Modelling Duqu 2.0 Malware using Attack Trees with Sequential Conjunction

Peter Maynard, Kieran McLaughlin and Sakir Sezer
Centre for Secure Information Technologies, Queen's University Belfast, BT3 9DT, Belfast, U.K.

Keywords: Duqu 2.0, Attack Trees with Sequential Conjunction, SAND, Malware Analysis, Threat Modelling.

Abstract: In this paper we identify requirements for choosing a threat modelling formalisation for modelling sophisticated malware such as Duqu 2.0. We discuss the gaps in current formalisations and propose the use of Attack Trees with Sequential Conjunction when it comes to analysing complex attacks. The paper models Duqu 2.0 based on the latest information sourced from formal and informal sources. This paper provides a well structured model which can be used for future analysis of Duqu 2.0 and related attacks.

1 INTRODUCTION

Threat modelling is used to visualise threats to a system or process and provide a method to identify vulnerable areas which otherwise might have gone unnoticed. Threat modelling can be applied to many different circumstances ranging from computer networks, software life cycle, malware analysis and physical security. Not only can it provide the analyst with a visual representation, it is also possible to apply quantification methods to each possible threat path, resulting in a ranking of the likelihood that a specific vulnerability could be exploited. In most cases threat modelling will take the form of the abstracted high level steps an adversary could take when attempting to compromise a process. This is normally represented as a graph, a tree or a combination matrix, and, quantification methods are then applied on top of the model.

Typically there are two main approaches to threat modelling, there are formalisms which are derived from, or extend, the original threat trees, and formalisms based on attack graphs. Threat tree based approaches are designed with the perceived threat at the top of the diagram, which is referred to as the root node. Underneath are multiple paths and leaf nodes which an adversary could take to exploit the preserved threat. Attack graphs are based on graphs where all nodes are connected to each other and the values of the vertices are used to measure the quantitative values. There is an inherent problem when using a graph to model attacks, they are typically not able to scale well.

Duqu 2.0 was identified by Kaspersky Labs (Kaspersky, 2015) in 2014 and have confirmed that it is the re-emergence of the Duqu malware which was found in 2011 which was active for a few months before it was deactivated apparently by its creators. Duqu 2.0 was first found to be active about three weeks before the Iranian nuclear talks were to take place in the same physical location, with speculation (Schneier, 2015; Zetter, 2015) on Duqu being used for espionage by a nation state. Duqu and Duqu 2.0 share large amounts of code with the exception that Duqu 2.0 was compiled with newer tool chains, it also has many similarities in the way in which the code is structured. Duqu 2.0 uses a range of compression and encryption algorithms to avoid being detected, it also hijacks existing process's security clearances to prevent anti-virus software from terminating the process.

Contributions: In this paper we develop a model of the Duqu 2.0 malware based on the latest analysis of the malware. There is a lot of information regarding Duqu 2.0 but no single formal definition has been published, which can be used to deeply analyse Duqu's behaviour in greater detail. Most of the information regarding Duqu 2.0 is only in news reports and white papers, which detail only the technical details of how it works. Modelling Duqu 2.0 using a formalised threat model gives future researchers a base to develop new detection techniques and allow for application of quantification methodologies.

The paper is organised as follows: Section 2 explores some implementations of threat modelling within cyber-physical systems. Section 3 discusses existing threat modelling formalisations, and we

Maynard, P., McLaughlin, K. and Sezer, S.
Modelling Duqu 2.0 Malware using Attack Trees with Sequential Conjunction.
DOI: 10.5220/0005745704650472
In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pages 465-472
ISBN: 978-989-758-167-0

model Duqu 2.0 in section 4 before presenting some conclusions.

2 BACKGROUND

In this section we discuss the use of threat modelling in the analysis of recent attacks, and threat modelling specific to cyber-physical systems.

Bryes et.al (Byres et al., 2004) discuss the use of attack trees in assessing vulnerabilities in SCADA systems, the authors develop eleven models each with there own specific attack goal. The models identify vulnerabilities within SCADA protocols and in existing methods of ICS deployment. Although these vulnerabilities are now well-known, they were not at the time of publishing. Byres demonstrated that the use of attack trees is suitable for use within the industry and identify flaws with the formalisation. It also became evident that threat modelling was in need of more exploration. Since the paper was published in 2004 there has been an uptake in threat modelling.

