

SYSTEM AND METHOD FOR EXTRACTING AND PRESERVING METADATA FOR ANALYZING NETWORK COMMUNICATIONS

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] This application claims the benefit of co-pending U.S. Provisional Patent Application 61/784,931 filed March 14, 2013, which is hereby incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The technical field relates generally to network communications, and more particularly to preserving network communications for network security and forensics purposes.

BACKGROUND

[0003] Network communications within and between organizations transfer a variety of data and vital information. However network communications are also one of the main ways attackers get inside the security perimeter of a network. Modern network security technology deploys many methods to prevent the attackers from causing harm to the organizations and individuals. None of them are perfect, especially today in the era of Advanced Persistent Threat (APT). Modern APTs can be developed and deployed targeting very specific targets and usually such APTs can be very hard to detect and eliminate. Recording the network traffic is one of the common methodologies in the advanced network defense. It is analogous to the security camera - while not optimized for the detection of the threat, it is very useful in the elimination stage, allowing a close examination of the threat-related subset of the traffic with the purpose of determining the root cause and infected network nodes.

[0004] The network traffic is ever increasing and that creates a challenge for a traditional network recording methods. For APT sometimes taking around months to fully penetrate the organization, a typical requirement for the network data storage is at least 6 months. For a large enterprise it is usually either impossible or impractical to collect and store the resulting amount of data. Even if the APT infection happens, the traffic caused by this attack is

miniscule in comparison with the amount of data the enterprise is forced to store in order to research that attack.

[0005] The current invention offers a methodology allowing significantly reducing the amount of data that needs to be stored while keeping all the essential information intact. Furthermore, other desirable features and characteristics of the present invention will become apparent from the subsequent detailed description and the appended claims, taken in conjunction with the accompanying drawings and the foregoing technical field and background.

SUMMARY

[0006] A method for advanced persistent threat detection on a network is provided. The method includes capturing data packets from a network and performing layered session decoding on the captured packets. Metadata is

[0007] extracted from the decoded packets and is stored for analysis. Analysis of the metadata is used to detect advanced persistent threats on the network.

[0008] A system for advanced persistent threat detection on a network is provided. The system includes a network and a processor coupled to the network. The processor is configured to capture data packets from the network and perform layered session decoding on the captured packets. Metadata is extracted by the processor and stored in a memory coupled to the processor. The metadata may then be analyzed to detect advanced persistent threats on the network.

DESCRIPTION OF THE DRAWINGS

[0009] The present invention will hereinafter be described in conjunction with the following drawing figures, wherein like numerals denote like elements, and

[0010] FIG. 1 is simplified block diagram of a system for preserving network communications in accordance with exemplary embodiments;

[0011] FIG. 2 illustrates the capturing process in accordance with exemplary embodiments;

[0012] FIG. 3 illustrates the packet processing in accordance with exemplary embodiments;

[0013] FIG. 4 illustrates the session re-assembly in accordance with exemplary embodiments;

[0014] FIG. 5 illustrates the Layered Session Decoding in accordance with exemplary embodiments; and

[0015] FIG. 6 illustrates the Metadata extraction and storage in accordance with exemplary embodiments.

DETAILED DESCRIPTION

[0016] The following detailed description is merely exemplary in nature and is not intended to limit the subject matter of the disclosure or its uses. Furthermore, there is no intention to be bound by any expressed or implied theory presented in the preceding technical field, background, brief summary or the following detailed description.

[0017] In this document, relational terms such as first and second, and the like may be used solely to distinguish one entity or action from another entity or action without necessarily requiring or implying any actual such relationship or order between such entities or actions. Numerical ordinals such as "first," "second," "third," etc. simply denote different singles of a plurality and do not imply any order or sequence unless specifically defined by the claim language.

