On the Resilience of the Dependability Framework to the Intrusion of New Security Threats

Marc Dacier

Eurecom Institute
and Symantec Research Labs
marc_dacier@symantec.com
http://www.eurecom.fr/dacier

Abstract. B. Randell has been instrumental, with others, in the definition of the dependability framework. Initially thought of with a strong emphasis on accidental faults, it has paid more attention over the years to intentional ones and, thus, to classical security concepts as well. Recently, a couple of incidents have received a lot of attention: the Hydraq and Stuxnet worms outbreaks. They have been used to highlight what is being presented as a new and growing security concern, namely the so-called advanced persistent threats (a.k.a. APTs). In this paper, we analyse how resilient the historical dependability framework can be with respect to these sudden changes in the threats landscape. We do this by offering a very brief summary of the concepts of interest for this discussion. Then we look into the Hydraq and Stuxnet incidents to identify their novel characteristics. We use these recent cases to figure out if the existing taxonomy is adequate to reason about these new threats. We eventually conclude this chapter by proposing some future avenues for research in that space.

Keywords: dependability, resilience, security, intrusion detection, fault tolerance, attack, vulnerability, advanced persistent threat.

1 Introduction

Among his many contributions, B. Randell is famous for having participated, for more than 30 years, to a collaborative effort aiming at formalizing properly the basic concepts and taxonomy of dependable and secure computing. The most recent document presenting the results of this work dates from 2004 [2]. Quoting that document, one can read that *"[. . .] the aim is to explicate a set of general concepts, of relevance across a wide range of situations and, therefore, helping communication and cooperation among a number of scientific and technical communities [. . .]"*. The wide acceptance of these concepts, all around the world and across scientific disciplines, is a testimony of the success of this life-long investment. A **system** in that taxonomy is an entity that interacts with other entities. B. Randell likes to repeat that any good definition must be recursively applicable. It is therefore interesting to envisage that taxonomy as a *system* itself, interacting with the real world. Having adopted that viewpoint, the author wonders how *resilient* the taxonomy is. Our definition of **resilience** is the one given by J.-C. Laprie in [9], namely:

Definition 1. *The persistence of service delivery that can justifiably be trusted, when facing changes.*

In particular, we are interested in looking at the very recent changes we are facing in the security threats landscape. Are these so-called "advanced persistent threats," or APTs, disruptive events that make the taxonomy unfit for its purpose or, on the contrary, do the concepts, as they are, persist in delivering the service we expect from them?

In the following, we will address these points by, in Section 2, offering a brief summary of the concepts we need in this discussion. In Section 3, we describe the Hydraq and Stuxnet incidents, two recent examples of APTs. In Section 4, we revisit the notions of intrusion, attack and vulnerabilities by building upon the work done in the MAFTIA European project (see [3] and [10]). Section 5 concludes this document by offering some perspectives for future research based on the previous analysis.

2 The Basic Concepts

The purpose of this Section is to ensure that this document is self-contained but we invite the readers familiar with [2] to skip it as it is taken almost *verbatim* from it. Conversely, we strongly encourage the others to read [2] as it contains much more interesting material than what we can offer here.

Dependability is a global concept that subsumes the usual attributes of reliability, availability, integrity, maintainability, etc. The notions presented here are the results of a very long process initiated in the 80's that led to a book entitled *Dependability: Basic Concepts and Terminology* [8] that contained the very same text in five different languages: English, French, German, Italian and Japanese. As more people became aware of these concepts and start using them, a continuous effort has taken place to expand, refine and simplify them. [2] offers a synthesis of all these years of interactions between communities and defines **dependability** as an integrating concept that encompasses the following attributes:

– **Availability:** readiness for correct service
– **Reliability:** continuity of correct service
– **Safety:** absence of catastrophic consequences on the user(s) and the environment
– **Integrity:** absence of improper system alterations
– **Maintainability:** ability to undergo modifications and repairs

Correct service is delivered when the service implements the system function and a **failure** occurs when the delivered service deviates from the correct one. An **error** is the part of the total state of the system that may lead to its subsequent failure. The adjudged or hypothesized cause of an error is called a **fault**.

