|  |
| --- |
| ﻿Extracting Network Traffic: |
|  | Network traffic is in the form of packets captured using Network Capture tools. Using Wireshark, a Network Analysis tool, we can extract all relevant details in packets. We converted the unstructured data in the packet captures to structured data by adding columns for specific fields : |
|  | \* TCP Flags and IP Flags- These flags are used for identifying the start and end of flows using SYN , RST and FIN flags. |
|  | \* TCP Sequence Number- These are used to identify the reconnections in a flow. |
|  | \* TCP Segment Lengths & UDP length - These are used to identify null packets. |
|  | \* Number of DNS Answer Records - These records are used to identify suspicious IRC communications. |
|  | \* Time To Live |
|  | \* Source Port |
|  | \* Destination Port |
|  | Finally, the pcap file is converted into a csv for analysis. |
|  |  |
|  |  |
|  | Flow Identification: |
|  |  |
|  |  |
|  | Flow Definition: |
|  | For the purposes of our framework, we define a flow as a collection of packets being exchanged between two unique IP addresses using a pair of ports and utilizing one of several Layer 4 protocols. |
|  | A flow is a conceptual structure that aggregates and summarizes all the packets sharing the following five fields: source IP address, source port, destination IP address, destination port and protocol. By working with flows we have all the information about the packets (except their payload) in a more condensed and meaningful structure. |
|  | The flows can be unidirectional or bidirectional. A bidirectional flow between Host A and Host B includes packets from both A->B and B->A. A unidirectional flow has only packets from A->B. We have applied bidirectional flow based analysis. |
|  | Flow Generation: |
|  | The Flow generation process is performed using our own Flow Generator written in Python, which can be used to identify both Unidirectional and Bidirectional flows. |
|  | We chose to write our own custom coded Flow Generator rather than using existing Flow and feature Generators such as Argus, Tranalyzer, Netflow and the latest being ISCX Flow Meter provided by ISCX itself, so as to gain greater flexibility in terms of Flow generation process (Active/Passive timeouts, RST Flag and FIN Flag) and Feature generation process per flow. |
|  | We have compared the resulting flows obtained by our tool with ISCX FlowMeter so as to establish the correctness of our Custom coded Flow Generator. |
|  | The ISCX Flow meter reported 3,72,000 flows on Train data, with each flow having more than exchanged. Our custom coded Flow Generator reported 4,20,000 flows on the Train data. However the difference of 48k is due to the fact that ISCX Flow Meter filters only TCP and UDP flows and discards other protocol packets. However our Custom Flow Generator considers all the flows in the process so as to provide more essence of the Background Traffic. |
|  | The working of our custom Flow Generator is as follows: |
|  | 1. Set Passive Timeout value, typically 3600 seconds |
|  | 2. Add feature FlowID as concatenation of Source IP, Destination IP, Source Port, Destination Port |
|  | 3. Sort all the packets based on their [Source, Destination, Source Port, Destination Port, Protocol, Time] using stable sort so as to group together the “possible” flows |
|  | 4. Iterate over Data and create a new flow for each new FlowID or Passive Timeout of current flow or checking for TCP flag viz RST and FIN (If in case). |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | Feature Definitions: |
|  |  |
|  |  |
|  | 1. Source and Destination IP: |
|  | In botnet detection context, their primary value comes from a possibility to quantify number of distinct connections. While as a feature it does not provide a definitive |
|  | conclusion, it is complementary in assessment of botnet communication patterns. |
|  | For example, in general the intervals between legitimate connections to distinct IP addresses are longer than the intervals in botnet communication [26]. |
|  | Ratio of Incoming to Outgoing traffic also calculated using IP addresses. |
|  | In practice however, a simple blacklisting of known IP addresses of C&C controllers have been widely employed. |
|  | Also, using IP addresses as features in classification results in overfitting the train dataset, resulting in unreliable evaluation. This is because the Bots can easily change their addresses, so in practice this feature will fail to detect them, easiest to evade. |
|  |  |
|  |  |
|  | 2. Source and Destination Port: |
|  | due to periodic change of source-destination port combination, detection of |
|  | some botnets simply based on this pair is infeasible. |
|  | Use of non standard ports also raises suspicion, which can be exploited as a feature. |
|  |  |
|  |  |
|  | 3. Protocol: |