Ten et.al. (Ten et al., 2007) again used attack trees to model threats to SCADA systems and have taken it a step further than Bryes (Byres et al., 2004). They have identified an analytical method to measure the vulnerabilities. The analytical method allows for systematically evaluating threats and countermeasures. One of the weaknesses which they identified by using Attack Trees is there is no way to model the sequence of the steps, which limits its usefulness. Though the attack trees approach works well for pentesting and studying security flaws.

Tanu and Arreymbi (Tanu and Arreymbi, 2010) model seven types of communication based attacks, such as replay, DoS, man-in-the-middle and command injection. Augmented vulnerability tree was able to model the communication based attacks, and identify issues with the current SCADA protocols. They recommended the use of message hashing for authentication and encrypted communication between all devices. Though the mitigation results are not novel, they were able to make use of threat modelling within a SCADA environment with a successful results.

Smith and Ma (Ma and Smith, 2013) have written about risk assessment in which they model the connections of the network based on the network traffic. This is done on a rule based system similar to firewall rules, once the network is modelled they identify what software and services are running on each node and map vulnerabilities to any CVE and CVSS found in a CVE database. They are then able to infer the most likely multi attack path using that information. This works well for mapping existing systems and performing a good audit of the network, and could be a very good first step for a penetration tester. However, this does not help to identify future or existing attacks, how they may be performed or how they can infiltrate the network.

Stuxnet is known as possibly the most advanced persistent threat ever seen, and it was targeting cyber-physical systems. Since then there has been more research in the field of SCADA and ICS security, than in previous years. Duqu/Duqu2.0, Havex, BlackEnergy, and the German steel mill attack are malicious malware which have been seen to specifically target critical infrastructure, without analysing these threats we will not be able to defend against them in the future. This is where threat modelling can help identify features to be used for defending our critical infrastructures.

3 EXISTING THREAT MODELLING APPROACHES

This section details four prominent threat modelling formalisations designed for specifically modelling attacks such as malware and network intrusions. Towards the end of this section we discuss the problems with each of them.

3.1 Time-dependent Attack Trees

Time dependent attack trees evaluate the probability of an attack as a function of time. The formalisation has been formally described in (Arnold et al., 2014). They make use of the sequential AND operator, as described in section 3.4 which allows for sequential operations. Due to the nature of the model, each node needs to be given specific time values, which is the expected length of time it would take for that step to be achieved. It uses the bottom-up algorithm to quantify the results. It is particularly difficult to model an attack and so the developers have used acyclic phase-type distribution (APH) expressions to simplify development of the models, they provide a prototype web-based interface for generating the APH expressions. It has been applied to a few test cases, and appears to be able to handle complex attacks such as Stuxnet.

3.2 Boolean Logic Driven Markov Processes

Boolean Logic Driven Markov Processes (BDMP) is a hybrid formalisation which is based on fault trees

combined with Markov graphs. Invented for use within the safety and reliability area (Bouissou and Bon, 2003) it was later applied to the security industry by (Pietre-Cambacedes and Bouissou, 2010) in 2010. Its goal is to find a better trade off between readability, modelling power and quantification capabilities with respect to the existing solutions particularly attack trees. BDMP's advantages over the traditional attack trees are its ability to use what they call triggers. Triggers allow modelling of sequences and simple dependencies by conditionally "activating" subtrees of the global structure. Because BDMP is based on fault trees there is an array of connections available such as AND, OR and PAND gates, this also gives the model the advantage that it is easy to understand and read. In (Kriaa et al., 2012) Kriaa models Stuxnet using BDMP. Quantification for BDMP is dependent on how the fault tree is modelled, and this allows for a very versatile set of metrics. Typical metrics include: overall mean-time to success, probability of success in a given time, ordered list of attack sequences leading to the objective, cost of attacks, handling of boolean indicator and so on. It is also possible to model defence-centric attributes which reflect the detection and prevention of the system already in place, this allows for a more realistic prediction of attack path. There is one tool (KB3-BDMP [1]) capable of developing BDMP models and performing analysis of the model, development and implementation of the tool was detailed in (Pietre-Cambacedes and Deflesselle, 2011).