[2] derives eight elementary fault classes, according to eight basic viewpoints. Security is more directly concerned with a subset of these faults, the malicious human-made ones. They are grouped into two classes:

1. **Malicious logic faults** that encompass development faults as well as operational faults.
2. **Intrusion attempts** are operational external faults. The document notes that an intrusion can also be performed by systems operators or administrators who are exceeding their rights.

Four major categories of means to attain the various attributes of dependability and security are given ([2])

– **Fault Prevention** means to prevent the occurrence or introduction of faults.
– **Fault tolerance** means to avoid service failures in the presence of faults
– **Fault removal** means to reduce the number and severity of faults
– **Fault forecasting** means to estimate the present number, the future incidence, and the likely consequences of faults.

For historical reasons, most computer security solutions have been developed in, and still belong to, the family of *fault prevention* techniques. Clearly, encryption algorithms, access control systems and firewalls, to name a few, all aim at *preventing* malicious actors from committing their crimes. The, once prevalent, dream of an impregnable fortress applied to the cyberworld has vanished over the years and it is now commonly agreed that a second line or, even better, multiple lines of defense are required. This notion of *defense in depth* borrows ideas, concepts and techniques typically found in the *fault tolerance* family. **Fault Tolerance**, which is aimed at failure avoidance, is carried out via **error detection** and **system recovery**. *System Recovery* involves two distinct sets of techniques, namely **error handling** and **fault handling**. *Error Handling* aims at eliminating errors from the system state and can be carried out either by rollback, rollforward or compensation. *Fault Handling* aims at preventing faults from being activated again. This implies that some **diagnosis** has been done to identify and record the cause(s) of error(s). It can then be followed by some **isolation**, **reconfiguration** or **reinitialization** phase(s).

Every system design has to rely on some working assumptions. These assumptions are key to derive the optimal choices of techniques to build a dependable system. Until very recently, most security practitioners were mostly concerned by security breaches that would always start from a given vulnerability being exploited either i) to disrupt or halt service, ii) to access confidential information, or iii) to improperly modify the system. Until very recently, the notion of insider misusing his privileges, though well known and studied in some circles, had received little operational attention. The Hydraq and Stuxnet incidents have been a real eye-opener for many, revealing the existence of a new kind of threat coined the **Advanced Persistent Threats** (APTs). Before discussing whether they truly are novel or not, the coming Section presents what these incidents are and what new light they have shed on the security threats landscape.

3 Advanced Persistent Threats (APTs)

In [12], the authors note that *"the term Advanced Persistent Threat (* APT*) permeated our lexicon some time ago and is now used as standard terminology for threats that cannot be stopped"*. They express their concerns about the fact that *"a whole cottage industry has grown up around* APT *with marketing material being created specifically for it"* and they eventually conclude their paper by explicitly asking if the APTs are real or if they have been manufactured.

Before 2010, little evidence was available at hand to decide how to answer this question. The existence of a wikipedia page[1] on the topic highlights the visibility APTs are getting but the references present on the page — the oldest one dating from November 2009 ([4]) — do not constitute sound scientific proofs of their existence. Before the arrival of Hydraq and, later, Stuxnet, all we had were anecdotal cases in military environments such as, for instance according to [11], the Moonlight Maze, a Moscow-originated attack on American military services systems, and Titan Rain, a China-sponsored hit that aimed at U.S. military computers and those of its partners, such as Lockheed Martin^{TM}. But in January 2010, information about the Hydraq attack came out. A few months later, another attack, Stuxnet, made it clear that APTs were a real new threat that had to be taken seriously into account, a threat whose modus operandi did not obey to the working assumptions made by most to protect their systems. In the two following subsections (3.2 and 3.1), we present these two incidents and we discuss their novel characteristics in subsection 3.3

3.1 The Hydraq Use Case

Hydraq[2] is a targeted attack that is also currently referred to as Aurora, Google Attacks, and the Microsoft IE Vulnerability (advisory number 979352) [7] . The attack apparently began mid 2009 and continued through December 2009. It was first publicly disclosed by Google on January 12, 2010. The main goal of the attackers was to gain access to intellectual property of specific companies. Google described the attack as highly sophisticated and well coordinated. At least 30 companies were targeted.

