From Situation Awareness to Action: An Information Security Management Toolkit for Socio-technical Security Retrospective and Prospective Analysis

Jean-Louis Huynen and Gabriele Lenzini
SnT - Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg, Luxembourg
{*firstname.lastname*}*@uni.lu*[*]

Keywords: Socio-technical Security, Information Security Management and Reasoning, Root Cause Analysis.

Abstract: Inspired by the root cause analysis procedures common in safety, we propose a methodology for a prospective and a retrospective analysis of security and a tool that implements it. When applied prospectively, the methodology guides analysts to assess socio-technical vulnerabilities in a system, helping them to evaluate their choices in designing security policies and controls. But the methodology works also retrospectively. It assists analysts in retrieving the causes of an observed socio-technical attack, guiding them to understand where the information security management of the system has failed. The methodology is tuned to find causes that root in the human-related factors that an attacher can exploit to execute its intrusion.

1 INTRODUCTION

Even a 'secure' system can turn out to be vulnerable when attackers target not the system's technical security mechanisms e.g., the cryptographic protocols, but the system's users.

In such situations, security cannot be achieved purely by technical solutions because of its dependency on the non-technical qualities of the system, such as the system's usability and the system's functional design. Thus, achieving security must consider human-related factors, like the cognitive and psychological traits that drive human behavior and the people's abilities to interact with information and communication technology.

Failing to understand this holistic complexity leads to the deployment of systems which are left at the mercy of attacks of socio-technical nature. And there is no rhetoric in this warning: the impact of such attacks is already huge. According to the Verizon's 2015 Data Breach Investigation Report that reports on the 80,000 security incidents that occurred in 2015, people is 'the common denominator across the top four patterns that accounts for nearly 90% of all incidents' (Brumfield, 2015). These figures call for a better understanding of the impact that humans have on security, and prove the need of measures more effective in reducing the risk of socio-technical attacks.

It is better to clarify that we are not claiming that organizations are unaware of or that underestimate security risks. Companies invest in Information Security Management (ISM) to keep attacks under control. They assess risks regularly, and implement measures that they deem appropriate. But in contexts where the human behavior is significant for security, predicting the effect of mitigations only reasoning on technical grounds may result in a false sense of security. For instance, a policy that force users to change passwords regularly, supposedly protecting from password theft, may nudge users to start using easy-to-remember pass-phrases (Adams and Sasse, 1999) which are also easy to guess. A policy that seems successful may be effective only thanks to 'shadow security' (Kirlappos et al., 2014), a spontaneous behavior that people take in opposition to an otherwise ineffective policy.

So, what reasoning can one resort to unveil a system's socio-technical weak points and what remediation can one apply to effectively strengthen security?

Addressing this question requires a change of perspective. Human users and the untangled factors that an attacker can manipulate to exert influence on people's behaviours should be included in the security best practices, analysis, and mitigation strategies.

This seems not to be such a revolutionary thought at least when we look at other disciplines. In *safety*, incident analysts follow practices to achieve similar

[*]This research received the support of the Fond National de la Recherche (FNR), project Socio-Technical Analysis of Security and Trust (STAST) I2R-APS-PFN-11STAS, and of the SnT/PEP-security partnership project.

Huynen, J-L. and Lenzini, G.
From Situation Awareness to Action: An Information Security Management Toolkit for Socio-technical Security Retrospective and Prospective Analysis.
DOI: 10.5220/0006211302130224
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 213-224
ISBN: 978-989-758-209-7
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

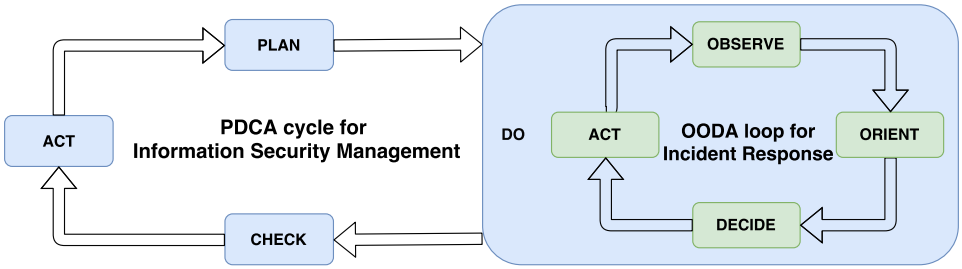

Figure 1: The PDCA process in the ISO 27001:2005 with the OODA loop for incident response proposed by Schneier.

goals i.e., understanding the root cause of events that involve human errors. Here, it is common to adopt a waterfall view of the universe (Anderson, 2008) where top-down approaches are followed to find out the causes of unwanted events, where bottom-up approaches are employed to foresee the negative consequences of a design choice, and where the identification of the different factors that foster an adverse outcome is used to set priorities and to optimize the efforts in defining and applying remedies.

