Describing Advanced Persistent Threats Using a Multi-agent System Approach

Sravani Teja Bulusu, Romain Laborde, Ahmad Samer Wazan, François Barrère, Abdelmalek Benzekri

Describing Advanced Persistent Threats using a Multi-agent system approach

Sravani Teja Bulusu, Romain Laborde, Ahmad Samer Wazan, Francois Barrère, Abdelmalek Benzekri
IRIT / Université Paul Sabatier 118 Route de Narbonne
Toulouse, France
sbulusu@irit.fr, laborde@irit.fr, ahmad-samer.wazan@irit.fr, francois.barrere@irit.fr, abdelmalek.benzekri@irit.fr

***Abstract*—Advanced Persistent Threats are increasingly becoming one of the major concerns to many industries and organizations. Currently, there exists numerous articles and industrial reports describing various case studies of recent notable Advanced Persistent Threat attacks. However, these documents are expressed in natural language. This limits the efficient reusability of the threat intelligence information due to ambiguous nature of the natural language. In this article, we propose a model to formally represent Advanced Persistent Threats as multi-agent systems. Our model is inspired by the concepts of agent-oriented social modelling approaches, generally used for software security requirement analysis.**

Keywords— Attack trees; Advanced Persistent Threats; multi-agent systems

I. INTRODUCTION

The advent of Advanced Persistent Threats (APT), also known as targeted attacks, have set up a new trend of cyber espionage. Unlike the conventional attacks, APT attacks are launched by an organized group of highly skilled attackers with well-focused motives (e.g., stealing data, stealing money, espionage, etc.). The primary purpose of the threat intelligence is to help the organizations in understating the context of such threats. In this regard, many security organizations publish security bulletins and intelligence reports detailing on several case studies. However, this knowledge is generally shared as a list of documents that are written in natural language, like *APT notes* on github[1]. This raises the question of comprehensibility as well as the reusability of such reports. Besides, none of the existing models provide good support to model APT as they do not consider the coordinated behavioural aspects between the multiple threat agents [1], [2]. For this reason it is not yet clear on how to model APT like targeted attacks. In this article, we propose a graphical model to represent APT as multi-agent system. Our model is inspired by the concepts of agent-oriented social modelling approaches, generally used for software security requirement analysis. We will illustrate our model using the carbanak APT case study.

The rest of the article is structured as follows. Section II briefs on carbanak APT attack campaign and our analysis. Section III illustrates our model using carbanak APT campaign. We also briefly introduce the terminology and notation. I section IV, we discuss on the usability aspects of our model. Finally, we conclude our work in Section V.

[1] https://github.com/kbandla/APTnotes

II. ANALYZING CARBANAK CASE STUDY

Carbanak APT[3], disclosed in public by Kaspersky labs in 2015, targeted financial institutions like banks with an ultimate motive to steal money. This APT campaign is mainly executed in three phases: 1) Infection and transmission, 2) Harvest intelligence on target system operations and 3) orchestrate attacks to steal money. Starting with a simple social engineering attack to spread the carbanak malware, the attackers executed series of attacks to gain access to the critical systems of banks (e.g., bank servers). In carbanak, variety of ways were found to steal the money from the banks, thanks to the intelligence gained from video and other monitoring techniques allowing attackers to understand the operational workflow of the victim. For instance, attackers created fake transactions, through online banking or SWIFT transfers. These transactions were intelligently made after the regular verification process as per bank operations. It is also worth noting that the attackers had deliberately put an upper limit to the amount of stolen money per victim to $10 million USD. This value represents the maximum amount of money that can be transferred via mules or the maximum money budgeted by banks for fraud risks.

From the carbanak case study, we acknowledge that an APT is a composite attack scenario involving multiple attacking agents (human or system) who use several attack methods and attack patterns such as coordinated attacks, parallel attacks, multi-step attacks in order to achieve their ultimate motives. All these aspects together make an APT sophisticated and complex while compared to a traditional attack. We designate them as the elements of an APT attacking system. In order to model such attacking system, we should be able to specify all these elements of the APT attacking system. Respectively, we provide a list of anticipated requirements based on the characteristic features observed in the example case study. It is to note that the list is not exhaustive and describes the minimum requirements needed to model APT:

- **Req1**– Must use unambiguous modelling language.
- **Req2**– Must facilitate to express all the attacking agents
- **Req3** – Must facilitate to express all the threat goals
- **Req4** – Must facilitate to express all the attack methods
- **Req5** – Must facilitate to trace the attack path
- **Req6** – Must facilitate to express the social and technical dependencies among the attacking agents (human/system).
- **Req7** – Must facilitate to express the protection goals
- **Req8** – Must facilitate to express the attack patterns used.

