Automated Defenses against Advanced Persistent Threats

Robert Charles Metzger

April 17, 2017

Recently, there have been numerous research papers proposing automated defenses against APT's. There are several questions that each of these efforts needs to address:

1. What is the raw data that will be used to detect APT activity?
2. Why will it identify APT activity?
3. How will this data be put into a canonical form?
4. How will this data be structured for analysis?
5. How will the knowledge base be constructed, or the machine learning performed?
6. What algorithms will be applied to the knowledge base and the input data to detect APT activity?
7. How will this handle previously unknown APT features (zero-day exploits)?
8. How will this handle intermittent activity over a long period of time across multiple entities?

The papers reviewed here discuss work that has at least been partially implemented, or is based on an implementation of a related work. There are 3 times as many papers that contain proposals for work that has not been implemented, and in most cases, show no evidence of a detailed design.

The research efforts surveyed for this paper used a great variety of raw data as input to the process of detecting APT's. They can be grouped as follows:

1. Dynamic information generated by the operating system, such as authorization logs and performance statistics,
2. Dynamic information generated by the network, such as DNS traffic and packet sizes and counts,
3. Dynamic information generated by cyber-security software, such as event logs generated by NIDS and HIDS,
4. Miscellaneous dynamic information, such as counts of file and Emails with attachments,
5. Miscellaneous static information, such as static analysis of application protocols or public databases of attacks.

Not surprisingly, all of those various inputs were built into a wide variety of data structures.

1. Directed and undirected graphs,
2. Relational and textual databases,
3. Miscellaneous proprietary data structures.

Just as the data varied widely, the researchers used a variety of algorithms for detecting APT's. They fall into the following groups.

1. Supervised machine learning, such as a Random Forest model,
2. Unsupervised machine learning, such as Association Rule Mining,
3. AI systems performing symbolic reasoning
4. Traditional graph algorithms, such as network flows and connected components,
5. Statistical tests, such as Hidden Markov models and sketch-based measurement,
6. Miscellaneous proprietary algorithms.

The most common way of handling the intermittent character of APT activity was to build data structures which were associative in character. By this we mean that data items were related semantically, such as with a directed graph, rather than chronologically. Building these data structures circumvented the sequential, chronological character of the raw input data.

The most common way of handling previously unknown APT features was to use raw inputs which reflected some inherent characteristic of APT attack vector (such as network traffic), rather than to match the actual behavior of an attack vector.

1. R. Abreu, D. G. Bobrow, H. Eldardiry, A. Feldman, J. Hanley, J., T. Honda, and D. Burke, "Diagnosing Advanced Persistent Threats: A Position Paper," in Proceedings of the 26th International Workshop on Principles of Diagnosis, 2015, pp. 193-200.

The authors view the problem of detecting APT's as a problem of diagnosis. They propose using "audit trails" of system events as the raw input to their process. These audit trails can either be existing log files or custom logs generated by modifying applications to generate information of interest.

Their approach begins with an "Activity Classifier," which annotates audit trails with semantic tags in pseudo-real-time. These tags are assigned by mapping a log entry to a higher-order activity that it corresponds to. The mapping system takes a supervised machine learning approach, using a set of activities and a collection of patterns that describe these activities. This Knowledge Base will be defined manually.

The system provides two ranking approaches. The supervised ranker assigns a threat value based on previously discovered threats. It has both a hand-coded classifier, and its tags are given highest priority. It also has a classifier which learns from training data. The unsupervised ranker ranks activities according to statistical normalcy. It is meant to handle previously unknown threats.

The proposed system uses the output of ranking to feed a model-based diagnosis engine. This needs the machine learning module to group items in the audit trails both temporally and spatially. This is how they propose to handle intermittent long-term activities. The engine uses probabilities to generate multiple possible diagnoses, which are ordered by likelihood.

The group did not have a knowledge base or working system at the time the paper was published. This work is of interest because it takes a reasoning-based approach to identifying APT's.

2. P. Bhatt, P., E. T. Yano, E. T. and P. Gustavsson, "Towards a Framework to Detect Multi-stage Advanced Persistent Threats Attacks," *in IEEE 8th International Symposium on Service Oriented System Engineering,* pp. 390-395, 2014.