[0018] Additionally, the following description refers to elements or features being "connected" or "coupled" together. As used herein, "connected" may refer to one element/feature being directly joined to (or directly communicating with) another element/feature, and not necessarily mechanically. Likewise, "coupled" may refer to one element/feature being directly or indirectly joined to (or directly or indirectly communicating with) another element/feature, and not necessarily mechanically. However, it should be understood that, although two elements may be described below, in one embodiment, as being "connected," in alternative embodiments similar elements may be "coupled," and vice versa. Thus, although the schematic diagrams shown herein depict example arrangements of elements, additional intervening elements, devices, features, or components may be present in an actual embodiment.

[0019] Some of the embodiments and implementations are described above in terms of functional and/or logical block components and various processing steps. However, it should

be appreciated that such block components may be realized by any number of hardware, software, and/or firmware components configured to perform the specified functions. For example, an embodiment of a system or a component may employ various integrated circuit components, e.g., memory elements, digital signal processing elements, logic elements, look-up tables, or the like, which may carry out a variety of functions under the control of one or more microprocessors or other control devices. In addition, those skilled in the art will appreciate that embodiments described herein are merely exemplary implementations.

[0020] Finally, for the sake of brevity, conventional techniques and components related to information security and other functional aspects of the system (and the individual operating components of the system) may not be described in detail herein. Furthermore, the connecting lines shown in the various figures contained herein are intended to represent example functional relationships and/or physical couplings between the various elements. It should be noted that many alternative or additional functional relationships or physical connections may be present in an embodiment of the invention. It should also be understood that FIGS. 1-6 are merely illustrative and may not be drawn to scale.

[0021] Over the last several years, organizations have accumulated significant park of automated network appliances in attempt to protect their networks. At the same time, the amount of successful APT attacks is constantly on the rise. There's no paradox here – as much as network security vendors are trying to create automated methods for APT detection, they essentially can only deal with attack vectors and methodologies that have already been exposed and researched. Individuals behind the APT attacks, on the other hand, have all the tactical advantages of a first mover.

[0022] After the attack is successful, sooner or later it will be exposed – either by its direct actions or by side effects. The goal of the defenders is to minimize the reaction time, i.e., time required to:

[0023] Mitigate the infection;

[0024] Come up with counter – measures so the infection won't repeat; and

[0025] Recover the damage.

[0026] Both steps a) and b) usually require significant efforts from highly-trained experts. Practice shows that recordings of existing traffic can significantly reduce the time required to research the APT behavior. As mentioned before, typically partial recordings from existing

automated systems are not enough – the reason the APT has infected the network is because existing appliances didn't detect it in the first place.

[0027] The foregoing means that in order to rapidly and efficiently research the network infection - ALL the traffic that happened on the network since the APT became active on the network is required for initial assessment. Assuming that initial "latent" period of the APT can take months or even years this requirement creates a significant burden for most organizations – while human experts can be hired only for the time of investigation and mitigations, the full network recording capabilities should be on premises and operational all the time.

[0028] The amount of data that needs to be recorded and indexed in the brute-force approach is enormous. In addition to pure technical challenges it creates additional security and privacy issues in and of itself. If all the traffic is conveniently stored in a large disk array, it presents a target for attackers to break into this storage and steal the data from there.

[0029] In other words, what experts see in the current network security tendencies is that there is a shift from storing not enough evidence to storing everything (i.e., overreacting and storing too much data).

[0030] The disclosed method stores a sufficient amount of network metadata to significantly improve the incident forensics/response efficiency while keeping the storage manageable. In order to achieve maximum performance it is important to do all the necessary processing, data compression and metadata extraction in-memory, i.e., without using the hard disks as intermediary data storage. The disks, in the indexed database form, should also be used only for a final storage of the indexed network metadata.

[0031] One embodiment of the present invention provides a configurable method of storing just enough network metadata to make a post-infection research as efficient as possible without deploying significant monetary and storage resources and introducing additional security and privacy risks.