In principle, such practices could be applied in security as well. An organization following those practices could improve its situation awareness and orient better its efforts in mitigating risks. Unfortunately, as we explain in § 4, applying in security off-the-shelves methods conceived for safety does not bring the desired result. We need to rethink and adapt them.

Suspending for a moment the discussion about the character and the difficulties of this adaptation, we believe that such a migration is possible, and this paper discusses and implements a way to do it. We refocus the study of security by centering on the impact that users have on a system, and we propose our methodology as part of the ISM life cycle.

Aim and Contribution. Elaborating on our previous work (Ferreira et al., 2015), we discuss the need of a root cause analysis in security (see §2 and 4) and present a two-way methodology along with a tool to analyse the impact of human users on security (see §5). Used retrospectively, the methodology helps an analyst investigate a security incident by clarifying how the attack could have pushed users to perform hazardous actions. Used prospectively, it supports an analyst investigate a system's resilience against specific socio-technical attacks and threat models so revealing vulnerable socio-technical factors that could help an attacker in its malicious enterprises.

This capability to identify the social and the technical factors that may have contributed to the success of an attack is inspired by Root Cause Analysis (RCA) techniques found in safety, in particular in one called Cognitive Reliability and Error Analysis Method (CREAM). In this work, we augment it with subsidiary capabilities (see §4), the most important of which is the ability to *generalize the knowledge* that an analyst gains inspecting a security intrusion. This knowledge is structured in such a way that it can be reused to analyse another system and to identify its socio-technical weaknesses. Levering on (Ferreira et al., 2015), we still call the methodology Cognitive Reliability and Error Analysis Method for Socio-Technical security (S·CREAM).

We also developed a tool to assist analysts follow our methodology, which we call *S·CREAM Assistant*. We exemplify its use in the analysis of the security of a One Time Password (OTP) solution: the Yubikey USB security token (see §6).

Our tool is meant to be accessible on-line. Its reasoning process and its knowledge base of vulnerabilities are designed to be available to anyone who wishes to integrate our methodology in an ISM system. Besides, its knowledge base is designed to grow as more retrospective analyses are performed. However, at the moment, S·CREAM and its tool are still in a proof-of-concept phase so in §7 we discuss limitation of the current version and what we need to do to make it a fully operational.

2 ON INFORMATION SECURITY MANAGEMENT

There are different processes that organizations can adopt in order to structure their ISM efforts. The ISO 27001:2005 standard (International Organization for Standardization, 2005) recommends organizations to follow a cycle called *Plan, Do, Check, Act (PDCA)* (Ishikawa and Ishikawa, 1988). The cycle guides organization in assessing information security risks and formulating security policies, and the meaning of the cycle's steps is as follows:

- **Plan:** list assets, define threats, derive the risks posed by these threats on the assets, and finally define the appropriate controls and security policies needed to mitigate the risks;

- **Do:** conduct business operations while implementing the security controls and enforcing the security policies defined in 'Plan';
- **Check:** check that the ISM system is effectively mitigating the risks without impeding the business operations i.e., detect defects in the ISM system.
- **Act:** correct to the defects identified in 'Check'.

In the PDCA cycle is the 'Do' phase where the organization operates. This is also the phase where attacks can occur and, subsequently, where Incident Response fires up. Schneier, in (Schneier, 2014), argues that Incident Response can also be seen as a loop, called *Observe, Orient, Decide, Act (OODA)* (Boyd, 1995), in which organizations engage in four steps: (i) **Observe:** observe what happens on the corporate networks. This is a phase of collection of data in which, for instance, Security Information and Event Management (SIEM) systems are used; (ii) **Orient:** make sense of the data observed within its context of operations also by processing the pieces of information gathered from previous attacks and threat intelligence feeds; (iii) **Decide:** decide which action should be carried out to defend against an attack, at best of its current knowledge; (iv) **Act:** perform the action decided earlier to thwart the attack. Figure 1 shows the PDCA cycle with the OODA in its 'Do' step.

Organizations engaged in such a process usually implement two types of analyses: a *prospective analysis* in the 'Plan' step to predict how their security measures are expected to protect their assets; and, usually after an attack, a *retrospective analysis* in the 'Check' step to identify the reasons of why their ISM has failed.

Shortcoming of Current ISM Processes. The existence of methodologies and tools to assess risks (e.g., OCTAVE Allegro (Caralli et al., 2007)) help company implement a PDCA process as we described, but there is no method that help them foresee what impact planned-checked-acted mitigations will have on the system's actual security. Indeed, introducing a security control to contain a risk (e.g., in the 'Act' step) may introduce new socio-technical vulnerabilities because of unforeseen interactions between the newly introduced controls and the organization's employees, business processes, and existing security measures.