The authors organize their framework around a multi-stage attack model, based on the Intrusion Kill Chain (IKC). They believe that a layered security architecture, which has appropriate sensors for detecting activities at each stage of the IKC, is necessary.

The raw input used by the system comes from a variety of logs: HIDS, NIDS, firewalls, Web servers, Email servers, etc. The logs used by the system are normalized so that they contain the originating control, timestamp, attack type, network source and destination. The normalized data from different logs is correlated by the timestamps.

The authors assert that "The automatic event correlation to detect APT's is a research issue. " They say they intend to investigate "Hidden Markov Models and other probabilistic techniques" (sic) for this purpose.

The authors implemented the parts of their system which normalize and analyze logs. Their work is of interest because the partial implementation used a "Big Data" approach which was implemented on a Hadoop cluster.

3. G. Brogi and V. V. Tong, "TerminAPTor: Highlighting Advanced Persistent Threats through Information Flow Tracking," in *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pp. 1-5, 2016.

The authors view detecting APT's as the problem of identifying a sequence of elementary attacks. These elementary attacks can be detected by an IDS. The project is built on the assumption that if two attacks are consecutive steps in a chain, then there must be information flows from the first event outputs to the second event inputs.

The proposed system receives input from IDS running on the systems to be protected. The IDS output is treated as a sequence of events in chronological order. Some events can be treated as information flows. Events are processed into tuples containing event type, timestamp, input object list, and output object list.

Tags are attached to the input and output object lists and are propagated through the representation of the information flows. As the tags are propagated, attack chains may be created or merged.

The authors do not mention the problem of zero-day exploits, which would seem to be a significant issue for using existing IDS. Their propagation method should overcome the issue of intermittent activity over long time period.

The authors implemented their system, and the paper presents experimental results using a mixture of simulated and actual log data for a two-month span. Their work is of interest because it suggests a straightforward way of leveraging existing IDS log data for APT detection.

4. S. Chandran, P. Hrudya, and P. Poornachandran, "An efficient classification model for detecting advance persistent threat," In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on,* pp. 2001-2009, 2015.

The authors assert that common system performance metrics and file counts can be used to identify APT's. They claim that when APT attack steps are active, a system will use more CPU time and memory than is normal.

The raw input to the system comes from sampling system performance metrics (CPU usage, memory usage, open ports) and file counts (system32 folder). Each sample of values is treated as a vector.

The authors use the Random Forest model, which is a supervised ensemble machine learning technique. The random forest algorithm uses random decision trees with bagging to get better accuracy in classification. The trees vote and the winner predicts whether the class is `malicious' or `benign'.

The input values collected should be independent of zero-day exploits. The sampling should address the problems of intermittent activity.

The authors have implemented their system, and the paper presents experimental results from monitoring a collection of Wintel systems over three months. Their work is of interest because it uses system performance metrics not usually associated with detecting malware.

5. I. Friedberg, F. Skopik, G. Settanni and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Computers & Security*, vol. 48, pp. 35-57, 2015.

The authors take a whitelist approach to identifying system behavior. Their system can use any type of log files generated by HIDS or NIDS, since they take a string-oriented, as opposed to semantic, view of the log content. The system detects anomalies that result from realistic APT attacks.

The raw input to the system is the set of logs generated by whatever cyber security software is in use. The model is based on the behavior of the particular components in the unique monitored environment. The syntax and semantics of the log lines are irrelevant, and since they are abstracted, the information can be shared with other organizations with no loss of privacy.

The system model has four components:
1) Search patterns, which are substrings of processed log lines,
2) Event classes, which classify log lines according to known patterns,
3) Hypotheses, which contain possible implications between log lines,
4) Rules, which are hypotheses that have been proven.

A unit of logging information, called a log atom, is a line, or a binary record, or an XML element from a log file. The system begins by collecting all of the log information and preparing it for evaluation. Each log atom gets a fingerprint, which is a binary vector indicates whether each known pattern is a substring of the log atom. The fingerprint is used to classify to which event classes the log atom belongs.

A hypothesis in this system is a correlation rule between events from two different event classes, which hasn't been validate yet. The system creates these hypotheses and tests them, to learn about dependencies between events. It evaluates a real-time stream of events against the available hypotheses.