[0032] The method may reassemble complete client-server conversation streams, apply decoders and/or decompressors, and/or analyze the resulting data stream using one or more content scanners. The method may also selectively perform deep packet inspection dealing with individual network packets. The method may further provide one or more layers of content decoding that may "peel off," for example, common compression, aggregation, file

formats and/or encoding schemas and may extract the actual content in a form suitable for processing and metadata collection. In addition, the decoders may uncover hidden transport mechanisms such as, for example, IPv6 tunnels or e-mail attachments. The method may operate on real-time network traffic (e.g., including 10 Gbps networks) and may allow, for example, the collection of the several network metadata streams to one final database for cross-reference research. The method may also be deployed next to complete and partial full-packet capture solutions for the purposes of providing indexed access for particular elements of the raw traffic.

[0033] The method may be installed, for example, on off-the-shelf Linux Operating System ("OS") and Intel-based hardware, and may allow the appliance to function as a standalone network appliance. The method may also utilize one or more Network Interface Cards ("NICs") for packet capturing. The method may also connect to other sources of information, like e-mail and HTTP proxy servers for example. Multiple instances of the method may also run on the appliance.

[0034] A machine-readable medium (e.g., CD) may be programmed with the method, for example, to be installed on any Linux running on PC hardware with 64-bit Intel CPU.

[0035] FIG. 1 illustrates one embodiment of a system in accordance with the invention. As can be seen, the system 100 includes several modules. According to fundamental embodiments of the invention, packets are captured (102) from a network. The packet types are determined (104) and the packets are decoded in block 106. The system may be suitable for a variety of applications, for example, accessing all layers of network traffic including the content of TCP/IP and UDP data exchanges. Packet type information and the decoded packets are sent to block 108 for session reassembly. Next, layered session decoding is performed in block 110. Block 110 may also accept captured (112) data from SMTP mail hub, or Web proxy servers. Metadata is extracted in block 114 in accordance with a configuration profile (116) using the layered session decoding information and layered packet decoding information. This metadata is collected via a metadata receiver 118 which is coupled to a data base 120 for storing the metadata. The stored metadata can then be analyzed via a graphical user interface 122. The operator may analyze the metadata manually or with the assistance of various automated processes 124 to detect advanced persistent threats to the network.

[0036] The system 100 digs deep into every network exchange and decodes application protocols and file formats while they are in progress or from off-line source. Conventionally, when the decoding is finished the content of the data itself is usually discarded. All this done in-memory without Disk I/O operations being involved during the whole metadata extraction phase. The system 100 typically stores (120) the metadata information about each network exchange. In addition to the common session attributes like IP addresses and ports, the system is capable of extracting all the application protocols used, IP tunneling if used, user names, file names, HTTP referrers and request URLs, server return codes and so on, as specified in the user configuration. Non-session based protocols like DNS are decoded on per-packet basis and the metadata is sent for storage similar to the session-based protocols.

[0037] Since typical network traffic consists of long sessions and file transfers, per-session metadata storage without real content constitutes at least 1/100 compression ratio vs. the raw network traffic and significantly reduces security and privacy risk in addition to compliance requirements usually associated with full data storage. On the other hand, storing the wealth of information about the session itself gives a trained network forensics specialist everything needed to research the network attack in all stages of progress.

The processing steps are:

- 1) Capture the raw network traffic;
- 2) Optionally Selectively decode packets (IPv6 tunnels, DNS, etc);
- 3) Optionally Reassemble sessions from TCP and UDP protocols;
- 4) Perform the Layered Session Decoding; and
- 5) Extract the Session metadata in accordance with the Configuration.

[0038] FIG. 2 illustrates the capturing process in accordance with exemplary embodiments. The purpose of this step is to implement various interfaces with live and off-line data sources with the goal of providing a uniform data representation inside the system.

[0039] Different capture modules may be implemented depending on the source of the network traffic. The most common sources are the raw network traffic coming from the Network Switch Span port 200 or the network tap 202; HTTP traffic coming via the ICAP connection from the HTTP Proxy appliance 204 and SMTP traffic coming via SMTP mail hub 206 implementation.

[0040] More than one capture method may be used in the same in-memory processing system. Additional capture modules may be also implemented in the future for specific network exchange protocols and off-line storage formats.

