A Collaborative Tool for Modelling Multi-stage Attacks

Ian Herwono and Fadi Ali El-Moussa
Security Futures Practice, Research & Innovation, BT, Ipswich IP5 3RE, U.K.
{ian.herwono, fadiali.el-moussa}@bt.com

Keywords: Cyber Security, Attack Patterns, Pattern Recognition System, Knowledge Sharing.

Abstract: Cyber-attacks that are conducted in multiple stages over short or long periods of time are becoming more common. One approach for detecting such attacks at an early stage is to make use of attack patterns and attack signatures to provide a structure for correlating events collected from various sensors in the network. In this paper, we present our ongoing work on a pattern recognition system that aims to support cyber-defence analysts in sharing their attack knowledge and threat intelligence in the form of attack patterns or scenarios that can later be used to discover potential security breaches in their network. Our main goal is to allow the analysts to associate the attack patterns with their own organisation's security data and thus benefit from the collective attack knowledge without revealing any confidential information. We present the architecture of the system and describe a typical process for modelling multi-stage attacks. We demonstrate how its analytics engine interprets an attack pattern, tasks the data source agents to fetch and correlate relevant security events, and reports the results back for visualisation and further investigation.

1 INTRODUCTION

Today's detective tools are prone both to false positives and to failure to detect novel or evasive attacks. The alerts issued by such tools are usually isolated episodes, localised in time and space, leaving the cyber-defence analyst to join up the dots and work out the bigger picture. Meanwhile, considerable valuable information lies latent in disconnected silos of low-level data. One approach to overcoming these obstacles is to make use of attack patterns to provide a structure for correlating incidents that are separated in time and space in order to aid recognition and diagnosis of malicious activity and guide appropriate and timely response.

The challenge addressed in our work is how to capture the knowledge of skilled analysts in the form of attack patterns or scenarios embodying regularities, causal relationships and correlated observables, then use these reliably to recognise attacks at an early stage, predict their evolution and allow actions to be taken to mitigate their effectiveness. A major obstacle is that while general patterns are often followed, there can be significant variation leading to uncertainties in both recognition and prediction. The situation may be further confused in large organisations or enterprises by multiple attacks happening at the same time. It is therefore important to encourage and support security experts to share domain knowledge and cyber intelligence with their peers either within or outside the organisation. They should combine their efforts to identify and validate various ways of collecting and examining attack evidence from multiple data sources in corporate network environment. However sharing attack data and threat intelligence between security providers and businesses without revealing sensitive information proves to be a major issue today. Our proposed solution is to have a platform in place where security analysts can share attack patterns within the same enterprise or between multiple enterprises, and use these patterns to check whether attacks have happened without having to reveal any confidential data. We designed a pattern recognition system to lay the groundwork for such a collaborative platform.

The remainder of this paper is structured as follows. Section 2 presents related works. Section 3 introduces the architecture of our pattern recognition system. Section 4 describes the attack modelling process. Section 5 shows how the system's analytics engine works. Section 6 provides the conclusions and discusses future work.

Herwono, I. and El-Moussa, F.
A Collaborative Tool for Modelling Multi-stage Attacks.
DOI: 10.5220/0006137103120317
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 312-317
ISBN: 978-989-758-209-7

2 RELATED WORK

Our work aims to design a system for recognising cyber-attacks that are conducted in multiple steps or stages using several attack paths to achieve its ultimate attack objective, e.g. exfiltration of corporate data. (Clark and Landau, 2010) analysed such multi-stage attacks and discussed how to trace them. (Alserhani et al., 2010) examined statistical modelling techniques for alert clustering. (Bhatt et al., 2014) developed a framework to detect multi-stage attacks using the *Intrusion Kill Chain* model (Hutchins et al., 2011). (Barnum, 2007) introduced the concept of attack patterns as a mechanism to capture the attacker's perspective. The work was related to the *Common Attack Pattern Enumeration and Classification* (CAPEC) initiative of the Department of Homeland Security (DHS) (https://capec.mitre.org).

