DAT detectors: uncovering TCP/IP covert channels by descriptive analytics

- Detectors based on Descriptive Analytics of Traffic(DAT) for revealing hidden TCP/IP covert channels in Passive Warden Scenario.
- DAT detectors are embedded in network Intrusion Detection System(IDS) to perform faster,lightweight analysis and labeling of flows in addition to IDS.
- Novelty of DAT is a first-level-security covert channel detection.
- Provides a level of suspicion and seperates incoming traffic.
- Covert Channels are classified based on flow sequences, entropy analysis and patterns according to Pattern Language Markup Language,etc. This paper includes statistical techniques and characteristics for detection.
- Detection methodologies in Passive Warden Scenario can be grouped into:
  1) Traffic irregularities – Packet Compliance Checking Phase
  2) Statistical Analysis – Multimodality, Autocorrelation, Descriptive Statistics
  3) Machine Learning – Inter-field Analysis(Offline Analysis)
- Comprehensive, lighweight and flexible.
- Covert Channel detection techniques can be classified as: Value to Symbol Correspondance, Value Range as Symbols, Container Fields, Timing Channels and Derivative Approaches.
- Classification of TCP/IP header fields according to their capacity to hide covert data is as: Tied Fields, Regular Fields and fixed fields.
- Analysis methods in this paper include:
  1) Multimodality – a) Distributions(Kernel density estimations) b) Symbols(Pareto Charts)
  2) Sum of Autocorrelation Coefficients
- Data Detectors include: Labeling Criteria, Packet Compliance Checking, Flow Transformation Matrix, Intra-field analysis, Inter-field analysis
- Detect either plain or encrypted text channels
- Challenges faced are: a) Noisy Channels b) Covert channels in address fields and bouncing covert channels c) Covert channels modeled according to field distribution properties d) Covert channels with encrypted messages.
- The datasets used in this paper are self generated which includes a)Normal traffic dataset containing real TCP traffic taken from the LBNL/ICSI enterprise Tracing Project labelled as “normal” b) A dataset with covert channels, combining binary, 4-bit and 8-bit symbol channels labelled as “covert”.
- The prototype implementation is done using blocks, features and fields.
- The DAT detector prototype was tested on a machine with the following characteristics: 8 x Intel(R) Core(TM) i7- 4770T CPU 2.50 GHz, 16 GB RAM, Ubuntu 12.04 LTS, kernel Ubuntu 3.13. A total of 10,973,448 packets corresponding to 1324 flows were processed and analyzed.
- TCP/IP field extraction using tshark(1hr 16mins 36s), flow-matrix generation and the inter-field analsysis processing using scripts written in Python(19mins 40s),intra-field analysis using python(220ms) were observed.
- Cases identified as false positives revealed that they were mostly triggered by fields defined as Free. All covert channels were detected and no false negative recorded.
- Given flow vector representations, covert channel can be captured in the form of patterns.
- DAT detectors exhibit a high flexibility and allow the easy incorporation of new traffic features for future detection methodologies.

Code Layering for the Detection of Network Covert Channels in Agentless Systems

- This paper investigates a framework leveraging the extended Berkeley Packet Filter(eBPF) to create ad-hoc security layers in virtualized architecture without the need of embedding additional agents.
- The growing interest in “cloud-native” solutions pushes the evolution from Physical Network Functions to Virtual Network Functions(VNFs) and Container Network Functions(CNFs).
- Goes in the direction of agentless systems to address social security threats.
- Usage of Code Layering to instrumnet VNF/CNF entities with monitoring and inspe tion capabilities to build netowrk functions and gain network insights.
- Expolited by attacker to exchange data between host and Command & Control server avoiding blockages.
- Run rich set of eBPF programs and collects condensed statistics on header fields and timings.
- Code layering is a technique that stratifies the software into a number of functional layers, which can be modified in an independent manner(no rebuilding).
- It included Reference Layered Architecture having 3 layers:
  1) Inspection Layer
  2) Management Layer
  3) Detection Layer
- Two majore classes of network covert channels include: 1) Storage Channels 2) Timing Channels
- Storage covert channels affecting Traffic Class, Hop Limit and Flow Label are considered.
- Tests ran on virtual machines running Debian GNU/Linux 10(1 core 4GB RAM), intermediate nodes ran a modified version of Zeek along with libpcap and user-space tools with hosts having Ubuntu 20.4,32GB RAM,3.60 GHz Intel i9-9900KF CPU.
- Used traffic collected on an OC192 link in different periods made available by CAIDA.
- For timings channels, iPerf3 was used to generate ad-hoc flows.
- Pidstat and nProbe Enterprise is used.
- Detection of Storage Covert Channels include:
  1) Detection of Channels Targeting the Flow Label
  2) Sensitivity Analysis – 3 attack scenarios – a)exfiltration attempt modeled via transmission of file requiring to taget 8500 Ipc6 packets.
     b)Different channels altering in time c) APT targeting datacenter or subnetwork.

