’ then this is a packet from the server containing data that is the output of the regular command that the client has previously sent. If the client receives a packet with the TOS field set to ‘k’ then this is a packet from the server containing keylogger data. If the client receives a packet with the TOS field set to ‘x’ then this is a packet from the server containing exfiltration file data. And finally, if the client receives a packet with the TOS field set to ‘d’ then this is a file being sent from the server that created a CREATE event in the directory that the server is monitoring (directory watch).

Considerations

One weak aspect of my program is that is only sends 1 byte of data per packet. A better way to do this would be to create my own bit character encoding system (like how ASCII) but I would only include characters that I feel are important and I would leave out fluff characters that I would not be using. This way I could represent a character with less bits, therefore I could better maximize the bits I have to work with within the IP/TCP header fields.

Another weak aspect of my program is that the covert TCP traffic looks like a port scan since my program sends SYN packets to random ports. A better way to do this would be to create TCP traffic that simulates a real conversion or use a different protocol. ARP or DNS would be good protocols to use and ARP and DNS packets are expected to be sent and received within a network. If I was to encode command data within an ARP packet and send an ARP packets

every X minutes this would not look suspicious as machines are sending ARP request and replies all the time (same goes for DNS).

When trying to detect covert activity something to look for would be packets with incorrect checksums, request packets that don't receive a response (TCP SYN or ICMP echo request), TCP/UDP traffic with no data in the payload, port scanning traffic, traffic destined for arbitrary high numbered ports, TCP 3-way handshakes that are destined for your computer or originated from your computer that isn't web traffic. If any suspicious traffic is encountered, it would be best to simply drop all packets to and from the suspicious IP in the firewall. It's also a good idea have as little ports open as possible.