The system described in this paper employs the knowledge-based model approach. Its knowledge base consists of a repository of attack patterns that will be captured from experienced security analysts, along with relevant datasets such as Web Proxy logs. We consider attack pattern as a structure to correlate security and network events that are separated in time and space in order to aid recognition of attacks at an early stage. Such an attack pattern should not be mistaken for an attack graph, which is a structure to represent all possible sequences of exploits that an intruder can carry out to attack computer networks (Ammann et al., 2002). The method to formally describe the attack pattern is specific to implementation, and CAPEC schema may well be used in the future.

3 SYSTEM ARCHITECTURE

The system architecture is depicted in Figure 1 and the components are described in the subsequent sections.

3.1 Authentication & Authorisation

A simple authentication system using username and password mechanism is currently implemented. The authorisation is based on administrative grouping. This ensures that only authorised users can use the system in accordance with their assigned roles, e.g. administrator, analysts, etc.

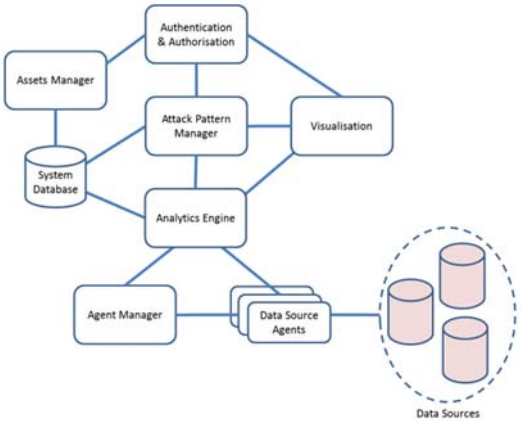

Figure 1: System architecture.

3.2 Attack Pattern Manager

The attack pattern manager is responsible for the entire lifecycle of each attack pattern, starting from its creation up to its removal. It acts as the main interface for user interactions. It also provides control over the usage of existing attack patterns, either for monitoring live events or investigating historical events.

3.3 Assets Manager

The assets manager provides simple management of (critical) assets that can later be associated with any of the existing attack patterns. It provides grouping of assets that can be identified by their IP address or network sub-domain using Classless Inter-Domain Routing (CIDR) notation.

3.4 System Database

The system database is usually a relational database that is used to persistently store the attack patterns, along with other application data such as details of available external data sources, authentication credentials, and critical assets.

3.5 Data Sources

Data sources are external data stores that collate structured event or log data generated by various systems in the network, e.g. IDS, DHCP, Web Proxy, etc. We assume that any necessary pre-processing and enrichment of various datasets, such as parsing of event attributes or IP address lookup, has already been carried out prior to storing the data.

Two types of data sources are currently supported:

- Conventional SQL database management systems such as Oracle, MySQL, or PostgreSQL, and
- ElasticSearch storage systems.

Details of a new data source, e.g. server/cluster details, tables and attributes mapping, etc. can be added at runtime via the admin user interface. An ElasticSearch data source may typically contain different types of event or log data as it is document-oriented and does not use fixed schema.

3.6 Data Source Agents

Data source agents are centrally-managed software agents that communicate with external data sources to determine quantifiable measures for specific cyber events, e.g. the number of failed login attempts within a five minutes time frame, or the number of detected malware within the last 24 hours. The query and filter parameters to match and aggregate the relevant events are specified by the user during the attack modelling exercise. In case of ElasticSearch data source, much of the required filtering and aggregation tasks are taken over by the ElasticSearch engine.

3.7 Agent Manager

The agent manager is the single point of contact within the system for instantiating and parameterising different types of data source agents. It is consumed by the analytics engine.

3.8 Analytics Engine

The analytics engine is responsible for matching the attack patterns against historical and live datasets. Historical data is normally used to test and validate attack patterns or to carry out forensic analysis. Such validation allows analysts to refine the patterns and readjust the measures/metrics in order to increase pattern detection accuracy. The analytics engine interacts with a number of agents to query data from different sources. Section 5 provides more details.