III. Proposed Anti-STS model

The current existing attack modelling techniques [1], [2], does not provide good support to model APT as they don't facilitate to express multiple attacking agents and their interaction dependencies to succeed with the attack (**Req2**, **Req6** and **Req8**). Therefore, we propose to view APTs as socio-technical systems representing the interplay of the attackers and technical systems in order to successfully execute the strategic plan. For this, it seems interesting to use a multi-agent systems approach integrating the social modelling concepts from STS model[4] and the Anti-goal modelling concepts from Secure KAOS[5] that are generally used in security requirements analysis. Respectively, in our model, we assume that the attackers' goal is to build an APT campaign system. We name our approach as **Anti-STS**.

Under this vision, the APT system has multiple attacking agents (human/system). We represent them as anti-system roles such as social engineer, hacker, system analyst, etc. who collaboratively work to manifest a threat (**Req2**). Figure 1 depicts the comparison between the notations of STS and Anti-STS.

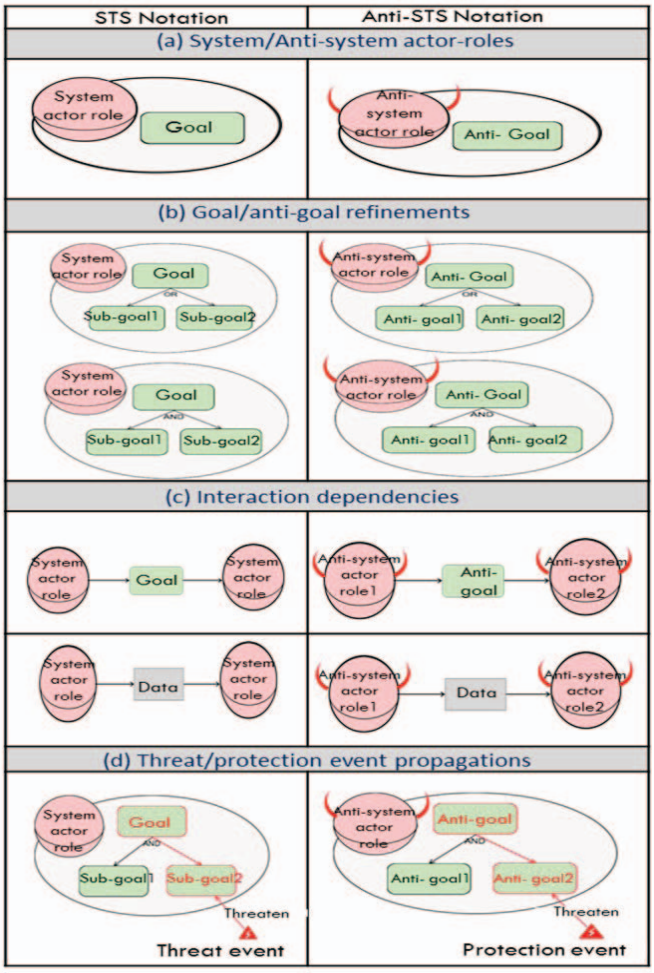

Figure 1: Comparison between STS and Anti-STS Notations

Accordingly, the pictogram used to represent the threat actor roles is depicted with horns. The malicious objectives are represented as the anti-goals within the scope of each threat actor, see Figure 1a. Threat scenarios are driven by some malicious motive (e.g., spoil reputation, gain money) and follow a procedural approach. The whole composite attack scenario is divided as responsibility goals of the respective threat actors and expressed as their root Anti-goal nodes (**Req3**). Subsequently, the steps involved in the attack methods are expressed as hierarchical threat actor/agent anti-goals trees using AND/OR constructs (**Req4**), see Figure 1b. Unlike Attack trees, a threat actor/agent can have more than one anti-goal trees. That means, the deliberate threat/Attack goals are defined as the root nodes. The tree structured refinements reflects how the threat goals can be fulfilled.