Additionally, in the case of a security breach, there is lack of methods (in the 'Check' step) encompassing *human-related aspects of security* that could be used to identify the root cause of ISM systems failures.

The operational phase (i.e., the 'Do' step) is also not tuned to consider socio-technical aspects of security. Indeed, if organizations use Security Information and Event Management sytems in their Security Operations Centers to monitor their networks, they don't really consider the human-related aspects of their operations. For instance, when an company is hit by a 'fake president' scam —a fraud consisting of convincing an employee of a company to make an emergency bank transfer to a third party in order to obey an alleged order of a leader, the president, under a false pretext— the organization can defend itself by focusing on the technical aspects of the attack (e.g., by blocking connections to a ranges of IPs). However, the social aspects of the attacks (i.e., that people does not recognize the phishing and falls for it) remain, while the attacker can easily adapt to the additional security controls and persist in using the same social engineering tricks i.e., phishing people but in a different kind of scam. Security practitioners mainly rely on user awareness campaigns to cover the social side of these family of attacks but we claim that identifying the reasons why employees fall for the scams would be more effective. Indeed, it could be that the phishing campaign exploits loopholes in the interactions between the organization's security policies and organization's business processes (hence employee's primary goals). Identifying clearly these reasons could help propose additional remedies that focus not on the technical, but on the social aspects of an attack. These new remedies ought to cause the attacker the hassle of adapting its behaviour and its attack instead of only fiddling with technical details. We believe that this strategy could prove beneficial to the defense as it forces the attacker to climb up the 'pyramid of pain' (Bianco, 2014).

Another deficiency that a ISM may suffer is the lacking of ways to learn from past failures in applying the remediation strategies. Indeed, without a proper way to determine the root causes for past failures, organizations are doomed to repeat ill-considered decisions and lose energy correcting these.

For all these reasons, we believe that security practitioners should have access to additional methods that could help pondering the consequences of their security choices, the technical and the social alike, to prevent the recurrence of security failures.

3 RELATED WORK

Among the numerous of works on Root Cause Analysis (RCA) in safety incidents we comment only those in relation to the information security domain. Those which we have found in the literature use RCA in specific situations: the process they follow is not flexible

Figure 2: The process usually used for a Root Cause Analysis.

enough to work in other and in general contexts. For instance, Coroneo *et al.* (Cotroneo et al., 2016) implement a strategy for the automated identification of the root causes of security alerts, but what their offer is a fast algorithm to timely response to intrusions and attacks which is not applicable to analyse general security incidents. Another example of highly specific application is a work that customize the search for root causes in software failures (Kasikci et al., 2015). It supports software testing with an automated debugging that provides developers with an explanation of a failure that occurred in production but, still, this is not a solution that can be generalized to other security incidents.

A completely different category of works, which, again, we represent here by picking a single work in the class i.e., (Schoenfisch et al., 2015), is that proposing more efficient RCA's reasoning algorithms. Works in this class to not relate with understanding socio-technical causes. Rather they focus on improving the performances of existing methodologies but not on extending them to work in security.

Our search for related work seems thus more insightful not in what it has found but rather in what has not been found, that is, for the lack of works attempting to migrate in security well established RCA methodologies with the aim of helping as widely as possible security practitioners. At the time of writing (October 2016), we have found no significant and related article that address this problem, nor have we found RCA in security when human are involved. Of course there is a plethora of research that stress the need of keeping the human in the security loop (see e.g., (Noureddine et al., 2015; Beautement et al., 2016)) as there are plently of works in field of socio-technical security, usable security, human factors in security, and similar topics. For reason of space, this list is too long to be considered here but, in the best of our understanding, we were not able to find in those works methodologies that could help security practitioners in a RCA of socio-technical security incidents.

In summary, our analysis of the state-of-the-art shows that either we were unable to find representative pieceworks of research that relate with what this paper presents or the research questions and the challenges that this paper tries to address are fact original.

4 RCA IN SAFETY & SECURITY

In safety, the objective of a RCA process is to identify the cause(s) of an incident. The process will help gain the knowledge necessary to operate on the factors that have caused the event and to introduce solutions (e.g., re-designed interfaces or guidelines to operate the system) to impede the event from happening again.

Following a RCA process typically requires four steps (see Figure 2): (i) *Data collection and investigation*, to produce a description of the incident; (ii) *Retrospective analysis*, to inspect the incident in search for the causes that have triggered the incident — but how the analyst conducts this investigation and how much the outcomes rely on the analyst's experience depends on the specific RCA technique used; (iii) *Recommendations generation*, to produce a list of recommendations whose goal, when implemented, is to avoid that the incident reoccur; (iv) *Recommendations implementation* to obtain a safer system, free from the caveats that caused the incident.

