A Context-Based Detection Framework for Advanced Persistent Threats

Paul Giura
AT&T Security Research Center, New York, NY
paulgiura@att.com

Wei Wang
AT&T Security Research Center, New York, NY
wei.wang.2@att.com

***Abstract*—Besides a large set of malware categories such as worms and Trojan horses, Advanced Persistent Threat (APT) is another more sophisticated attack entity emerging in the cyber threats environment. In this paper we propose a model of the APT detection problem as well as a methodology to implement it on a generic organization network. From our knowledge, the proposed method is the first to address the problem of modeling an APT and to provide a possible detection framework.**

I. Introduction

Advanced Persistent Threats (APTs) are some of the most fast growing information security threats that organizations face today [1]. They are operated by very skilled and well funded attackers targeting sensitive information from specific organizations. In the past, APTs were mostly directed at political and military targets [2]. However, over the past few years, attackers increasingly use APTs to strike enterprise targets for financial gains [3]. In these cases, the ultimate goal is to steal intellectual property (IP) created from expensive research, to gain access to sensitive customers data or to access strategic business information that could be used for illegal insider trading or to disrupt organizations business.

APTs have become very sophisticated and diverse in the methods and technologies used, particularly in the ability to use organizations' own employees to penetrate the IT systems [1]. The data breach investigation report presented in [4] concludes that, in 86% of the cases, evidence about the data breach was recorded in the organization logs but the detection mechanisms failed to raise security alarms. This suggests that traditional network defenses can become ineffective in detecting APTs and a new approach is required.

In this work we propose a framework for the APT detection problem and a methodology to implement it in a generic organization network. We introduce a conceptual attack model, called the *attack pyramid*, starting from the attack tree concept [6], [7]. The attack pyramid has the possible attack goal (e.g. sensitive data) at the top, and the lateral planes represent the environments where the events associated with an attack can be recorded. The proposed detection scheme correlates into contexts all the events recorded in an organization that can potentially be relevant for security.

APT can best be defined using the words deriving the acronym. *Advanced* (A) means that attackers are well trained, organized, well funded and utilize a full spectrum of network intrusion technologies, if needed, crafting their own tools. *Persistent* (P) refers to the persistence of the attack over long periods of time. Attackers give high priority to a specific task, rather than opportunistically seeking immediate gain, and maintain a prolonged presence in the compromised organization networks. *Threat* (T) refers to the attackers intention to inflict damage and create loss by disrupting services or stealing proprietary data. APTs are characterized as "low and slow" advanced operations, "low" for maintaining a low profile in the networks and "slow" for long execution time.

Analyses of specific APT instances conclude that each attack is unique and highly customized for each target [1]–[3], [5]. However, across many attacks the stages of the APTs are similar and they differentiate mostly in the specific methods used to pass each stage. Figure 1 shows these stages in the order in which they are typically executed.

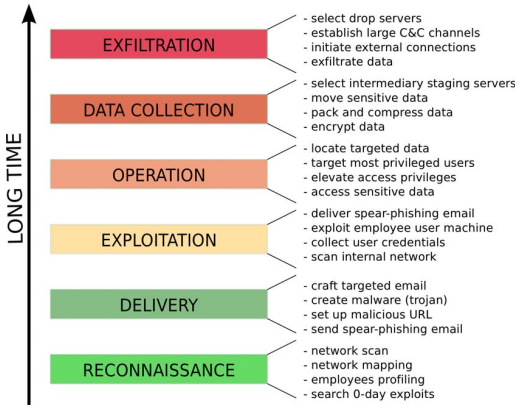

Fig. 1. Typical stages of an APT.

Reconnaissance: In this stage attackers gather information about the target organization resources, employees and relationships with other entities that can be leveraged to reach the target [2]. They scan the network to search for open network services, what network perimeter defense systems are used and what employees have access to the targeted information. Next, attackers build profiles for each targeted employee using public social networks information that employees might be members of (e.g. LinkedIn, Facebook etc).

Delivery: In most cases [1], [2], [5], this stage involves the preparation of a *spear-phishing email*[1] using information

[1]A phishing email targeted at a specific employee, operation is sometimes called *whaling* if the targeted employee is a privileged user.