3.3 CoPNet

CoPNet is a hybrid threat model which combines attack trees and coloured Petri nets. It was partially defined by (Bouchti and Haqiq, 2012), in this informal specification their case study is based on a SCADA network. The case study is a simple SCADA network with a 3-bus power grid which contains a HMI monitoring the three generators, they model the network and a range of possible threats then identify the most likely attack path using their quantification method. It uses attack trees to model the attack to help in simplifying the development and allow for importing existing models. Once an attack is modelled using attack trees, CoPNet has detailed a method which can convert attack trees into coloured Petri nets where they can then perform the threat analysis. The authors provide partially working tools available to develop a CoPNet model. When testing we we're unable to generate a usable results from the tools.

[1] http://researchers.edf.com/software/kb3-80060.html

3.4 Attack Trees with Sequential Conjunction

Attack trees with sequential conjunction (SAND) is an enhancement of attack trees, which were popularized by Schneier (Schneier, 1999). SAND enhances Attack Trees by defining the use of a sequential AND operator. This allows for the child nodes to be completed in sequence adding another level of complexity without losing the simplicity of attack trees and maintaining the advantages. SAND was defined in 2015 and has formally been described in (Jhawar et al., 2015), though this is the most formal definition of the operator it has been used by other formalisations previously. There is one primary tool which supports SAND, called ATSyRA (Pinchinat et al., 2014), a tool built on top of the Eclipse IDE.

3.5 Problems with Current Threat Modelling Approaches

To effectively model the Duqu 2.0 malware it was necessary to identify some requirements upon which to chose a formalisation. It needs to be easy to understand in both the raw and visual form. An effective formalisation must be able to represent sequential events or dependencies to be able to model a complex process such as Duqu 2.0. A practical formalisation also needs to support some form of quantification so the model can be used for analysis of the malware. It is also desirable to have a formal specification of the formalisation to ensure that our model is built to a correct standard.

Time Dependent attack trees upon first glance appear to meet all the requirements. They have been formally defined, support sequential operations and a working tool is available to help generate models. Though as the name suggests, the only quantification which it supports is based on time, and that the model has to be built using acyclic phase-type distribution (APH) expressions which abstracts the model, thus losing information, and reduces the readability of the model. These two points make it unsuitable for our application as we wish to develop a model which can be easily understood and provide a base for further quantification metrics to be applied.

BDMP has a similar ability to represent sequential operations by using triggers. It has been formally defined, though it lacks the readability of traditional attack trees. BDMP is a hybrid formalisation combining attack trees and Markov graphs, this combination requires the model designer to have a solid understanding of BDMP before they can start working on a model, as well as the modelling tool being depen-

dant on propitiatory software. Due to the high learning curve of BDMP over some of the other formalisations the authors feel that due to BDMP's complexity it reduces the likelihood of others continuing to use and share the produced models.

Visually CoPNet is easy to understand and because it is based on attack trees, its raw form is also easy to understand, but there is an extra step which is needed to convert from attack trees into the model. The conversion software is currently a prototype. This extra requirement, along with an informal definition of the formalisation, does not make it easy for other analysts to continue working with a model. Furthermore the informal paper (Kriaa et al., 2012) says that it can handle sequential operations but does not detail how it can be modelled.

Attack Trees with Sequential AND (SAND) are based on Attack Trees, which makes comprehending the raw and visual representations of the formalisation easier than the other formalisation. It is also a well known and widely used approach which has been previously applied to cyber-physical systems. With the addition of the Sequential AND operator, SAND allows modelling of sequential operations and complex systems. SAND has been formally defined which allows users to develop a model with reassurance that it can be expanded in future works.

4 MODELLING DUQU 2.0

Based on the analysis of available threat modelling techniques in the previous section, we have determined that SAND should be used to model the Duqu 2.0 malware. In this section we will introduce the basics of SAND and provide a detailed review of Duqu 2.0.

Attack trees are ordered to allow for systematically identifying different ways in which a system or process can be attacked. Attack Trees are ordered with the attacker's goal at the top, called the root node, and subsequent child nodes represent the attacker's sub-goals. The nodes are connected using disjunctive (OR), conjunctive (AND) and sequential conjunctive (SAND). The leaves of the tree represent the attackers actions.