[0041] Optional Selective packet decoding

[0042] FIG. 3 illustrates the packet processing in accordance with exemplary embodiments. Certain very important transmissions need to be analyzed on the packet level before attempts are made to assemble them into meaningful sessions for the forensics research. For example, IPv6 is very often being used by APT to obfuscate the network transmissions. Some network exchanges are session-less and should be processed as such. A typical example of this being the DNS protocol. Certain classes of APT and malware rely on DNS for downloading additional software after the initial infection. Accordingly, block 300 determines if special processing is required. If not, the packet is sent directly to session reassembly 108. However, if special processing is required, the layered packet decoding operation 106 is used and block 302 determines if an IP Tunnel was utilized to add tunnel information for session metadata extraction 114, and then routes the packet to session reassembly 108.

[0043] The system accommodates those requirements by extracting the important data from DNS requests directly and detecting and decoding the IPv6 tunnels. Obviously this type of processing cannot be done for the session-oriented data exchanges, like HTTP or SMTP.

[0044] Optional Reassembly Sessions from TCP and UDP protocols

[0045] FIG. 4 illustrates the session re-assembly 108 in accordance with exemplary embodiments. When the system captures raw network packets 400, whether directly from the network or from the off-line storage, it compiles streams of singular packets into the logical sessions featuring common characteristics to minimize the amount of the similar information that needs to be stored. For example if the network session consists of 100 packets 1.5 KB each arranged into the TCP/IP file transfer via FTP, the resulting extracted information may consist of the source and destination IP addresses and ports, total length of the session, username/password used, file name, type and length, time of day and date. Those characteristics are common for all the packets involved in the session, resulting in significant compression of information in comparison to recording all the raw packets with their payload.

[0046] The system deploys a TCP/UDP reassembly module that keeps track of all the current network streams and adds singular packets from the network together with their payloads into the corresponding streams for further content decoding and metadata extraction. All this processing may be done in memory to be able to perform on live network speeds. The module processes packets one by one from the one or more input queues, each queue representing, for example, a network interface in the promiscuous mode or PCAP file with the stored packet stream, finds an appropriate network stream represented by a descriptor and data area in memory, extracts the data from the packet and put it into the memory data 402 offset and updates the stream descriptor accordingly. Session streams are stored in memory 404. The module may also decide that the particular network stream is closed – based, for example, on processing a specially formatted network packet or the stream condition in general, for example a timeout. In this case the TCP/UDP reassembly module ensures that the stream and its data is fully decoded and erases it from the memory. The data may be erased before the session descriptor, taking care that stray packets and, for example TCP retransmits arriving late and out of order are properly recognized as such in accordance with TCP RFC states. The stream data in memory consists of one or more linked memory “chunks” of fixed size similar to the typical disk file system. New chunks for the data stream are allocated as transmission progresses.

[0047] Layered session decoding

[0048] FIG. 5 illustrates the layered session decoding 106 in accordance with exemplary embodiments. Although the very primitive initial metadata is available immediately from the packet information (IP addresses, TCP and UDP ports and similar), an advantage of this is provided when metadata is extracted from the all the data being transferred. In particular, when the previous module (the session re-assembly 108 or the session capture module 112 (as shown on FIG. 1) decides to send a reference for particular session for analysis. The main decoder loop handler relates the session descriptor and the data stream and deploys series of decoders in the loop. The decoding loop happens until none of the decoders can recognize the data anymore. This event triggers the end of the decoding loop. There’s logical separation between Application Protocol Decoders 500 and Format decoders 502– mostly because of internal data representation differences. The Protocol Decoders 500 can analyze a client and server stream simultaneously in order to properly process the communication. The Format

Decoder 502 typically deals with single buffers extracted from the client-server communication by the Protocol Decoders 500. Such loop structure guarantees that all the data would be properly decoded, even if it was, for example, double-compressed.

[0049] Each decoding cycle produces a protocol or format – specific metadata that gets added to the session description. All the processing may be done in memory to facilitate top performance.