The attack itself[3] was not very sophisticated: an unpatched Internet Explorer vulnerability was used as one of the propagation vectors. As a result of visiting a malicious page, a Trojan was installed on the computer as a very standard backdoor Trojan. As a matter of fact, that very same Trojan had been observed

[1] `http://en.wikipedia.org/wiki/Advanced_Persistent_Threat`

[2] `http://www.symantec.com/business/security_response/writeup.jsp?docid=2010-011114-1830-99`

[3] `http://www.symantec.com/connect/blogs/trojanhydraq-incident`

in another attack in July 2009. It does not use any anti debugging or anti analysis tricks. It uses some basic obfuscation in the form of spaghetti code on some of its components.

The sophistication of the attack lies in the fact that very few companies and very few individuals within each company have been targeted. Also, the modus operandi of the attackers was to, manually but remotely, take advantage of the credentials saved within compromised machines still connected to the company's network. Indeed, one of the components of this Trojan is based on VNC[4] code and has the ability to allow an attacker to control and stream a live video feed of the desktop of a compromised computer to a remote computer in real-time. As a result, the attackers were able to determine when the innocent user had left his machine unattended (e.g. by seeing that the screen saver has been activated for a certain amount of time). They were then able to remotely take control of the machine, as if they were sitting in front of it, in the middle of the company's intranet. As many users do store temporary credentials in their machines as well as sensitive information (e.g. the browser history and the associated session cookies), the malicious external user had a unique viewpoint to gather proprietary information without having to run any attack, without raising any alert since he was simply misusing the privileges of the legitimate user.

3.2 The Stuxnet Use Case

The Stuxnet incident is very different from the Hydraq one. Stuxnet was discovered in July 2010, but is confirmed to have existed at least one year prior to that and likely even earlier. The majority of infections were found in Iran. We refer the interested reader to [6] for a very thorough presentation of that threat, including detailed analysis of the propagation method and infection statistics as well as a detailed outline of the Stuxnet code architecture. Explaining the way Stuxnet operated lies outside the scope of the paper and we will only highlight here a very few elements that make this accident unique and relevant for our discussion.

According to [6], Stuxnet is a threat targeting a specific industrial control system likely in Iran. The ultimate goal of Stuxnet is to sabotage that facility by infecting specific types of Simatic programmable logic controllers (PLCs) devices to operate as the attackers intend them to, most likely out of their specified boundaries. Industrial control systems (ICS) are operated by a specialized assembly like code on the PLCs. The PLC devices are loaded with blocks of code and data written using a variety of languages, such as STL or SCL. The compiled code is an assembly called MC7. These blocks are then run by the PLCs in order to execute, control and monitor an industrial process. To access a PLC, some specific software needs to be installed. Stuxnet specifically targets the WinCC/Step7 software. With this software installed, the programmer can connect to the PLC with a data cable and access the memory contents, reconfigure it, download a

[4] `http://www.symantec.com/connect/blogs/hydraq-vnc-connection`

program onto it or debug a previously loaded code. In a typical ICS, like the one targeted by Stuxnet, the machines running the Step7 software were located in a so-called *air-gapped* network, that is a network that has no physical connection of any sort neither with the company intranet nor with the Internet.

To carry out their attack, the hackers had a number of technical challenges to overcome, namely ([6]):

- They needed to obtain the ICS's schematics since each PLC is configured in a unique manner
- To test their codes, they most likely had to set up a mirrored environment including the necessary software and hardware.
- Some of their malicious code needed to be signed by a trusted entity to run on the targeted ICS. They, quite likely, obtained the required certificates by physically entering the premises of two trustworthy companies to steal them.
- To carry out the sabotage successfully, they had to find a way to fool the operators to prevent them from shutting down the PLCs before the intended destruction. They manage to do this by compromising the Step7 software in such a way that they could intercept requests sent between the Step7 management console and the PLCs. Thanks to this, Stuxnet was able to modify the data sent to or returned from the PLCs without the operator of the PLCs realizing it.
- To reach the targeted *air-gapped* network, Stuxnet copies itself to inserted removable drives (e.g. USB keys). It is quite possible that the initial infection occurred thanks to an unknowing third party, such as an external contractor, who had to access the facility for maintenance or corrective tasks. Once in the network, Stuxnet used a total of four unpatched Microsoft vulnerabilities to propagate and a peer to peer mechanism to update itself within the LAN.