|  | It is used for filtering out non related traffic (e.g., non-IRC traffic in the centralized detectors) thus reducing the volume of flows requiring further processing. |
|  | Presence of a protocol in an environment where it is not expected to be also raises suspicion. |
|  | Eg – IRC traffic rare in enterprise networks |
|  |  |
|  |  |
|  | 4. Duration: |
|  | The difference in the start and end time of the Flow. It can be useful as Bots display similar communication patterns and their communication with Botmaster etc can be of similar durations. |
|  | As such, initial connection attempts in the majority of botnets are known to be unidirectional and very short followed by much longer communication sessions. |
|  | Some IRC Botnets can have very long chat sessions. |
|  |  |
|  |  |
|  | 5. First Packet Length: |
|  | First packet in the flow of a botnet is generally very similar as the Bot is initiating the connection, connecting to a Botmaster. |
|  | The first packet is shorter than the rest of the communication. |
|  |  |
|  |  |
|  | 6. Flow Size Features: |
|  | These features characterize the communication patterns of Bots with the consideration that the network traffic generated by the Bots should be similar and more uniform as compared with normal users. |
|  | This makes sense as the Bots are programmed malicious softwares and thus cannot be exactly random. So, the bots following the same algorithm (eg – for port scanning) will display similar behaviour in the network traffic. |
|  | \* Total Number of Bytes Exchanged: This is the sum of the packet lengths (in bytes) of the packets exchanged between two hosts in a single flow. |
|  |  |
|  |  |
|  | \* Average Packet Length: The average of packet lengths of all the packets exchanged between two hosts in a single flow. |
|  |  |
|  |  |
|  | \* DPL (The total number of subsets of packets of the same length over the total number of packets in the same flow): Bots tend to have a lot of packets with the same length while communicating with their Botmaster and also when launching an attack like DDOS or suspicious activity like port scanning. |
|  | Same Length Packets refers to the number of different sets of same length. So, if its ratio with the total number of Packets is low, then it means a lot of packets with the same length have been exchanged. |
|  | \* Standard deviation of payload packet length: The Standard Deviation of packet lengths of all the packets exchanged between two hosts in a single flow. |
|  | \* Number of packets exchanged in a flow |
|  |  |
|  |  |
|  | 7. Reconnect: |
|  | Number of times a Source tries to reconnect to the destination in a single flow. This can be useful in 2 ways: |
|  | \* A Bot may be carrying out some suspicious activity like port scanning and the destination may realise/detect this. As a result, it may terminate its connection because of some firewall/IDS and then the Bot will repeated try to reconnect. |
|  |  |
|  | \* The Bot may try to evade detection by reconnecting. If a Bot keeps reconnecting randomly, its flow based features being monitored will change and may not match with other Bots, making it difficult to detect it. |
|  |  |
|  |  |
|  | Both these reasons indicate a high number of reconnects as suspicious. Though, these reconnects can most generally be simple retransmissions as well for legitimate users due to poor connectivity. |
|  | A method to identify reconnects is using Sequence Numbers – if a source sends many packets with the same sequence number it is taken as a reconnection. |
|  |  |
|  |  |
|  | 8. Number of null packets exchanged: |
|  | Bots tend to have a higher number of packets exchanged having no data for the following reasons: |
|  | \* Bots sometimes need to check which of the bots or the C&C servers are still alive. |
|  | \* Hosts in decentralized botnets tend to exchange small packets to probe their peers. |
|  | \* IRC hosts exchange small chat packets with the C&C servers. |
|  |  |
|  |  |
|  | 9. Time Based Features: |
|  | \* Average Bits per second |
|  | \* Average Packets exchanged per second |
|  | \* Average Inter-arrival Time of Packets For a single flow, this is calculated as the average of the time elapsed in the exchange of a packet between the two hosts. |
|  |  |
|  |  |
|  |  |
|  |  |
|  | 10. IOPR : Ratio between the number of incoming packets over the number of outgoing packets. |
|  | 11. Number of DNS records returned: |
|  | If a source makes a lot of DNS queries and the number of answer records returned (normally less than 4) is unusually high, it indicates IRC communication. Alongwith it, if the communication is taking place over non standard port, it indicates the hosts are trying to hide their IRC traffic. |
|  | Thus, it may be a Bot communicating with its IRC Botmaster to check for live IRC servers. |