The approach should detect APT attacks which use zero-day exploits, since it identifies the anomalies which are the by-products of those attacks. The pattern-oriented approach avoids the problems that intermittent activities can produce.

The authors have implemented their system and the paper presents a substantial body of experimental results. Their work is of interest because of its unique approach to analyzing system log information and because of the extensive experimental evaluation.

6. P. Giura, and W. Wang, "A Context-Based Detection Framework for Advanced Persistent Threats," *2012 International Conference on Cyber Security,* pp. 69-74, 2012.

The authors approach detecting APT's through a 3-D version of the "attack tree" concept, called an "attack pyramid." The attack pyramid has a plane for each of the stages of an APT: reconnaissance, delivery, exploitation, operation, data collection, exfiltration.

An attack tree has the target at the root, and intermediate steps to achieve the target are children. Two child nodes are connected with `AND' and `OR' nodes, depending on whether both events or either event must happen to move to the parent node. A detected APT looks like an attack tree, except that the path crosses multiple planes of the pyramid.

The raw input used by this system is all events recorded by automatic logging mechanisms, regardless of the threat level they pose in themselves. While the horizontal planes of the pyramid correspond to the stages of the APT, the vertical planes (which are the surfaces of the pyramid), are the areas where events are generated. The surfaces are the physical plane (events related to devices or locations), the user plane (events related to privileged users), the network plane (all events recorded by firewalls, routers, IDPS, etc.), and the application plane (events recorded by level 7 of the network hierarchy and host and server applications).

The system processes all of the events from the various surfaces and correlates them into contexts using correlation rules. The contexts are passed to the alert system, which applies detection rules for each context. It uses a signature database which identifies "known bad" contexts, and computes risk and confidence values for the context. If these exceed limits, then an alarm is triggered, and a human analyst takes over.

The system handles intermittent APT activities through the use of the attack pyramid. It is not clear how it handles zero-day exploits.

The authors have implemented part of their algorithm and the paper presents experimental results from executing it on several logs that contained one day's data. Their work is of interest because of their use of the attack pyramid model as the basis for a system design.

7. J. R. Johnson and E. A. Hogan, "A graph analytic metric for mitigating advanced persistent threat," In *Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on*, pp. 129-133, 2013.

The authors present a graph analytic metric that can measure the vulnerability of a computer network to attacks like APTs, which involve lateral movement between systems and escalation of privileges. The authors assert that there are two conditions which enable an APT. The first is that the attacker gets administer privileges on at least one machine in the network.

The raw data that their algorithm uses is authorization requests.

The algorithm begins with a simple connectivity graph for the computer network and builds a reachability graph. In this graph, edges only occur if there are login privileges from one system to another. The paper includes pseudo-code descriptions both for the generation of the reachability graph, and calculation of the gamma-V metric.

Once the reachability graph is constructed, the gamma-v metric can be computed each time an authorization request is received. The metric identifies the number of vertices in the reachability graph that are on a path to a specified node. If granting the request decreases the gamma-v, with respect to a node of special interest, the request can be denied.

Since the metric is only calculated on authorization events, and can be quickly recalculated, intermittent activity is irrelevant. Since only authorization events are considered, zero-day exploits are also irrelevant.

The group implemented their graph algorithm, and the paper presents experimental results on its performance. This work is of interest because of its unique use of authorization requests as input data, and because it defines and implements a new graph theoretical algorithm.

8. M. Lee and D. Lewis, D. "Clustering disparate attacks: mapping the activities of the advanced persistent threat," Last accessed June 26 (2013).

The authors assert exposure to APT's is related to exposure to malware distributed by email. They define targeted attacks as email containing malware that has a small number of selected recipients.

The authors use Emails containing malware as their raw data. They examine those Emails manually and screen out those which were sent in volume, or which weren't sent to individuals. The residual Emails are likely to have been targeted.

This input data is used to create a graph in which nodes represent Email addresses which have received a targeted attack, and edges connect nodes that were subject to the same attack. The algorithm repeatedly searches for clusters of recipients who have received the same Email, until all linked recipients have been identified.