What is important to understand is that Stuxnet is an extremely complicated and sophisticated malware. As opposed to Hydraq, it was extremely innovative in its infection and propagation techniques. With the first PLC rootkit ever seen, its unprecedented number of zero-day vulnerabilities, its anti virus evasion techniques, its peer to peer updates system, its USB key-based infection vector and, last but not least, its ultimate goal, Stuxnet was a true eye-opener for many.

3.3 Discussion

In the preceding subsections, we have seen how different the Hydraq and the Stuxnet malware were. At this stage, one could wonder why they have been presented together as examples of a new threat dubbed APT! It is true that Hydraq exhibits a very low sophistication in terms of technical complexity. Stuxnet on the other hand is an incredibly large and innovative malware. However, they do have in common a number of underlying characteristics that explain why they both are good examples of APTs, Advanced Persistent Threats:

- **Reconnaissance:** In both cases, the malicious actors had to carry out some significant reconnaissance activities before launching their attacks. In the

Hydraq case, companies have not been targeted blindly and employees within these companies have also been carefully selected, based on the possible *return on investment* that they represent. It is even clearer in the Stuxnet case. Before launching the attack, a number of steps had to be carried out such as:

  - stealing digital certificates
  - obtaining the ICS's schematics
  - setting up a mirror environment
  - identifying external personnel likely to introduce the USB key for the initial infection
  - providing the key to that initial infection vector
  - etc.

- **Entry Point:** In both cases, the weak link misused to defeat the protective mechanisms put in place was less technical than socio-technical. In the Hydraq case, a number of specific users were lured to visit a web page hosting a zero-day attack that compromised their browser. In the Stuxnet case, quite likely, an unknowingly user plugged into his computer a USB key he has found or he had been given.
- **Unknowing insider:** In both cases, the malware took advantage of the privileges granted to an unknowing insider to carry out their tasks. In the Hydraq case, a large number of human beings were actively monitoring and using the compromised machines, misusing the privileges of their legitimate owners, thanks to the VNC[5] backdoor installed. In the Stuxnet case, the code injected into the Step7 software misused its granted privileges i) to rightfully modify the code on the PLC devices, ii) to intercept the information sent to and received from these devices, iii) to fool the operators by providing them with an erroneous representation of the real operational status of the ISC under attack.
- **Goal:** In both cases, the ultimate goal of the attack was very precisely articulated. The attackers knew what they were looking for and that goal was quite different from what constitutes the bulk of today's Internet attacks. In the Hydraq case, intellectual proprietary information such as source code or specific email accounts was what they were looking for. In the Stuxnet case, the likely goal was the physical and permanent destruction of a specific ISC, not an every day temporary DDOS attack against a web server!
- **Time:** In both cases, the duration of the attack and the persistence of the malicious actors was remarkable. In the Hydraq case, it is believed that the attack went undetected for most of the second half of 2009. In the Stuxnet case, its earliest sample variant has been seen in June 2009 [6] and Siemens reported that they were investigating reports of malware infecting Siemens WinCC SCADA systems on July 19, 2010. Furthermore, it is believed that it must have taken as much as 6 months to the attackers to set up their mirror site and test their code before being able to launch it.

[5] `http://www.symantec.com/connect/blogs/hydraq-vnc-connection`

– **Resources:** In both cases, the resources required to be able to prepare and launch the attacks were extraordinary. In the Hydraq case, most of the costs were induced by the fact that the attack was carried out manually by a, quite likely large, team of remote malicious human beings monitoring and taking control of the victims' machines to look for information within the targeted company's intranets. In the Stuxnet case, Symantec has estimated that five to ten core developers had to work during 6 months just to build the malware itself, not counting all other people needed to succeed in deploying it [6]

– **High profile target:** In both cases, the targets were security-aware and had implemented proper security defenses. In the Hydraq case, some of the companies attacked were among the ones who were very cautious about security and had all classical, apparently proper, measures in place. In Stuxnet, an air-gapped network was, among many other things, a protection technique that most people would have considered to be perfectly secure.

To make a long story short, one could argue that the success of these attacks lies in the fact that they have defeated their opponents by changing the rules of the game. By adopting a new modus operandi, by inventing a new strategy, they have, so to speak, invented a new game. To reach their goals, they have cheated and no one caught them. Indeed, their careful *reconnaissance* enabled them to circumvent the protection mechanisms in place and to identify the best *entry points* to carry out their attacks successfully. Once inside the targeted system, they took advantage of the privileges granted to *unknowing insiders* to reach a very well articulated *goal* defined in advance. By being able to spend as much *time* and *resources* as needed, they have managed to breach *high profile targets* and remain under the cover for several months before being discovered.