978-0-7695-5014-5/12 $26.00 © 2012 IEEE
DOI 10.1109/CyberSecurity.2012.16

gathered in the reconnaissance stage. For example, the phishing email might contain invitation to an event scheduled by an organization that the targeted employee trusts [2] with an URL from where the invitee can download a file with related event documents, or an attachment representing the event agenda. Email is the most common entry vector for a compromising malware but other channels can be used as well (e.g. USB based malware, time activated Trojan, etc).

Exploitation: Once the malware makes its way into the organization network to the targeted employee, the downloaded malware is eventually installed and activated. Next, it creates a Command & Control (C&C) connection from the victim machine to the remote master. Once the attackers secure a C&C connection to the victim's computer, they stealthily continue to collect information about the machine's security configurations, related system information, sniff passwords, collect user emails to support future attacks, gather network usernames and directory listings of network shared folders.

Operation: This stage involves persistent presence in the organization network over long periods of time. Attackers move horizontally in the network and identify the servers storing the sensitive information, users having the right access privileges and create the strategy to collect end export the targeted information. Often, operators target the privileged users with new spear-phishing emails, and, if exploits succeed, they elevate their privileges to access the sensitive data.

Data Collection: In this stage operators use the privileged users credentials harvested in the previous stage to get access to the targeted data. Attackers often use sophisticated tools to create redundant C&C channels that can be used in cases of sudden adjustments in the organization security configurations. Similarly, once the targeted information is accessed, redundant copies are created on one or more internal servers that act as "staging points" where the information is segmented, compressed and encrypted before being exfiltrated.

Exfiltration: In this stage the information gathered and carefully packaged in the staging servers is transferred over encrypted channels to multiple external servers that act as drop points. The use of multiple drop point servers is an obfuscation strategy to prevent investigators from finding the final destination of the data.

APTs use a full spectrum of intrusion methods and technologies, therefore are very hard, if not impossible, to detect with conventional network defense mechanisms. In this work, we propose a methodology to approach such a complicated and unconventional attack. We believe that APTs are almost impossible to prevent but, in many cases, they can be detected at one of the operation stages. With this paper we make the following key contributions:

- We propose a conceptual model of an APT starting from the attack tree concept. We propose the modeling of APT operations as an attack pyramid with the attack goal at the top, and the lateral planes as the various environments where the events are recorded (e.g. physical, user, network, application planes, etc.).
- We provide the description of the detection framework using the recorded events, the correlation and detection rules, the historical data, the APT confidence level indicator, the risk level indicator as well as the APT context (e.g. employees, business resources, sensitive data etc.)

The rest of the paper is organized as follows. In Section II we introduce the APT model. In Section III we propose an APT detection framework and in Section IV we present some preliminary experimental results. Section V summarizes the related work and in Section VI we present our conclusions.

II. Attack Model

In this section we first describe the attack tree concept and then we introduce a new APT model, the attack pyramid.

A. Attack Tree

A *threat tree* is a method to represent a threat in a tree structure, first introduced by Edward Amoroso in [6], and later popularized by Bruce Schneier in [7] as an *attack tree*. An attack tree is built as follows: the attack goal is placed as the root and different ways to reach the goal are represented as children. Each two children nodes can be AND (marked with an arc between edges) or OR nodes, representing if both (AND) or each (OR) have to happen to reach the root. Then, each child node can be treated as a goal and new sub-trees can be built having it as root. Creating the possible attack trees gives the organization's information security administrators a way to detect the possible vulnerable elements in the path to the targeted data. However, it is impossible to enumerate all the APT scenarios and to build all possible attack trees because attackers might use a path that was not covered by the initial attack tree model. Figure 2 shows a simple example of an attack tree targeted at the product source code of an organization and a possible path that APT can follow over a long period of time. Root's children nodes are four possible ways to access the source code: to *steal server* with source code OR to download directly from *repository* OR from *user* machine OR to get it from an *insider*. In an APT, first an employee might be targeted with a spear-phishing email (1). After some time, the malware code is executed and attackers take control of the user machine (2). They create a C&C channel (3) to one of the network servers that employee has access to, and is closer to the source code (4). Repository server is accessed (5a), source code is pulled from the repository (5b), data is stored on temporary servers (5c) and finally is exfiltrated (6). The