The work does not address zero-day exploits, and these would seem to be irrelevant to the method. The analysis is done off-line, and so intermittent activity is compressed into the sampling window.

The authors have a prototype implementation which uses Perl and MySQL. The paper presents some experimental results. Their work is of interest because it addresses the targeted nature of APT's. Their work is limited in applicability since there are other methods for gaining initial access to a target network, such as phishing and drive-by downloads.

9. M. Marchetti, F. Pierazzi, A. Guido, and M. Colajanni, "Countering Advanced Persistent Threats through security intelligence and big data analytics," In *Cyber Conflict (CyCon), 8th International Conference on,* pp. 243-261, 2016

The authors approach detecting systems that have been compromised by APT's using internal information from network probes, and external information, such as Open Source Intelligence (OSINT), blacklists, etc. The purpose of the systems they describe, is to generate a list of internal systems, which are ranked by scores for exposure and compromise.

There are three classes of raw input for this system: 1) network logs collected by SIEM and NIDS systems, 2) lists of servers and clients, 3) Open Source Intelligence data.

The network logs are used to compute a compromise score for each client system based on compromise indicators. The analysis algorithms are aimed at ranking host systems which might be involved in three phases of an APT: maintaining access, lateral movement, and data exfiltration.

To identify hosts which are involved in maintaining APT access, the system uses network logs to build a bipartite graph of internal and external hosts. It designates as suspicious an external host if one of these questions is true: 1) Is it on a blacklist? 2) Is the domain name generated? 3) Are the access patterns regular? 4) Are there non-matching flows between an internal and external host?

To identify hosts which are involved in lateral movement, the system analyzes the number of internal hosts that each internal client is connecting to over time. This number is normally stable, but if it increases, it is likely to be involved in lateral movement.

To identify hosts which are involved in data exfiltration, the system analyzes the volume of outgoing traffic. It compares the byte counts, number of destinations, and number of connections to external hosts. It compares these values over time with the other internal hosts.

The clients with the highest compromise indicators are evaluated by searching OSINT data for exposure indicators that suggest that the client may have been a target of a social engineering attack. These indicators include: 1) social network activity, 2) social network connections, 3) personal information leakage through social networks, 4) organizational information leakage publicly available via employee’s social networking.

The analysis performed in this system should not be affected by zero-day exploits or intermittent APT activity.

The authors have implemented significant parts of their system. The paper presents some experimental results on ranking client systems using network security logs. This work is of interest because it ranks the likelihood that internal systems have compromised by APT’s, and because the unique set of input data it uses.

10. S. Meckl, G. Tecuci, M. Boicu, D. Marcu, and L. A.Center, “Towards an Operational Semantic Theory of Cyber Defense against Advanced Persistent Threats,” in *STIDS 2015 Proceedings,* pp. 58-65, 2015.

The authors view the problem of detecting APT’s as a continuous collaboration of three automated reasoning processes. The first module collects evidence which needs corresponding hypotheses. The second module generates hypotheses which need supporting evidence. The third module tests hypotheses based on their evidence. They propose using the output of IDS, such as Bro or Snort, as the raw input to their application.

This research uses a hybrid knowledge representation. It contains an APT ontology and reason tree patterns with conditions. The tree patterns are learned by the system by interacting with a cyber security expert. The expert supplies examples, explanations, and analogies in an interactive dialog.

The work is an adaptation and extension of research that the group had previously developed for intelligence analysis, and implemented in several systems. This implementation differs from the previous work because all the operations can be performed by cognitive agents, rather than requiring human intervention.

The collection agents that interface with the IDS return evidence as propositions, with a value that indicates the probability that the assertion is true. The method used by the first reasoning agent for generating hypotheses from the database is not well explained. The algorithm for the second agent is basically a recursive decomposition of hypotheses into a tree, until the leaves refer to specific data logs and data items which are to be matched. The reasoning agent which tests hypotheses does so in terms of subjective probabilities ("unsupported" to "certain"). The leaf level hypotheses are tested based on collected evidence, and the probabilities are combined upwards through the tree.

The paper does not address how the proposed system would handle zero-day exploits. Since the reasoning modules are continuously generating / decomposing / evaluating hypotheses, the system should be able to deal with intermittent activities, assuming that it is in continuous operation.