These threats really are game-changers. The true novelty is not so much that someone has invented them. It is that someone has actually implemented and carried out them for real. Not for cyber espionage or cyber war. Targeting civilian environments.

The reality of their existence emphasizes the requirement for security techniques that are not purely preventive. Fault prevention techniques can not stop these threats and we need to detect when an attacker has managed to get a foothold into a targeted environment. In the classical dependability context, one would think that fault tolerance techniques should be up to the task: by detecting error states likely to lead to failure and by taking corrective measures. Security practitioners would think of intrusion detection and countermeasures to do this. In the following Section, we take a closer look at these notions to see how effective they can be in dealing with these new APTs. If known intrusion detection techniques are up to the task, why should we bother?

4 Attacks, Vulnerabilities, Intrusions and Intrusion Detection

As we have seen in the previous Sections, APTs are well planned attacks, carried out persistently and in a stealthy mode. The possibility of preventing them from succeeding, even partially, is very low. It is therefore of prime importance to be able to detect when one is subject to such an attack. In this Section we try to specifically look at intrusion detection techniques, under the light of the dependability taxonomy. By doing so, we will see if that framework is resilient to the intrusion of these new threats. In other words, we are aiming to see if the concepts we have at our disposal enable us to characterize the problems at stake and, also, to provide suitable means to protect against them. Having done this exercise, we will identify a couple of possible research avenues that are worth exploring to be ready for the next wave of such attacks.

4.1 The MAFTIA Contributions

MAFTIA was a European funded project (IST-1999-11583) looking at Malicious- and Accidental-Fault Tolerance for Internet Applications. It produced, among other things, a deliverable proposing a design of an Intrusion-Tolerant Intrusion Detection System [3] which offered a precise definition of the terms attacks, vulnerability, intrusion and intrusion detection. These concepts have been refined into a final deliverable called "*Conceptual Model and Architecture of* MAFTIA" [10]. We offer in the following an abbreviated version of these definitions, focusing on the ones of interest to us here, and we will revisit them in the next Subsection under the light of the upcoming APTs.

An **intrusion** is defined in [10] as a deliberately-malicious software-domain operational fault that originates externally to the (technical) system boundaries. There are two underlying causes of any intrusion:

1. A malicious act or *attack* that attempts to exploit a weakness in the system
2. At least one weakness, flaw or *vulnerability*

From this definition, it is clear that, for an intrusion to occur, two distinct conditions must be met: i) a vulnerability must exist and ii) an attacker must have successfully taken advantage of it. This is the only case where an intrusion can exist and, therefore, be detected. Conversely, there will be no intrusion to detect in the following cases:

1. There is no vulnerability and no attack occurs. This is the ideal trivial case: the system is secure.
2. There is no vulnerability but an attack occurs. This is the most frequent case. In today's world, millions of attacks are launched against systems that are well protected. This is sometimes referred to as the background attack radiation noise.
3. There is a vulnerability but there is no attack trying to take advantage of it.

4. There is a vulnerability and there is an attack trying to take advantage of it but unsuccessfully. This can happen because of the incompetence of the attacker or because of intrinsic characteristics of the attack (e.g. race conditions).

We note that it can be interesting to know that an attack has been launched, even if it has not succeeded, i.e. even if no intrusion has occurred. In fact, this is what most intrusion detection systems do today. For instance, let us consider network-based intrusion detection systems. Almost all of them are capable of detecting port scans[6] and will raise alerts whenever they identify them. In such cases, are the *intrusion detection* systems actually *detecting intrusions*? It depends. If the port scan is detected in a supposedly highly secure environment, such as an air-gapped ISC, where no such thing should ever occur, then, yes, this alert highlights the fact that the system is in an error state that is likely to lead to failure. Furthermore, that state is an intrusion as defined here above: someone has already successfully used a vulnerability of the system to get access to it in order to launch this abnormal port scan. More than the port scan itself, the alert in such case warns the security officer about the compromised state of the system. However, if a similar port scan has been detected on the Internet against public machines, without knowing if the probed machines did contain a vulnerability or not, without knowing if the sent packets were really part of an attack or simply of a reconnaissance phase, one cannot determine whether an intrusion really occurred. Therefore, amusingly enough, in most cases, today's *intrusion detection* systems do **not** detect intrusions, as they have been defined in [10]. They deliver interesting information though. They detect *error states* but not intrusion states. They can inform the security officer, for instance, that there are attackers out there knocking on the door. Depending on the **security policy** in place, it should then be decided to initiate some *system recovery* actions to transform the system state into a new one that does not contain the detected error. In the case of the port scan detected on the Internet machines, it is quite likely that nothing will ever be initiated. In the ISC case, to the contrary, some *error handling* and, more importantly, *fault handling* techniques need to be applied.