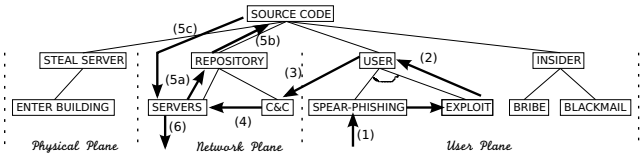

Fig. 2. The attack tree of an APT aimed at source code data.

nodes on the left occur in the physical plane when someone is actually breaking into a building and stealing a server. While the nodes on the right occur in the network plane and user plane. There are two primary issues related to APTs. First, detection and alerting may be very good within a plane, but

they are usually not correlated across planes. Second, even within a plane, detection is not usually correlated over a very long period of time such as weeks, months or even years. This insight led to the creation of another APT model called the *attack pyramid*. So the attack pyramid still looks similar to the standard attack tree, but now its clearly delineated that an attack path may go across planes. An attack may take the circuitous path shown on figure following the stages presented in Section I. Now the challenge becomes one of correlating possibly long-term events across planes and determining they are part of a coherent attack.

B. Attack Pyramid

We propose to model an APT as an *attack pyramid* and we designed the detection framework based on this model. The goal of the attack is placed at the top of the pyramid, and the lateral planes represent the environments where the attack evolves (e.g. physical plane, user plane, network plane, application plane, etc.). Figure 3 shows an intuitive mapping between the attack pyramid concept and the APT stages described in Section I. Pyramid planes are dependent on the specifics of each organization and are defined based on the environments where the events are recorded. Assume that an organization can provide a comprehensive understanding of all the candidate planes that provide facilities to reach the targeted goal G. In order to reach the goal G, the attackers can explore the vulnerabilities and approach the goal by "crawling" from one or multiple planes. Therefore, in the end, a detected APT looks like an attack tree that spans multiple planes. Figure 4

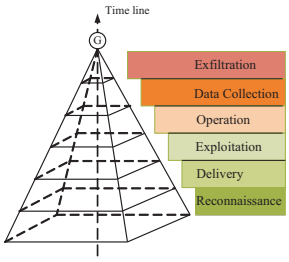

Fig. 3. The attack pyramid.

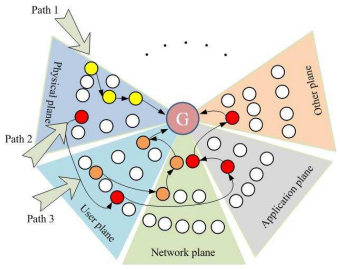

Fig. 4. The unfolded attack pyramid.

provides an intuitive view of an APT tree. The pyramid planes are unfolded to provide the planar view of the attack. The small points represent recorded events in the planes and the colored connected events represent correlated events representing a possible attack. Path 1 represents correlated events in physical plane, Path 2 and Path 3 represent correlated events spanning multiple planes.

C. Events

We propose to use all events recorded in an enterprise environment to detect advanced cyber attacks, not only the events that represent security alerts. The reasoning is that, before the real security incident is detected, an analyst does not know where to look for something bad or relevant for security decision. In general, we identified three types of possible events that can be recorded in an enterprise.

1) Candidate Events: All the events recorded by an organization logging mechanisms in any form. For example, network flow events represent a type of candidate events because they don't necessarily represent attack activity, but can potentially include flow traces generated by an APT.

2) Suspicious Events: Events reported by the security mechanisms as suspicious, or represent events associated with abnormal or unexpected activity. For example, an event corresponding to the upload of a large file from an internal host, that was not engaged in a similar activity in the recent past, can be suspicious.

3) Attack Events: Events that traditional security systems aim to detect with regard to a specific attack activity (e.g. antivirus signatures, etc.). In general, attack events reported by the security mechanisms. For example, a DNS request to a known malicious domain, or a packet containing a known malware binary code.

D. Planes

There is rarely the same way to reach the goal by applying the same sequence of techniques because attackers tend to avoid repeating the same pattern to not be caught. It is often the case that correlated events that lead to the detection of the attack events are generated in multiple planes. The most common examples of pyramid planes are the *physical plane*, the *user plane*, the *network plane* and the *application plane*. Table I lists the possible events sources in each plane.

Planes	Event sources
Physical	building entry logs, hiring events, assets status logs, etc.
User	hierarchy updates, contact updates, affiliation updates, etc.
Network	firewall logs, IPS/IDS logs, netflow logs, etc.
Application	authentication logs, DNS logs, email logs, http logs, etc.