An example model based on the formalisation (Jhawar et al., 2015) using SAND is shown in Figure 1, it details a file server offering ftp, ssh, and rsh services. The Attack Tree shows the ways which an attack can get root access. There are two ways either without authenticating (no-auth) or by authenticating (auth). The first case (no-auth) the user must gain privileges then perform a local buffer overflow

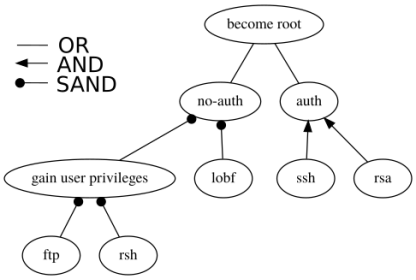

Figure 1: Example of the SAND formalisation.

attack (lobf). This is where the sequential AND operator is used, as the steps must be completed in order for the attack to succeeded. To gain user privileges, the attack must first exploit the FTP service so that they can upload a list of trusted hosts which *rsh* will used to allow authentication to the server. In this example SAND is executed in left to right order across the nodes, (note that later this paper it is executed top to bottom due to the size of the diagrams). The second method of becoming root is to abuse a buffer over in both the ssh daemon (ssh) and the RSAREF2 library (rsa). These nodes are linked with AND, as each of the steps can happen in any order.

Duqu 2.0 is primarily an information gathering and exfiltration malware, which has a lot of support for remaining hidden from detection. It runs completely within RAM to avoid being detected and it also leverages anti-virus detection system's defences to help remain hidden. It was first found on the network owned by Kaspersky spying on their activity. It had used three highly complex zero-day exploits. Not only is it able to hide from detection it has over one hundred plugins, supporting various functions, and the ability to encrypt and compress using a wide range of algorithms. Duqu 2.0 comes in two versions, *full* and *light*, full is around 18MB and contains all the plugins needed, the light version contains just the bare minimum with the ability to install plugins. One of the key features is its ability to create an internal proxy server for all the infected clients within the network, the traffic is then covertly transferred into and out of the network without raising suspicion.

We have modelled Duqu using the SAND formalisation. The next three sections will discuss the model in detail, which has been broken down into three manageable parts as follows: Figure 2, Initial Compromise and Lateral Movement, Figure 5, Execution of the Payload, and Figure 4, Command and Control and Plugin Operations.

4.1 Part A - Initial Compromise and Lateral Movement

It is believed that Duqu was delivered by a targeted spear-phishing campaign. When a victim received an email and they downloaded and executed a word document resulting in the, then zero-day, exploit CVE-2014-4148, which is a vulnerability in the windows TrueType font. The attacker crafts a custom TrueType font which, when it gets parsed by the operating system, allows the attacker to execute arbitrary code with the security permissions of the kernel, virtually unrestricted.

Once the malware has kernel access it attempts to propagate laterally through the network. Again, it is able to take advantage of another zero-day exploit (CVE-2014-6324) to gain domain administrator privileges using a 'pass the hash' exploit. Essentially

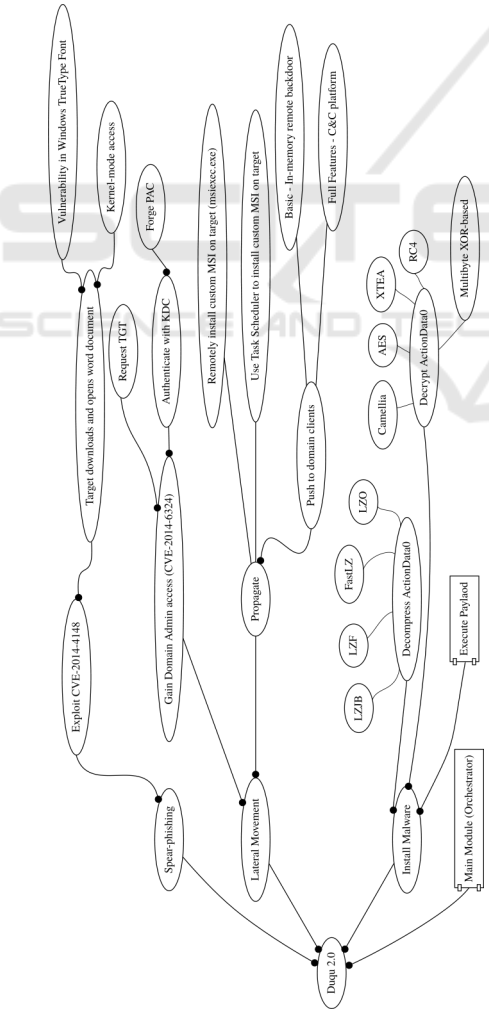

Figure 2: Part A - Initial Compromise and Lateral Movement.

it requests a token from the Kerberos authentication server, decrypts it and forges the security groups so that it looks as if the domain user is in the administrators user group. This gives the malware unrestricted administrative access to the domain controller and the rest of the network.