Thus, *stricto sensu*, today's intrusion detections are, in most cases, not detecting intrusions but ongoing attacks or reconnaissance phases of an attack.

4.2 APTs and Intrusion Detection

If we take for granted that APTs, as described before, will be elaborate enough to find a way to get into the targeted system without being discovered and, then, will misuse granted privileges of an unknowing insider to carry out their malicious activities, what error detection mechanism could identify their presence? What

[6] A port scan occurs when a remote machine is probing the availability of a service on several other machines or of several services on a given machine. This is, typically, detectable by seeing a number of incoming requests that failed to get a valid response because of the non availability of the scanned services

do the notions of vulnerability, attack and intrusions, as defined here above, map to?

If we look at the problem from a microscopic and purely technical viewpoint, when the worm is in the apple, when the targeted attack has passed the defensive perimeters, in the worst case, there will not be any vulnerability to take advantage of or attack to be seen. In the Stuxnet case, it is the role of the Step7 software to communicate with and even reprogram the PLCs. There is no vulnerability or attack to be found when Stuxnet uses the very same method to reprogram the very same devices with a malicious intent. Similarly, in the Hydraq case, the remote malicious users are not running any attack to take advantage of any vulnerability when misusing credentials hosted by the machine they remotely connect to.

Nevertheless, the system, in both cases, has clearly been intruded and one would expect intrusion detection systems to play a role here.

It is worth pointing here to a typical example of intrusion given in [10]: "*[. . .]An insider abusing his privilege (i.e. a misfeasance): the vulnerability lies in the specification or the design of the (socio-technical) system (violation of the principle of least privilege, inadequate vetting of key personnel)*". In his seminal work on intrusion detection [1], Anderson proposes "*changes to computer audit mechanisms to provide information for use by computer security personnel when tracking problems*". In particular, he introduces the notion of audit reduction and the use of *some sort of statistical analysis of user behavior [. . . that . . .] might represent a way of detecting masqueraders*. This was later formalized as the so-called *behavior-based* intrusion detection techniques in the taxonomy by Debar et al. [5]. These techniques that detect deviations with respect to the normal behavior of a system under scrutiny have been extremely fashionable in the 90's but have rarely been transitioned[7] into the commercial solutions that have started flourish after the DDoS attacks of February 2000. Instead, the products preferred to adopt another paradigm, named the *knowledge-based* approach [5] (a.k.a. misuse detection techniques). These solutions rely on the knowledge of the attacks and of their symptoms to build detectors specifically tuned to recognize them. As a result, the domain evolved from a state-driven detection approach to an event-driven one, resulting in the paradox mentioned before when intrusion detection systems do not detect intrusions anymore but attacks, i.e. intrusion attempts. It seems that the rise of APTs is forcing us to reconsider the original approaches. If the attackers become clever and stealthy enough to hide all their attempts, all we have left to look at is the system state, hoping that it will deviate substantially enough from the normal one to let a detector catch the difference. In other words, behavior-based approaches seem to have a bright future before them.