TABLE I
POSSIBLE PYRAMID PLANES WITH THE EVENT SOURCES IN EACH PLANE.

Physical plane: Records all the events that associate possible targets with physical devices or working locations. Events in this plane can be correlated with events from other planes that contain the same users, devices and working location attributes. In most cases physical plane events are used as evidences that the traffic originated from a device in a working location is indeed generated by the claimed user. Additionally, events in this plane record the creation of organization assets that can be the goals of the attacks, such as the start of a new source code repository in a specific physical working location.

User plane: Captures the social engineering process that happens in APTs. At the initial stage, users who have direct access to the possible APT goals with privileges to access the sensitive data will be closely monitored and all events associated with sensitive data access will be recorded. Additionally, there will be recorded both host based events and network events related to privileged users and their possible social ties.

Network plane: All the events recorded by network flow sensors, firewalls, routers, VPN access, intrusion detection and prevention systems will be recorded in this plane. We expect that this plane to hold the largest number of events. These

events can be used to establish the volumetric baseline for network traffic or to correlate with events in other planes.

Application plane: Application gateways (such as http, SIP, RTP, DNS, email, p2p, SSH, ftp, telnet, DHCP, etc.), server and end host application logs will be recorded in this plane. It is often the case that attackers will use spear-phishing on employees inside the network to start the malware delivery, which might be recorded by the email gateway, or a SMS/MMS message containing a malicious URL, which might be blocked at the SMSC message center.

Other planes: Organizations can easily expand the attack pyramid model by adding other interesting planes. A simple extension can be to consider events captured not only by the network layer tools, but also by the application level rules and policies such as database denied accesses, blocked URLs, etc.

III. Detection Framework

A. Detection Framework Design

Figure 5 shows the proposed APT detection framework. The mapping to the attack model is as follows. F_1 to F_n represent the data feeds with the collected events in the organization, multiple feeds can generate events in the same pyramid plane (e.g. IDS/IPS and flow records in the network plane). The

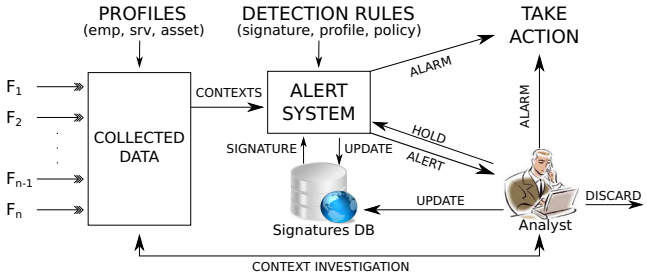

Fig. 5. Advanced attack detection framework.

profile selection is applied to the collected data, events are correlated using correlation rules into contexts, and the contexts are exported to the alert system. The alert system applies the detection rules for each context by using the signature database with "known bad" information whenever requested. Based on evaluating the detection rules, the confidence and risk levels for each context are updated and the thresholds are checked. Next action is determined based on the risk and confidence evaluation. If the risk and confidence parameters are in the alarm zone then an alarm is triggered and the APT incident response is initiated. If the alert system raises an alert, then the security analyst is notified to start investigating the alert. At this point the analyst has four options: 1) to raise an alarm and confirm the reported alert needs immediate attention, 2) to hold the investigation until more evidence is collected about the context in the alert system, 3) to start context investigation by looking at the data source that generated the alert, or 4) to discard the alert in the case of a false positive. Note that, for both scenarios when an alert is upgraded to an alarm, the signatures database is updated with information about the detected attack so that ongoing contexts can benefit from the newly detected incident.

B. Pyramids and Goals

Attack pyramids can be represented as follows. Suppose we consider a set of g goals to protect $\mathbf{G} = \{G_1, \ldots, G_g\}$. Then for each goal, we build a pyramid with the goal as the top and the lateral planes representing the environments where events are recorded. Therefore we can identify a goal i with a set of n planes $G_i = \{P_1, \ldots, P_n\}$. Different pyramids with different goals can share one or more planes. However, for simplicity, without reducing the generality of our model, we describe the details of a pyramid with a single goal G, the scenarios with multiple attack goals being an extension of the single goal analysis.