3.9 Visualisation

The visualisation component provides graphical views of the attack patterns, their monitoring status and results. An external visual analytics tool can be loosely integrated into the system's user interface in order to support security analysts in their further investigation of potential threats.

4 ATTACK MODELLING

4.1 Attack Pattern

Each attack pattern is essentially a plan template embodying an attack strategy, typically consisting of multiple steps or stages. Associated with each stage and also with the overall pattern are observable events and conditions that would normally occur during, before and after execution of the plan.

The attack pattern repository will be populated based on the experienced analyst's knowledge of historic attack cycles. For example, a Distributed Denial of Service (DDoS) campaign often follows a pattern: due diligence, intelligence gathering, vulnerability scanning, defacement and DDoS. If the events associated with due diligence, intelligence gathering, and vulnerability scanning are observed, then we can predict that defacement and DDoS will probably follow after intervals similar to those observed previously.

4.2 Quantifiable Measure

When modelling an attack the crucial step is to specify a quantifiable measure or indicator for the relevant events at each stage. The user needs to select which external data source such measure should be derived from and which parameters to use for querying and aggregating the event data.

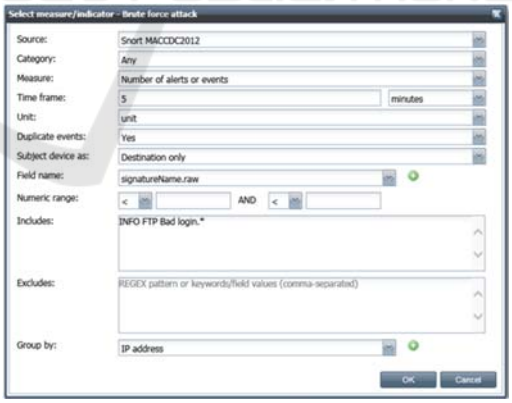

Figure 2: Specifying quantifiable measure.

Figure 2 shows an example setup for a data source agent to detect possible brute force attack by observing the number of failed login attempts into an FTP server. The selected "Snort MACCDC2012" data source represents an ElasticSearch data source that contains Snort IDS alerts that were generated from the 2012 Mid-Atlantic Collegiate Cyber Defence Competition dataset (MACCDC, 2012).

Essentially the agent needs to count the total number of alerts with the signature “INFO FTP Bad login” that were reported within five-minute time windows for each destination IP address.

The user should then assign the measure a threshold value which will later be examined by the analytics engine prior to making the decision whether or not to trigger a transition to the subsequent stage. Dependency between the events of successive stages can be specified to indicate whether events at a particular stage should only be observed after some characteristic data from one of the preceding stages, e.g. IP addresses of affected hosts, has been passed on by its data source agent. Events will be observed either periodically (e.g. until their threshold value is exceeded) or only once (e.g. to check if certain events have occurred in the past).

Figure 3 shows the complete setup for detecting an attack stage “Brute force attack” using the above-mentioned measure. As it will be the first stage in the example attack pattern (Figure 4) it has no dependency on other stages. The threshold is set to 20, i.e. twenty failed login attempts within five-minute time window, which should be examined periodically (monitor mode).

Figure 3: Attack stage “Brute force attack”.

Figure 4: Example attack pattern.

The “Brute force attack” stage is followed by two subsequent stages, i.e. “Suspicious activity for privilege gain” and “Suspicious activity for information leak attempt”. Both subsequent stages look for suspicious activities involving the destination hosts that were selected at the preceding stage, i.e. threshold conditions were met.

5 ANALYTICS ENGINE

The analytics engine owns the task of matching attack patterns against sequences of observed events from live or historical datasets in order to determine if a potential attack campaign is under way. The engine consists of independent process entities whose workflows are dictated by the attack patterns. Each active attack pattern is assigned a single parent process which will create as many child processes as necessary over time.

5.1 Parent Process

Once an attack pattern is activated, e.g. to analyse historical cyber events, a (parent) process is started and the following tasks will be performed.

5.1.1 Pattern Data Retrieval

The process control entity retrieves the pattern data from the system database. It extracts the information about which of the attack stages should be monitored from the start, hence referred to as *start stages*.