The problem, of course, is that these approaches are able to detect that the system is in a state that someone considered to be suspicious enough to be re-

[7] However, we have to acknowledge the fact that very interesting results have been obtained by a number of, mostly academic, researchers. These approaches are also often referred to as *anomaly detection* techniques

ported. They do not necessarily know how we ended up in that state. They have no notion of causes of the errors, no notion of attack or vulnerability. They are pure error detection mechanisms and, as explained in Section 2, to avoid failure, error detection techniques must be accompanied by system recovery ones. This includes error handling and fault handling mechanisms. Unfortunately, these are areas where little effort has been invested so far, in the security space, but which APTs seem to force us to address. It is not clear though what kind of rollback or rollforward mechanisms could be implemented when an intrusion detection system detects that proprietary information has leaked out. These are certainly interesting challenges for the future. Similarly, the fault handling domain, as defined within the dependability concepts, encompasses different kinds of techniques (diagnosis, isolation, reconfiguration, reinitialization) that have barely received any attention within the security community. It would be very interesting to see how the body of knowledge accumulated in these areas could, or could not, be applied to the security realm and help in coping with these new threats.

5 Conclusions

In this paper, we have looked at some of the classical dependability concepts under the light of very recent incidents that have been called advanced persistent threats or APTs. We have outlined the key novel characteristics of these threats and have emphasized to what extent they are a game-changer for the security community.

We have shown that these new threats are forcing us to reconsider the seminal ideas that had led to the creation of the intrusion detection domain. By mapping intrusion detection to error detection, in the dependability taxonomy, we have highlighted a number of families of techniques that have not received enough attention to enable the implementation of effective fault tolerant techniques against APTs, i.e to avoid failures due to these new threats.

To conclude, we acknowledge the novelty of these threats and the influence they are likely to play on the security research in the coming years. The dependability concepts prove to be resilient to this abrupt change in the security threats landscape. They are adequate to reason about them and, more importantly, they help in identifying research directions where novel contributions are sorely needed.

Acknowledgements. The ideas and notions reported here are the results of numerous fruitful interactions with many colleagues, in particular Dominique Alessandri, Tom Anderson, Jean Arlat, Algirdas Avizienis, Christian Cachin, Eric Chien, Michel Cukier, Hervé Debar, Yves Deswarte, Klaus Julisch, Mohamed Kaâniche, Karama Kanoun, Carl Landwehr, Corrado Leita, David Powell, James Riordan, Bill Sanders, Robert Stroud, Olivier Thonnard, Michael Waidner, Andreas Wespi. Special thanks go, of course, to Brian Randell and the greatly missed Jean-Claude Laprie.

References

1. Anderson, J. P.: Computer Security Threat Monitoring and Surveillance. Technical report, 1980.
2. Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, January-March 2004.
3. Dacier, M. (ed.): Design of an Intrusion-Tolerant Intrusion Detection System. Deliverable D21 of the European funded project MAFTIA (IST-1999-11583), January 31, 2003, 111 pages
4. Daly, M. K.: Advanced Persistent Threat (or Informationized Force Operations). 23rd Large Installation System Administration Conference (LISA), Usenix, November 4, 2009, Baltimore, MD, USA.
5. Debar, H., Dacier, M. and Wespi, A.: A revised taxonomy for intrusion-detection systems. Annals of Telecommunications, Vol. 55, No. 7-8, pp.361-378, DOI: 10.1007/BF02994844, Springer-Verlag, 2000.
6. Falliere, N., O Murchu, L. and Chien, E.: W32.Stuxnet Dossier. Symantec White paper, V.1.4, February 2011, 68 pages, available online at `http://www.symantec.com/connect/blogs/w32stuxnet-dossier`
7. Ferrer, Z. and Ferrer, M. C.: In-depth Analysis of Hydraq, The face of cyberwar enemies unfolds. CA ISBU-ISI white paper, March 12, 2010, 37 pages
8. Laprie, J.-C. (ed.): Dependability: Basic Concepts and Terminology. Springer Verlag, 1992.
9. Laprie, J.-C.: From Dependability to Resilience. The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, Fast Abstract session, June 24-27, 2008, Anchorage, Alaska, USA,
10. Powell, D. and Stroud, R. (eds.): Conceptual Model and Architecture of MAFTIA. Deliverable D21 of the European funded project MAFTIA (IST-1999-11583), January 31, 2003, 111 pages
11. Somaini, J.: How to Combat the Cyber Espionage Threat. Industry Perspectives News article, Symantec, available online at `http://eval.symantec.com/mktginfo/enterprise/articles/b-article_how_to_combat_espionage_threat.en-us.pdf`
12. Treadstone: The mythical Beast That Hides in Your Closet. White paper available online at `http://www.treadstone71.com/whitepapers/TheMythicalBeastThatHidesinYourCloset.pdf`