C. Planes and Events

We assume a plane P_i is defined by a set of events $P_i = \{e_i^1, \ldots, e_i^{k_i}, \ldots\}$, and an event e_i^j generated in plane P_i has the format $e_i^j = (t, id, a_1, \ldots, a_{m_i})$, where j is the event index in the plane P_i, t the time when event was recorded, id the event identifier in the pyramid, and $a_1, \ldots, a_{m_i}$ the attributes for all the events in plane P_i.

D. Correlation Rules

After collecting the events from various sensors feeds, one important problem is to correlate the events relevant to an attack context. To characterize the attack and to define the correlation between events across planes, we define a set of correlation rules $R = \bigcup_{i,j=1}^{n} R_{ij}$, where R_{ij} represents the correlation rules set for events in planes P_i and P_j. A correlation rule r_c in R_{ij} is defined as a function of the events in plane P_i and P_j to the set $\{0, 1\}$ as follows:

$$r_c(e_i^{k_i}, e_j^{k_j}) = \begin{cases} 1, & \text{if } F(a(e_i^{k_i}), a(e_j^{k_j})) = \text{TRUE}, \\ 0, & \text{otherwise.} \end{cases}$$

where $a(e_i^{k_i})$ represents the attributes of $e_i^{k_i}$. Essentially, if we consider the events as points in pyramid planes, a correlation rule creates an edge between correlated events if a boolean function, the correlation function F, computed over the attributes of both events resolves to TRUE. For example, given two events $e_i^1 \in P_i$ and $e_j^1 \in P_j$, such that $e_i^1 = (t_1, 1, a_1, \ldots, a_{m_i})$, $e_j^1 = (t_2, 2, b_1, \ldots, b_{m_j})$ a correlation rule can be defined as:

$$r_2(e_i^1, e_j^1) = \begin{cases} 1, & \text{if } a_1 = b_1 \text{ AND } a_2 < b_2, \\ 0, & \text{otherwise.} \end{cases}$$

E. Detection Rules

We identified three major types of detection rules based on the data source for creating them:

1) Signature based rules: Require checking the new observed events and behavior against known attacks and malicious behavior. In order to be effective, the signatures database should be constantly updated.

2) Profiling based rules: Require checking the observed profile and behavior of the monitored entity (e.g. users activity,

services metrics, etc) with profile and behavior baselines. The profiling parameters are changing with the evolution of entity activity because the input data is also changing.

3) Policy based rules: Are the static rules based on the organization policies. The rules are updated when new policies are in effect. These rules can use parameters from the signature databases or variables from the profiling systems, not necessarily based on dynamic data statistics.

F. Attack Context

We define an attack context A_j as a set of events correlated across multiple planes corresponding to the monitored entity, along with specific context variables. For example, a context can represent privileged users having access to the sensitive information, internal servers hosting the sensitive data, or critical resources considered essential to run organization business. A formal representation of a context is $A_j = \{E, R, W, H, C, L, G\}$, where $E = \{e_1^{k_1}, \ldots, e_i^{k_i}, \ldots\}$ represents the set of correlated events that are relevant for the advanced attack with goal G, R the correlation rules set for the context, W the time window when the context is active (e.g. hours, days, months, etc), H the historical information about known attacks, C is the attack confidence indicator and L is the risk level of the attack.

The risk level and the confidence indicator are used to quantitatively evaluate the threats to the goal. Essentially the attack confidence is defined as the weighted attack confidence across the planes because not all the attacks in each plane might be of equal value to the detection of APT. For example, the detection of a brute force attempt in the user plane might have less weight than the detection of a connection to a known drop server. In the latter case the APT is in exfiltration stage, while in the first case the attack might be in the reconnaissance stage still out of the network perimeter. C is computed as:

$$C = \sum c_i \cdot w_i, \sum w_i = 1, i = 1, \ldots, n \qquad (1)$$

where c_i represents the the detection confidence and w_i the weight of an attack event in plane i. Essentially, c_i values represent the confidence indicators for events recorded in different planes that are correlated in the context.