5.1.2 Agent Parameterisation

The data source agent associated with each of those start stages is created. The parameters for its quantifiable measure, such as type of measurement, time frame, threshold value, etc. are passed on to the agent.

5.1.3 Agent Scheduling

The time interval, at which each data source agent executes its task, is configured in the scheduler. The agent may then carry out its assigned task periodically.

5.1.4 Result Examination

The result from each agent, i.e. the measurement value, is reported back to the process control entity. The agent indicates whether or not the threshold condition has been satisfied.

5.1.5 Child Process Creation

Each time the threshold at particular start stage has been met the control entity creates a new child process for observing the subsequent stage. The reporting agent (of the parent process) may hold characteristic information about the relevant events,

e.g. hosts' IP addresses, which need to be monitored at the subsequent stage. Each child process can be seen as a path that needs to be followed throughout the attack cycle separately.

5.2 Child Process

Throughout its lifecycle a child process will perform similar tasks as its parent, i.e. parameterising and scheduling data source agents, examining agent measurement values, and creating new child process for subsequent stages. A child process may terminate as follows:

- **End of attack cycle**: This means that the final attack stage in a sequence has been reached and the associated data source agent has indicated that the threshold has been met and no more subjects (e.g. hosts) remain to be monitored at that stage.
- **Timeout**: A child process times out if the measurement threshold value has not been met after a specific time; from the security point of view this may suggest false alarms or that a potential attack campaign has not progressed any further.

5.3 Message Flow

Figure 5 depicts the flow diagram for a simple attack pattern consisting of two stages, i.e. *Malware alerts*, and *IPS alerts*. At the start stage (Malware alerts) the system should report any device present in the network that had more than five unique malware alerts within the last 24 hours. Such information can usually be obtained from a database that collated alerts from host-based malware scanners. At the subsequent stage (IPS alerts) the system should continuously check whether any of those reported devices had also triggered an alert on the network's Intrusion Prevention System (IPS).

The following communications and data exchange happen between the process entities once the attack pattern is activated:

1. The Parent Process Control entity (PPC) retrieves the pattern data from the system database and determines the start stage, i.e. *Malware alerts*. It then sends a request to the Agent Manager (AM) for instantiating an Anti-Malware data source agent with the corresponding measurement parameters and a threshold value (i.e. 5 alerts). Eventually PPC instructs the scheduler (SCH) to trigger the agent (AMA) at a specific time interval.
2. Each time the trigger fires, AMA retrieves the relevant data from the Anti-Malware data source (ADS), calculates the measurement value (i.e. the number of unique malware alerts per device within the last 24 hours), and applies the threshold. It then reports the result back to PPC; the result contains the computed measurement value, a flag indicating whether or not the threshold has been satisfied, and other information (e.g. IP address list).

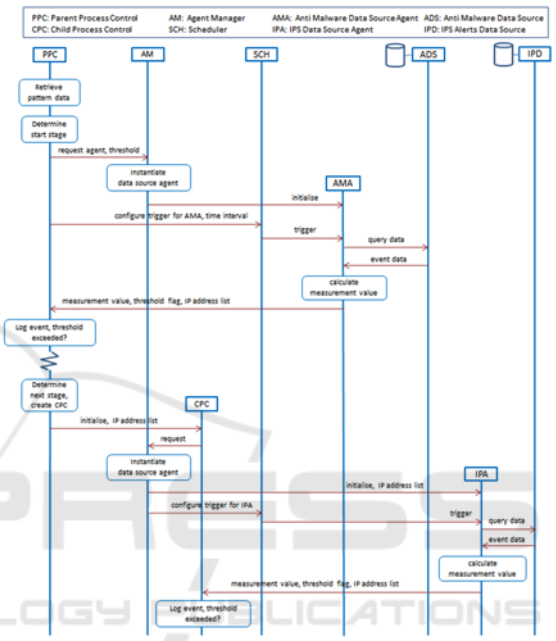

Figure 5: Flow diagram for an attack pattern.