Once Duqu 2.0 has administrator access to the domain controller it begins to propagate itself across the network by remotely installing malicious a MSI file on domain clients. An MSI file is an executable installation file used by windows, Duqu 2.0 can create these executables which contain the necessaries to deploy itself on client machines, Figure 3 shows the MSI structure. It can install the MSI file remotely with 'msiexec.exe' or using the task scheduler. There are two versions of the MSI; one basic in-memory remote backdoor, and another larger version (18MB) which contains many of the advanced plugins. The MSI file is encrypted and compressed using a unique combination of algorithms, making it hard to apply signature detection upon the files.

4.2 Part B - Execution of the Payload

This section discusses Figure 5 which is focused on the payload of the malware.

The MSI's file structure, as shown in Figure 3, contains two executable blocks, one is 'Custom action dll' which is used to decrypt and decompress the payload called 'ActionData0'. ActionData0 also contains encrypted and compressed executable code used to deploy and manage the malware. Duqu is able to adapt to its environment by using different payloads, it can dynamically target CPU architectures as well as a specific process. Kaspersky was able to identify five payload containers each with similar configuration. In this paper we have gone into detail on the variation payload *Type L* with a focus on the Kaspersky anti-virus process as this has been the only process covered in such detail. The other payloads are similar to Type L, we have included the differences in the model, and briefly discuss them towards the end of this section.

It first attempts to locate and execute 'api-ms-win-shell-XXXX.dll', with 'X' being a decimal, the number is based on the machine/boot time. If it can find that it will generate a PRNG kernel object, this allows for the value to be unique and easily identified by another process if it uses the same number generator. If it already exists it will load the value located in it, and if not it will open a device driver and issue some IOCTL codes to the driver. This is used by another module further on.

It starts to search though all the running processes

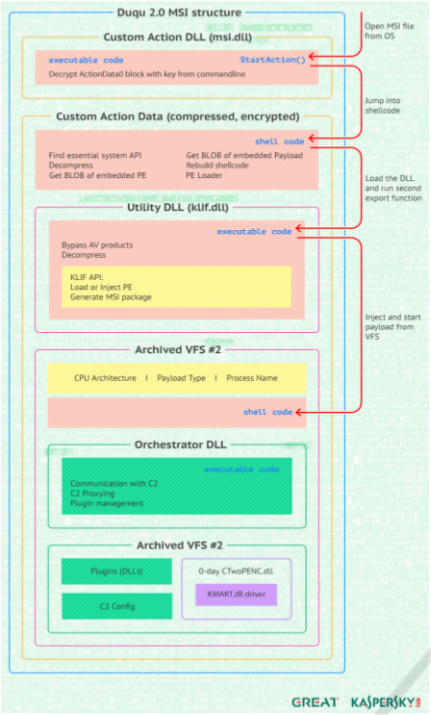

Figure 3: Kaspersky's breakdown of Duqu's MSI structure. (Kaspersky, 2015).

on the host for a matching string, in our model it specifically looks for 'avp.exe'. Which is the Kaspersky anti-virus process. Once it is located, it will attempt to gain kernel access using CVE-2015-2360, with the intention of loading the 'KMART.dll' driver with kernel level permissions. Loading 'KMART.dll' with high permissions is done so as to allow hijacking of the anti-virus' security tokens.

Now it iterates over a list of hard coded registry keys, with the intention of finding the location of installed security software, again in this instance we are specifically looking for 'avp.exe'. Once the location is found, it confirms that the file is executable by checking the environmental values resolve correctly, the file will open, and it begins with 0x5A4D (MZ). With the executable correctly identified it will map it to memory and patch it to jump back to 'kilf.dll' where it communicates with 'KMART.dll', which is a signed driver with kernel level permissions. Once all this is done, it now appears to the operating system and other processes that, Duqu is actually a process owned by 'avp.exe', the anti-virus. This effectively means that it will be ignored by the anti-virus software as it believes it is secure, and the defences which Kaspersky have built into their application will also be hijacked.