In our model, an APT incident is detected when a set of observed events makes the attack confidence indicator C and a risk level L raise above specific thresholds C_0 and L_0 respectively. If the risk level L reaches L_0, with confidence above C_0, then an APT alarm will be triggered. The thresholds are parameters specific to each organization environment. When new events are identified and correlated to the existing events in APT contexts, the confidence C can be increased. The confidence indicator is intended to provide a measure of the expected false positive parameter. The risk level, L, can be a numeric value representing the possible attack stage from 1 (Reconnaissance) to 6 (Exfiltration), or the minimum distance to the goal defined using a specific metric (e.g. number of remaining hops to data server), or just a binary indicator used to fire an alarm whenever a particular attack event is detected. Figure 6 shows the relationship between the risk level L and

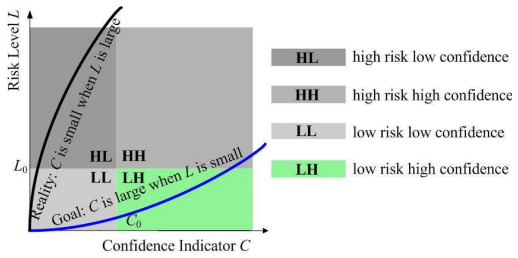

Fig. 6. The relation of risk level L and confidence indicator C.

the confidence indicator C. When both parameters are low (LL), the attack is not clear. When the risk level is high and the confidence is low (HL), then immediate action is required and the correlation rules do not properly capture all the relevant events in order to detect the attack with high confidence. In the case when both the risk level and the confidence are high (HH), then the attack is probably detected in late stages with some damage to the organization. The final and ideal case is when the risk is low and the confidence indicator is high (LH). In this case the APT is detected with high confidence in early attack stages, and most likely the organization is not compromised. For the purpose of this paper we consider the risk level to have only two possible values 0, 1 and we define it based on the confidence indicator as $L = 1$ if C $\geq C_0$, and $L = 0$ otherwise.

IV. Evaluation

We tested the feasibility of our method by attempting to build user profiles using data collected in a large enterprise, with the goal to identify possible attack contexts within and across profiles. The evaluation is rather illustrative and does not show all the results because of the space constrains.

Data: We used the data collected from several security feeds over a period of one hour just to maintain reasonable running times and to illustrate how the proposed framework can be implemented. We used events recorded in the network plane (VPN logs, IDS logs and firewall logs) and application plane (authentication logs and Internet proxy logs). We believe our approach is easily scalable to a larger data set that can include more feeds collected over longer periods of time.

Attribute	Value
# of feeds	10
# of events	19,584
# of user profiles	3,461
# of IP address profiles	465

TABLE II
Testing data characteristics.

Correlation Rules, Profiles, Contexts: We attempted to built profiles for users that authenticated with the intranet. First, the events were normalized to the format in Section III. Then, we built profiles for users with user ID (UID) attribute not null in at least one recorded event. If the UID was null for one event, we built profile for the source IP recoded in the event. Any two events are correlated and added to the same profile if they had the same UID or the same source IP attributes. The correlation function for each profile was defined only on the UID and sourceIP attributes. If we identified a UID

and a source IP in the same event, we merged the UID and source IP profiles into a context with the user ID representing the goal. Table II shows the data statistics.

Detection: We consider only the security related events for all feeds. For each feed type (e.g. Authentication, VPN, IDS, Firewall, Proxy events) we build a simple profiling based detection rule that compares the number of security messages per profile N_i with the average number of messages per profile $\bar{N}$ for profiles that have at least one message. We defined the confidence indicator for each feed type (e.g. c_i from Eq. 1) as $c_i = N_i/\bar{N}$. We select the weights for the context confidence indicator to be $w_i = 0.20$, equal across types.

feed types (planes)	Auth	VPN	IDS	Firewall	Proxy
avg msgs per profile (N)	4.54	3.09	7.87	17.68	44.20
confidence weights (w_i)	0.20	0.20	0.20	0.20	0.20
detection 100% confidence	$C_0 = 1$, Detected = 75, FP = 0%				
detection 75% confidence	$C_0 = 0.75$, Detected = 104, FP = 27.88%				
detection 50% confidence	$C_0 = 0.50$, Detected = 202, FP = 62.87%				
detection 25% confidence	$C_0 = 0.25$, Detected = 501, FP = 85.03%				

TABLE III
EXPERIMENTAL DETECTION RESULTS.