[0050] Extract the Session metadata in accordance with the Configuration

[0051] FIG. 6 illustrates the Metadata extraction and storage in accordance with exemplary embodiments. The metadata may be extracted using multiple sensors 600 as shown. The metadata collected on the previous step is being matched to the user-provided configuration. The entries that are of interest of the user (for example format names, file names and user names but not full HTTP headers) are being converted to key-value pairs and sent to the database for storage.

[0052] The Database 120 may be local or remote. In case of the remote database the SSL encryption over the proprietary protocol is being deployed to safely and reliably transfer the data to the destination.

[0053] More than one metadata collection system 602 can be deployed on the network, storing the results of the processing in the same database 120. Due to effective 100:1 (or more) compression, the system can be successfully deployed on high-speed network in many-to-one configuration, that's it – multiple Collectors sending the network metadata to the same database for centralized storage and correlation.

[0054] While at least one exemplary embodiment has been presented in the foregoing detailed description, it should be appreciated that a vast number of variations exist. It should also be appreciated that the exemplary embodiment or exemplary embodiments are only examples, and are not intended to limit the scope, applicability, or configuration of the disclosure in any way. Rather, the foregoing detailed description will provide those skilled in the art with a convenient road map for implementing the exemplary embodiment or exemplary embodiments. It should be understood that various changes can be made in the function and arrangement of elements without departing from the scope of the disclosure as set forth herein.

CLAIMS

What is claimed is:

1. A method, comprising:
capturing data packets from a network via a processor to provide captured packets;
performing layered session decoding on the captured packets to provide decoded packets;
extracting metadata from the decoded packets;
storing the metadata; and
analyzing the metadata to detect advanced persistent threats on the network.
2. The method of claim 1, further comprising the processor determining a packet type for the captured packets.
3. The method of claim 1, further comprising the processor reassembling a session from the decoded packets.
4. The method of claim 1, further comprising the processor capturing data from a network proxy server.
5. The method of claim 1, further comprising the processor capturing data from an SMTP hub.
6. The method of claim 1, further comprising the processor extracting the metadata in accordance with a configuration.
7. The method of claim 1, further comprising the processor utilizing application protocol decoders when performing the layered session decoding.
8. The method of claim 1, further comprising the processor utilizing format decoders when performing the layered session decoding.

9. A system, comprising:
a network;
a processor coupled to the network and configured to;
capture data packets from the network to provide captured packets;
perform layered session decoding on the captured packets to provide decoded packets; and
extract metadata from the decoded packets;
a memory coupled to the processor for storing the metadata; and
wherein, the metadata may be analyzed to detect advanced persistent threats on the network.
10. The system of claim 9, wherein the processor is further configured to determine a packet type for the captured packets.
11. The system of claim 9, wherein the processor is further configured to reassemble a session from the decoded packets.
12. The system of claim 9, wherein the processor is further configured to capturing data from a network proxy server of the network.
13. The system of claim 9, wherein the processor is further configured to capturing data from an SMTP hub of the network.
14. The system of claim 9, further comprising the processor extracting the metadata in accordance with a configuration.
15. The system of claim 9, further comprising application protocol decoders for performing the layered session decoding.

16. The system of claim 9, further comprising format decoders for performing the layered session decoding.

17. A non-transitory computer readable medium embodying a computer program product, said computer program product comprising:

an advanced persistent threat detection program configured to:

capture data packets from a network via a processor to provide captured packets;
perform layered session decoding on the captured packets to provide decoded packets;

extract metadata from the decoded packets; and

analyze the metadata to detect the advanced persistent threats on the network.

18. The non-transitory computer readable medium embodying a computer program product of claim 17, wherein the advanced persistent threat detection program is further configured to determine a packet type for the captured packets.

19. The non-transitory computer readable medium embodying a computer program product of claim 17, wherein the advanced persistent threat detection program is further configured to reassemble a session from the decoded packets.

20. The non-transitory computer readable medium embodying a computer program product of claim 17, wherein the advanced persistent threat detection program is further configured to extract the metadata in accordance with a configuration.