The four alternative payloads, G, I, K, and Q are very similar to L but with some slight differences.

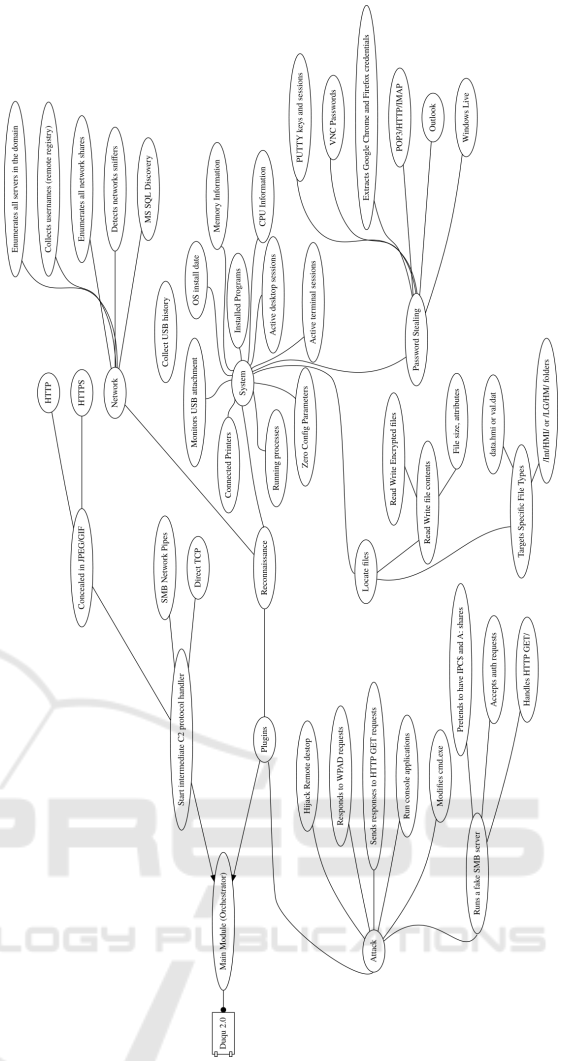

Figure 4: Part C - Command and Control and Plugin Operations.

Type G, is almost identical to L, but it skips hijacking the processes' security tokens and automatically selects a process with a known configuration. Type I, is the same a G, but it searches for the processes using a hashed value. Type K, runs from the context of the current process and blocks the threat until its complete and Q does the same but runs asynchronously.

4.3 Part C - Command and Control and Plugin Operations

This section will discuss Figure 4, which is focused on the command and control and plugins. Command and control instructions are handled with an array of possible protocols, HTTP and HTTPs, which covertly transmits the data as a JPEG/GIF to reduce the likelihood of being detected. Duqu can also use SMB net-

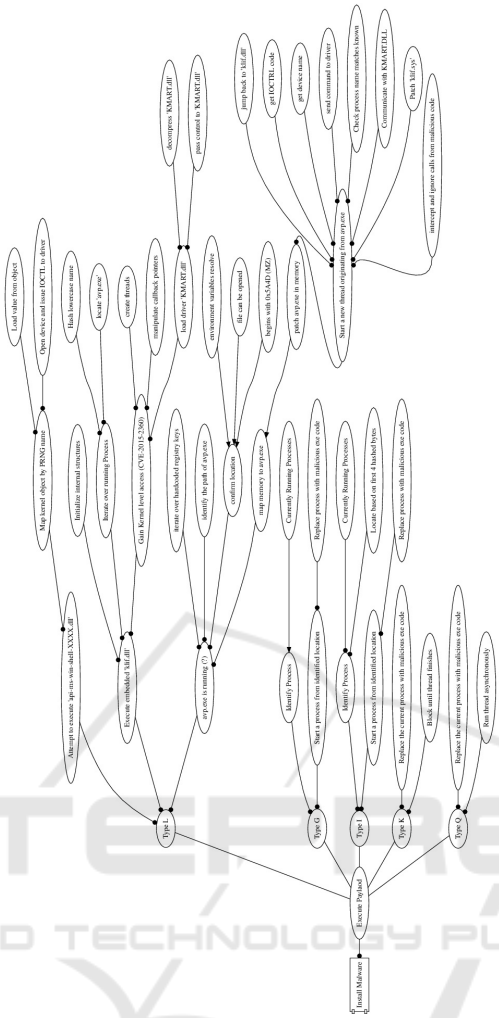

Figure 5: Part B - Execution of the Payload.

work pipes and generic TCP connections. It forwards remote desktop connections and controls all aspects of the malware remotely using these methods.