The interaction dependencies are expressed similar to STS notation goal delegations and data exchange, except that here the intentions are towards manifesting a threat scenario (Figure 1c). These social and technical dependency interactions allow to comprehend the strategic plan as well as to trace the attack path between the disjointed anti-goal models (**Req5** and **Req6**). This allows to simplify the sophistication of APT by breaking the complex strategic scenario into multiple anti-goal models (**Req1**). Furthermore, we use threat propagation functionality in STS to represent the propagation of protection events such as defensive strategy goals to mitigate particular malicious goal (Figure 1d). Accordingly, the protection event propagation allows not only to link the protection goals to the anti-goals, but also it facilitates to view the protection impact on the whole Anti-goal tree model (**Req7**).

IV. Carbanak specified in Anti-STS

Our principal motive is to define structured representation to facilitate the modelling and understanding. We refer to the threat intelligence reports and adopt following three steps to facilitate the modelling of Carbanak APT case study:

A. Define threat actors and respective root Anti-goals

First is to identify the threat actors in the case study. The Carbanak APT involved different people with varying skills such as social engineering experts to make spear phishing, the technical hackers to build the malware, system administrators to operate a set of C2 servers, banking experts to design the transfers of money and mules to get the cash at ATMs. In addition, there were several interactions between IT systems like the C2 servers and the malicious processes that were running in the victim's computer. In Figure 2, we have defined the threat actors as separate roles with the different skillsets. This will facilitate to split the anti-goal refinements thereby reduce the complexity. Since APT includes an organized group of threat actors, we additionally considered a lead threat actor responsible for managing or even financing the group. We express this lead role as "Carbanak lead actor". The ultimate goal, which is stealing money, is the root Anti-goal node defined in the scope of carbanak manager agent.

B. Refine Anti-goals and identify interaction dependencies

After identifying the threat actors and their root goals, we need to refine them with the help of the attack methods and patterns employed in APT. In this process, we come across the dependency interactions. For instance, the anti-goal "steal money" of the carbanak lead actor can be refined (see Figure 2)

into three goals reflecting the three procedural phases executed in carbanak APT. The lead actor interact with other threat actors possessing necessary skills in order to fulfil some of its anti-goals. In Figure 2, we expressed this social interaction dependency as anti-goal delegation notation. The delegated anti-goal refinements are defined in the scope of the respective threat actors. These further refinements may give rise to new anti-goal delegation needs. For example, the threat actor responsible for social engineering may have to await the hacker to obtain the malicious files embedded with malware program code.