The group had a working implementation of their previous systems, which they are adapting to the APT problem. This work is of interest because it takes a reasoning-based approach to identifying APT's.

11. D. Moon, H. Im, J. D. Lee, and J. H. Park, "MLDS: multi-layer defense system for preventing advanced persistent threats," *Symmetry*, vol. 6, no. 4, pp. 997-1010, 2013

The authors believe that detecting and preventing APT's requires multi-level system which employs a variety of detection methods. They propose a system with a variety of custom agents deployed throughout the protected environment.

The raw input to the system comes from custom monitors deployed on clients, servers, and network devices. The client monitor observes system states using configuration data, whitelisted users, etc. The server monitor observes system state using process information, whitelisted executables, etc. The network monitor observes network state using traffic frequency and volume, whitelisted domains, etc.

These monitors report to the analyzer. Their system statically analyzes suspected malware using a signature and hash value. It dynamically analyzes suspected malware using sandboxbased execution. When a file is determined to be malware, it is reported to the agents, which prevent future execution, transmission, or communication.

The paper does not address zero-day exploits or intermittent activity.

This team has not implemented their system, but the paper presents detailed scenarios of how their system would detect various APT's, indicating detailed design work. The work is of interest because of the system architecture it proposes, and for its taxonomy of attack methods and corresponding detection and prevention techniques.

12. J. M. Namayanja and V. P. Janeja, "Discovery of persistent threat structures through temporal and geo-spatial characterization in evolving networks," *in Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on,* pp. 191-196, 2013.

The authors approach identifying the targets of APT's by characterizing the temporal and geospatial attributes of network communication. They seek to locate central nodes in a network, in order to find patterns conducive to persistent threat behaviors.

The raw data that they use network traffic flows over time and IP address location data. The source and destinations of packets are used to construct a graph in which the vertices are systems and the edges represent transmissions actually made during a time period.

The algorithm begins by finding the connected components of the graph. This work is concerned with the central nodes of the graph, and the authors assume that these are a part of the largest connected component. Then the data is distributed into bins based on timestamps, and central nodes are identified. These nodes are considered consistent if their occurrence is over a threshold, and inconsistent otherwise.

The authors assert that central nodes that are inconsistently available are more likely to be the subjects of APT's. They enhance their analysis by using IP address information to map nodes to countries and organizations.

This team has not implemented their system, but the paper presents detailed scenarios of how their system would detect various APT's, indicating detailed design work. The work is of interest because of the system architecture it proposes, and for its taxonomy of attack methods and corresponding detection and prevention techniques.

13. F. Quader, V. Janeja, and J. Stauffer, "Persistent threat pattern discovery" in *Intelligence and Security Informatics (ISI), 2015 IEEE International Conference on*, pp. 179-181, 2015.

The authors approach identifying APT's with the following characteristics of network traffic: repeated, consistent behavior over time, a single source of transmissions; each transmission constitutes a threat, and the pattern is not typical and not obvious.

The raw input to their system is network data collected by the Snort IDS. Their system preprocesses this input to pick only the attributes of interest, and puts the records into bins of equal time spans.

The main algorithm identifies High Priority Threats by applying Association Rule Mining using the characteristics listed above. Then it intersects the bins to find those threats which are persistent over time.

The authors do not discuss any implications of zero-day exploits for their work. Their binning mechanism addresses intermittent activity over long time periods.

The authors have implemented their design and the paper presents experimental results by analyzing publicly available test data. Their work is of interest because of the use of Association Rule Mining in the detection algorithm.

14. L. Shenwen, L.Yingbo, and D. Xiongjie, "Study and research of APT detection technology based on big data processing architecture," in *Electronics Information and Emergency Communication (ICEIEC), 2015 5th International Conference on*, pp. 313-316, 2015.

The authors detect APT activity with a combination of static and dynamic detection of malicious code. The authors believe that a solution to the problem of intermittent activity over long periods is to be able to store and process extremely large volumes of log data quickly. This motivates their Big Data approach.

The raw input to this system is network traffic and system logs. Network data is collected with a custom network flow probe. It is processed with protocol analysis to find file transfers. System logs are copied into NoSQL and SQL databases for access and search.