3. PPC logs the result and checks the threshold flag. If the threshold has not been satisfied, no further action is taken. AMA will keep periodically querying new data and reporting the result to PPC.
4. Once the threshold is exceeded, PPC extracts the measurement parameters of the subsequent stage (i.e. *IPS alerts)* and proceeds with the creation of a new Child Process Control entity (CPC). PPC extracts the list of device IP addresses from the received agent's (AMA) result and passes it on to CPC. The system will be using temporal information and the IP address list to correlate events retrieved from two different data sources, i.e. Anti-Malware and IPS Alerts data sources.
5. CPC sends a request to AM for instantiating an IPS data source agent (IPA) with the task to compute the number of IPS alerts within the last 24 hours on devices with the specified IP

addresses. CPC then instructs the scheduler (SCH) to trigger the agent (IPA) at a specific time interval.

6. Whenever IPA receives a trigger signal, it will then retrieve the relevant data from the IPS Alerts data source (IPD), calculate the measurement value (i.e. the number of alerts per device within the last 24 hours), apply the threshold, and send the result back to CPC. CPC will log the result and check the threshold flag. If the threshold has not been satisfied, no further action is taken. IPA will keep observing new relevant events until it times out.

The results that have been logged by the parent and child process entities can be processed by the system's *Visualisation* component and presented to the users.

6 CONCLUSIONS

Our vision was to build a collaborative platform where security analysts of different organisations can combine their efforts and contribute to a repository of attack patterns that prove to be up-to-date, comprehensive and reliable for detecting sophisticated cyber-attacks at an early stage, such that appropriate countermeasures can be initiated in timely fashion. Our approach was to model an attack or security breach as a sequence of observable events. We found that defining an attack pattern was not a straightforward task unless combined with the ability to analyse historical or sample attack data at the same time. It was essential for security analysts to have access to relevant data sources in order to derive the metric or measurement parameters such as threshold value or time window as part of the attack modelling process.

Our ultimate goal was to allow security analysts to share attack patterns and apply them to their own organisation's security data without revealing any confidential information. To some extent this has already been supported in our current system, but further work is required to carefully identify the security and privacy requirements and implications in enterprise environments and to develop techniques and policies to ensure that sensitive data is not shared inappropriately.

ACKNOWLEDGMENTS

This work has been carried out in the framework of the *Collaborative and Confidential Information Sharing and Analysis for Cyber Protection – CISP* project, which is partially funded by the Commission of the European Union. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers, the CISP project, or the Commission of the European Union.

REFERENCES

Clark, D. D., Landau, S. 2010. The Problem isn't Attribution; It's Multi-Stage Attacks. In *Proceedings of the Re-Architecting the Internet Workshop* (Philadelphia, US, Nov 2010). ReArch 2010. ACM.

Alserhani, F., Akhlaq, M., Awan, I. U., Cullen, A. J., Mirchandani, P. 2010. MARS: Multi-stage Attack Recognition System. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications* (Perth, WA, April 20-23, 2010).

Bhatt, P., Yano, E. T., Gustavsson, P. M. 2014. Towards a Framework to Detect Multi-Stage Advanced Persistent Threats Attacks. In *Proceedings of the IEEE 8th International Symposium on Service Oriented System Engineering* (Oxford, UK, Apr 2014). SOSE 2014.

Hutchins, E., Cloppert, M., Amin, R. 2011. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. In *Proceedings of the 6th International Conference on Information Warfare and Security* (Washington, DC, Mar 2011).

Barnum, S. 2007. An Introduction to Attack Patterns as a Software Assurance Knowledge Resource. In *OMG Software Assurance Workshop* (Fairfax, VA, Mar 2007).

Ammann, P., Wijesekera, D., and Kaushik, S. 2002. Scalable, Graph-based Network Vulnerability Analysis. In Proceedings of the 9th ACM Conference on Computer and Communications Security (Washington, DC, Nov 2002). CCS'02.

MACCDC. 2012. Capture files from Mid-Atlantic CCDC (Collegiate Cyber Defense Competition). URL: https://www.netresec.com/?page=MACCDC.