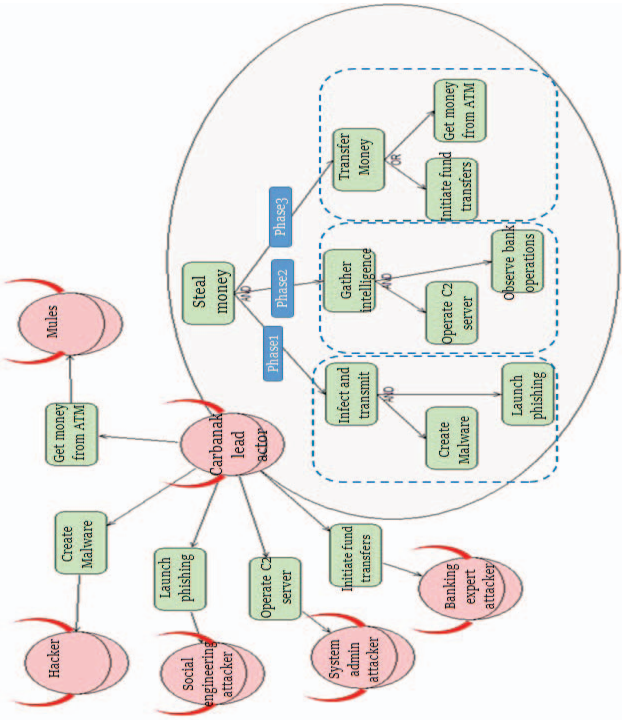

Figure 2:Anti-goal refinement of the Carbanak lead actor

In similar fashion, our Anti-STS model also facilitates to view the technical interaction dependencies between the system threat agents. In the threat intelligence report of the carbanak APT campaign, it is given that at some point the malware program from the infected machine establishes a connection with the command and control (C2) server (operated by "system admin attacker") in order to share the proxy settings as well as to download the "kldconfig.plug" file. This file seemed to have contained the names of the processes that needed to be monitored. This implies a technical interaction dependency which allows the attackers operating the C2 server, to gain remote access to the infected machines. Figure 3 depicts an example of such interaction. It is to note that this figure reflects only small portion of the anti-goal refinement of the malware program code.

C. *Specify the given control measures as protection events*

Finally, in the threat intelligence reports a security measure is proposed to detect and/or protect against the described APT. In our carbanak example, Kaspersky labs proposes to detect the malware by looking for a ".bin" file in folder **"..\All users\%AppData%\Mozilla\"**. Our model allows the expression of this information using the protection event notation, see Figure 3. In addition, the propagation of protection events facilitates to view the impact of the protection across the Anti-goal model.

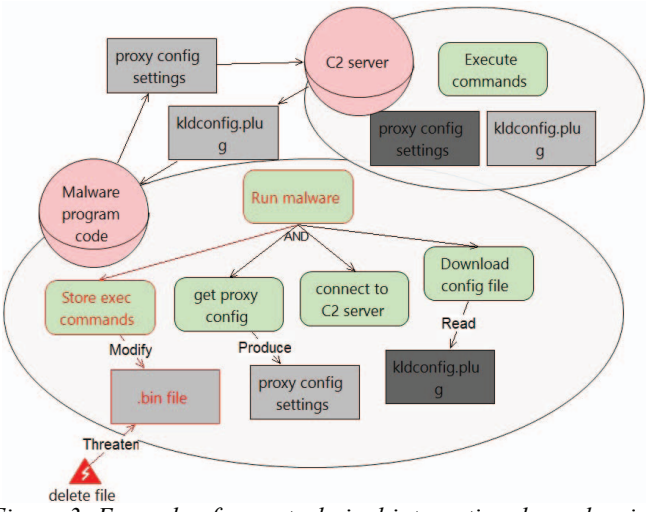

Figure 3: Example of some technical interaction dependencies

V. Discussion and Conclusion

In this article, we proposed an anti-social modelling approach to model APT as an anti-system. Our modelling approach, named as as Anti-STS, revolves around the core idea of expressing the social and technical dependencies between the threat actors in an APT campaign. It is worth noting that, although it may seem that our model is relevant to STS approach only, the driving motivation and the usability context of our Anti-STS are completely different and complementary. Our model is purely aimed towards developing a common platform to formally represent the threat intelligence on notable APT like attacks. This will permit the users to have a common modelling language to express the APTs which can be shared and reused with less ambiguities.

For future works we intend to improve our model to highlight and distinguish the varying attack patterns employed in APT campaigns. (**Req8** in Section II). Furthermore, it would be interesting to integrate multiple APT campaigns which will help the risk analysts to compare and analyse the different versions of carbanak APT.

Acknowledgment

This work is part of project IREHDO2 funded by DGAC and operated by DGA. The authors thank all the security experts at Airbus as well as the anonymous reviewers who helped us with their useful comments.

References

[1] G. De Santis, A. Lahmadi, and O. Festor, 'Modeling and analysis of Advanced Persistent Threats'.

[2] Matulevicius et al, 'Syntactic and semantic extensions to secure tropos to support security risk management.', 2012.

[3] Kaspersky Labs, 'The Great Bank Robbery: the Carbanak APT'. .

[4] 'STS-Tool | Tool for STS-ml modeling language'. [Online]. Available: http://www.sts-tool.eu/. [Accessed: 07-Oct-2016].

[5] A. Van Lamsweerde, S. Brohez, R. De Landtsheer, and D. Janssens, 'From system goals to intruder anti-goals: attack generation and resolution for security requirements engineering', vol. 3, 2003.