Exploits are detected with static and dynamic methods. Unknown attacks are identified dynamically by "suspicious" transmissions and unknown URL's. They are identified statically by covert channels and interrupted connections. Known attacks are detected by retrieving APT signatures from a library, mapping the signature to the current evidence, and testing the signatures using a nearest neighbor algorithm.

The authors have implemented their design and the paper presents experimental results from analyzing publicly available test data. Their work is of interest because it claims to be a completely implemented system and applies Big Data parallelism through a Hadoop cluster. Their paper does not present enough detail to distinguish between the design and implementation.

15. S. Siddiqui, M. S. Khan, K. Ferens, and W. Kinsner, "Detecting advanced persistent threats using fractal dimension based machine learning classification," in *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics,* pp. 64-69, 2016.

The authors believe that the essential strategy for detecting APT's is to monitor and analyze the attributes of TCP/IP connections in real time. The authors refer to other works which argue that APT activity generates a relatively small number of packets in short sessions, or in very long sessions. This contrasts with normal traffic, which generates large numbers of packets within short durations.

The basic input to this system is packet capture files. The packet streams are filtered to remove zero length packets and retransmitted packets. The packets are then summarized into session records which contain two metrics: the total number of packets transferred, and the duration of the session.

The authors present pseudocode for their "correlation fractal based algorithm". They compare the results it obtains that that of k-Nearest Neighbors, with k=3. The idea is to separate the malicious traffic into one class, and the benign traffic into another class.

The authors have implemented their algorithms for finding APT's based on anomalous traffic patterns. The paper presents experimental results based on publicly available test data. Their work is of interest because of the algorithms presented, and the unique choice of features used to detect APT's.

16. A. Vance, "Flow based analysis of Advanced Persistent Threats detecting targeted attacks in cloud computing," in *First International Scientific-Practical Conference Problems of Infocommunications Science and Technology,* pp. 173-176, 2014.

The author approaches the problem of detecting APT's from the perspective of flow based monitoring. It applies a statistical model to network traffic in order to find anomalies that characterize APT's.

The system depends on a collection infrastructure that uses network gateways with flow collectors which capture netflow packets. The raw input is packet size, total packet count, and total packet bytes. These quantities are structured as vectors for the algorithm.

The algorithm uses sketch-based measurement. This is a method for detecting change. It builds a model of normal traffic behavior from past history. Then it searches for changes that deviate from the baseline.

The author implemented his algorithm and applied it to network traffic passing through Internet access points at two cloud data centers. The paper presents experimental results and compares them to the security incident reports at those centers. This work is of interest because it provides evidence that the approach is several times more effective than alternatives that were being used.

17. G. Zhao, K. Xu, L. Xu, and B. Wu, “Detecting APT Malware Infections Based on Malicious DNS and Traffic Analysis,” *IEEE Access*, vol. 3, pp. 1132-1142, 2015.

The authors seek to detect APT attacks using malicious DNS and traffic analysis. The system identifies APT C&C domains using techniques for finding malicious DNS usage. Then it analyzes traffic originating at the suspect address with anomaly and signature oriented detection techniques.

The authors studied large volumes of DNS traffic and the network traffic of suspected malware C&C servers. From this study, they identified 14 features which can be used to identify APT C&C domains.

The system has four parts. The Data Collector is located at the perimeter of the network to record inbound and outbound traffic. The Malicious DNS Detector finds domains related to APT malware and obtains the corresponding C&D server IP addresses. The Network Traffic Analyzer uses signatures to detect known malware. It uses protocol and statistical anomalies to find unknown malware and generate signatures. The Reputation Manager assigns scores to IP addresses. The initial set of malware signatures comes from the VRT Rule sets of snort.
The feature extracted from the DNS and network traffic raw input fall into 5 categories: domain name-based, DNS answer-based, time value-based, TTL value based, and active probes.

The Malicious DNS Detector uses the J48 decision tree algorithm to classify domains. The Reputation Manager decides whether a host inside the network is behaving like it is infected, using a statistical classifi8er.

The authors have implemented their system and the paper presents experimental results from running it at a large institution, four weeks for training and four weeks for testing.   This work is of interest because it provides evidence that the approach and the traffic feature set are effective.