  3) Channels Targetting other Ipv6 fields

- Compute a measure of variances built by grouping packets to make pattern-like behaviours to emerge.
- Measure CPU and memory usage to understand impact of in-kernel algorithm.
- Agentless approach does not introduce further delay or packet loss on the inspected traffic.
- Perfomance can be measured based on Impact of Packet Transmission and CPU and Memory usage.
- Impact of the packet size and transmission rate for UDP flows as well as the Maximum Segment Size (MSS) for TCP streams is considered.

- eBPF-based mechanism approaches for a small overhead with respect to baseline and does not affect packet transmission in any way.
- This framework exploits tools like Prometheus, Performance Co-Pilot, Vector as well as specific eBPF programs.
- Uses bin-based data structure to reveal hidden communications,prevents to store and process sensitive details
- Improvements can be made to consider different threats, utilization of eBPF for actively manipulating traffic(sanitize flows)

ARPNetSteg: Network Steganography Using Address Resolution Protocol

- Steganography is a technique that allows hidden transfer of data using some media such as Image, Audio, Video, Network Protocol or a Document, without its existence getting noticed.
- Algorithm ARPNetSteg that implements Network Steganography using the Address resolution protocol.
- Can transfer 44 bits of covert data per ARP reply packet.
- Use of Address Resolution Protocol (ARP) Packet to carry secret message over the network.
- Technique solely implemented over a LAN
- Steganography techniques can be classified into three categories:
  1) Storage Based Techniques
  2) Time Based Techniques
  3) Hybrid Techniques
- Physical mapping between link layer and network layer done by ARP.
- ARP is stateless protocol
- The Address Resolution Protocol provides no mechanism for authenticating the source address of a machine sending an ARP response.
- Uses intra-protocol steganography (Protocol steganography technique that uses a single network protocol) using ARP protocol that partially uses the concept of ARP spoofing to transfer covert data over a local area network.
- The technique runs two algorithms: one at Sender side and one ar Receiver side.
- Encode Covert message string to Hexadecimal Code and scan LAN for free or unallocated local IP addresses.
- Generate random IP addresses till we get unallocated local IP address and then covert message sender waits for ARP broadcast request from receiver for this local IP address.
- Puts First 11 or lesser hexadeciml digits of the covert message in the Sender Hardware Address field of ARP Reply message.
- Last hexadecimal digit of Ethernet Address is used to store Control Information.
- Sender side algorithm includes generation of random numbers to successfully sending ARP reply.
- Receiver and sender enter same seed value known in prior
- Receiver enters a wait state waiting for spoofed ARP reply from Sender
- Random IP generation and processing of ARP reply is done till Control quad in received ARP reply is non zero.
- Scapy with Python was used to create and send ARP packets.
- Covert message may or may not use all of the 44 bits.
- If control quad is 0x0, sender wants to send more data and all last 11 bits were used.
- If control quad value is 0xf, it is last message with 0 padding otherwise it has padding.
- Wireshark is used to capture packets coming in and going out
- Accuracy increased largely with 1 retry(ARP packets sent again), but with further increase, increase in accuracy was minimal.