The plugins are separated into two groups, attack and reconnaissance. The attack section details some of Duqu's offensive plugins which range from remotely controlling a desktop, to responding to WPAD requests, and running a fake SMB server to steal login credentials. The reconnaissance describes features which can be used to collect information, and is divided into Network and System features. It is able to detect when a packet capture tool is being run and take precautions, it can also enumerate though the network identify machines. It can steal passwords from web browsers, email clients and the operating system. It also targets specific file types which are related to the operation of critical infrastructure.

5 CONCLUSION

In this paper we have modelled the fundamental mechanisms of the Duqu 2.0 malware in a robust and formally defined model. The model was built using the latest information on Duqu 2.0 gathered from various formal and informal sources. The paper has provided the security community with a complex model, which can be expanded to perform quantification analysis of malware. This paper has identified features which can potentially be used to detect and prevent such malware from appearing in the wild. The paper has identified a suitable formalisation which can be used for future research into malware analysis as well as network threat modelling. By using Sequential AND to model Duqu we were able to generate a model which is easy to work with and allows us to extract common features for future use.
The advantage of using SAND over existing formali-

sations is it's simplistic modelling of complex attacks. The textual representation allows for analysis of the model without specialist tools or complex syntax.

REFERENCES

Arnold, F., Hermanns, H., Pulungan, R., and Stoelinga, M. (2014). Time-Dependent Analysis of Attacks. In *Principles of Security and Trust (POST)*. Springer.

Bouchti, A. and Haqiq, A. (2012). Modeling cyber-attack for SCADA systems using CoPNet approach. In *2012 International Conference on Complex Systems (ICCS)*, pages 1–6.

Bouissou, M. and Bon, J.-L. (2003). A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. *Reliability Engineering & System Safety*, 82(2):149–163.

Byres, E. J., Franz, M., and Miller, D. (2004). The use of attack trees in assessing vulnerabilities in scada systems. In *in IEEE Conf. International Infrastructure Survivability Workshop (IISW 04). Institute for Electrical and Electronics Engineers*.

Jhawar, R., Kordy, B., Mauw, S., Radomirovi, S., and Trujillo-Rasua, R. (2015). Attack Trees with Sequential Conjunction. In *ICT Systems Security and Privacy Protection*, number 455 in IFIP Advances in Information and Communication Technology, pages 339–353.

Kaspersky (2015). The Mystery of Duqu 2.0: a sophisticated cyberespionage actor returns.

Kriaa, S., Bouissou, M., and Pietre-Cambacedes, L. (2012). Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments. In *2012 7th International Conference on Risk and Security of Internet and Systems (CRiSIS)*, pages 1–8.

Ma, Z. and Smith, P. (2013). Determining Risks from Advanced Multi-step Attacks to Critical Information Infrastructures. In *Critical Information Infrastructures Security*, number 8328, pages 142–154. Springer.

Pietre-Cambacedes, L. and Bouissou, M. (2010). Attack and Defense Modeling with BDMP. In Kotenko, I. and Skormin, V., editors, *Computer Network Security*, number 6258, pages 86–101. Springer.

Pietre-Cambacedes, L. and Deflesselle, Y. (2011). Security Modeling with BDMP: From Theory to Implementation. pages 1 – 8.

Pinchinat, S., Acher, M., and Vojtisek, D. (2014). Towards Synthesis of Attack Trees for Supporting Computer-Aided Risk Analysis.

Schneier, B. (1999). Attack Trees. *Dr. Dobb's*.

Schneier, B. (2015). Duqu 2.0. *Schneier on Security*.

Tanu, E. and Arreymbi, J. (2010). An examination of the security implications of the supervisory control and data acquisition (SCADA) system in a mobile networked environment. *Proceedings of Advances in Computing and Technology, (AC&T)*.

Ten, C.-W., Liu, C.-C., and Govindarasu, M. (2007). Vulnerability Assessment of Cybersecurity for SCADA Systems Using Attack Trees. In *IEEE Power Engineering Society General Meeting, 2007*, pages 1–8.

Zetter, K. (2015). Kaspersky Finds New Nation-State AttackIn Its Own Network.