We computed the detection confidence for each context using Eq. 1, and we reported as detected the cases when the context confidence indicator is larger than the threshold C_0 for each case. Table III shows the parameters used, the number of contexts under attack and the false positives rate for each confidence level. We assume no false positives for 100% confidence. We compute the false positives rate (FP) as the number of contexts detected by using $C_0 = 0.75, 0.50, 0.25$ and not detected by using $C_0 = 1$. As expected, the number of attacks detected increases as the confidence threshold decreases, because of the false positives inherited from the detection mechanisms in each plane.

V. RELATED WORK

In last recent years there was an increasing research interest towards efficient complex event processing. The system presented in [8] provides a method to process multiple streams of events by using a query language that implements specific operators using finite state machines. Similarly the general architecture proposed by IBM in [9] is intended to serve the aggregation and correlation of multiple streams of events. These systems can be used to process real-time streams of events but the detection of an advanced security attack spanning for long periods of time is not their primary goal. Since there are few instances of well documented cases of APTs [1], [2], [5], there is not much research literature that addresses the APT problems. From our knowledge, our effort is among the first steps to model these sophisticated attacks, and to formalize the problem in specific terms. However, our assumptions about the attacks are based on publicly reported cases in [1]–[5]. In defining the APT model we used the threat tree approach introduced by Edward Amoroso in [6], later popularized by Bruce Schneier in [7] as attack trees and, more recently, used in [10] to describe probabilistic complex attacks. However, our final model, the attack pyramid, is intended to be more comprehensive and to correlate nodes of the tree based on more flexible rules. In this way we are guaranteed that, regardless of the attack vector used, if a presumably benign event is recorded, based on the correlation with other events we might be able to discover if the event represents an attack activity that was not reported initially. Finally, in [11] authors introduce the concept of enhanced monitoring using historical data. In their approach whenever an event is recorded, a historical database is queried for the context of the event, and all similar events are returned as the result of the query. Our approach is different in that we explicitly store each context that is relevant for each monitored entity rather than defining a general correlation function for the context definition across all events recorded.

VI. CONCLUSION

We propose a model for the APT detection problem as well as a methodology to implement the detection model on a generic organization network. We introduce the attack pyramid model, starting from the attack trees and provide an APT detection framework that takes into account all the events in an organization. We propose a methodology to correlate all the relevant events across all the pyramid planes, to detect the APT within a context, and finally we evaluated the feasibility of our approach with sample data from an enterprise network. In future work we plan to investigate different methods to assess the confidence and risk for each attack context, while using a larger number of feeds and a richer set of correlation and detection rules.

REFERENCES

[1] RSA, "RSA Security Brief: Mobilizing Intelligent Security Operations for Advanced Persistent Threats," http://www.rsa.com/innovation/docs/11313_APT_BRF_0211.pdf, February 2011.

[2] B. Krekel, G. Bakos, and C. Barnett, "Capability of the People's Republic of China to conduct cyber warfare and computer network exploitation," The US–China Economic and Security Review Commission, Washington, DC, Research Report, 2009.

[3] SANS Technology Institute, "Assessing Outbound Traffic to Uncover Advanced Persistent Threat," http://www.sans.edu/student-files/projects/JWP-Binde-McRee-OConnor.pdf, May 2011.

[4] Verizon, "2010 Data Breach Investigations Report," http://newscenter.verizon.com/press-releases/verizon/2010/2010-data-breach-report-from.html, July 2010.

[5] Damballa, "The Command Structure of the Aurora Botnet," http://www.damballa.com/research/aurora/, March 2010.

[6] E. G. Amoroso, *Fundamentals of computer security technology*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.

[7] B. Schneier, "Attack Trees - Modeling Security Threats," *Dr. Dobb's Journal*, December 1999.

[8] L. Brenna, A. Demers, J. Gehrke, M. Hong, J. Ossher, B. Panda, M. Riedewald, M. Thatte, and W. White, "Cayuga: a high-performance event processing engine," in *Proceedings of the 2007 ACM International Conference on Management of Data*, ser. SIGMOD '07, 2007.

[9] IBM, "A Conceptual Model for Event Processing Systems," http://www.ibm.com/developerworks/webservices/library/ws-eventprocessing/index.html, February 2010.

[10] S. A. Camtepe and B. Yener, "Modeling and Detection of Complex Attacks," in *3rd International Conference on Security and Privacy in Communication Networks*, September 2007.

[11] M. Balazinska, Y. Kwon, N. Kuchta, and D. Lee, "Moirae: History-Enhanced Monitoring," in *CIDR*, 2007.