Following this process, an incident's causes are investigated thoroughly even when it can be attributed to 'human error'. In his accident causation model, Reason (Reason, 1990) shows that 'human errors' are *active failures* that, when combined with *latent failures*, can transform a simple hazard into an accident. Thus, it is rare that a person, despite liable for an accident, is blamed as a conclusion of step (ii); rather the root cause is found in the complex interplay among the human, the system, and the context where human and the system operate, elements that are considered fertile ground for 'human errors' to happen.

4.1 From Safety to Security

Such a way to look at users as potential victim of a system design's deficiencies is advisable also in security. Too often, human errors in security are seen as inevitable, the end of the causative chain. In the Annual Incident Reports 2015, by ENISA, published in September 2016, human errors are pointed out be the 'root cause category involving most users affected, around 2.6 million user connections on average per incident' (ENISA, 2016). The conclusion that humans are at the root of all such incidents is worrisome but not helpful. It does not suggest how to improve security without removing the humans. And in the lack

Figure 3: S·CREAM's overall process.

of a comprehensive understanding of the reasons why systems and processes allow humans to be induced in security critical errors through which the system is being attacked, the problem of reducing human-caused insecurity incidents remains.

That said there are two neat differences between the RCA currently used in safety and a potential (retrospective) RCA as it should be used in security and the they are both about defining when to stop the search for causes. First, the search for root causes should not end to the attacher who, in security, is the obvious root of all evil. Second, when humans seem to be responsible of the incident, the search should not stop and point the user as the cause of the incident either, at least, not without looking also for the *triggers of human behaviour* that the system could have left to the control of the adversary. Such quest may reveal that the blame is, at least in part, on the system and not on its users. An example is when users click on poisoned links. Many security policies utter that clicking on links is a bad habit, but users, who are daily stormed by trusted and untrusted mails most of them carrying links, hardly can discern foes from friends. So perhaps the cause should be looked in the system's failing to authenticate an email's source or in the reasons why people fail to recognize friends from intruders that 'pretext' to be friends but not in the 'users' as such.

The retrospective analysis of the RCA in safety is not the only methodology that has a potential application in security.

The safety field also make use of techniques to predict the performance of systems that are going to be operated by humans. Predicting how an event can unfold is highly dependent on the description of the context, tasks, and failure modes. Potential paths that an actual event can follow are usually represented in binary trees called event trees (see THERP for instance (Swain et al., 1980)), where branches represent what may happen when an event (a leaf) succeeds or fails. Eventually, probabilities are computed for each outcome and recommendations are produced to enhance the reliability of the system.

This prospective approach is used in Human Reliability Analysis. It relies 'heavily on process expertise to identify problems and its methods themselves or expert estimation for quantification' (Boring, 2012). The overall process for a prospective analysis follows the same processes introduced earlier for an RCA, that is Figure 2, except for step (ii) which is called *Prospective analysis*.

However, there are a few key differences between the approach in safety and an approach that we devise in security. They emerge preponderantly and make migrating the existing retrospective and prospective techniques from safety to security be not a straightforward task. Such differences, which bring up several challenges that need to be addressed and resolved, emerge in the four steps of the RCA. Table 1 summarizes the differences which raise a five major challenges: (C_1) *Addressing the lack of knowledge and structured data:* The challenge is to compile and format factual information about the investigated attack to allow for the RCA to be performed, to describe what the attacker does and what are the attacks effects on the user and on the systems security. Furthermore, The RCA should provide precise information regarding the data to collect. (C_2) *Investigating Attacks*: The RCA for security must output a set of contributors and human-related factors that are likely to explain the success of attacks, or potential attacks. The new analysis should safeguard against one inherent shortcoming of RCA: the possible lack of objectivity. (C_3) *Creating reusable knowledge*: To integrate with existent computer securitys techniques, the RCA technique should provide direct links between the attackers capabilities and their effects on a systems security. The challenge is to be able to augment a said threat model with capabilities that an attacker can gain by performing user-mediated attacks allowed by the threat model. (C_4) *Match patterns of known attacks*: The RCA, in addition to the retrospective analysis of past attacks needs to provide a socio-technical security analysis where, from a systems description, socio-technical vulnerabilities, along with their contributors are listed. (C_5) *Being flexible*: The new method should be flexible enough to adapt to new threat, attacks, and technologies.

5 OUR PROPOSAL: S·CREAM

Figure 3 shows the steps of the process that we propose to address the five challenges. We describe each steps separately, hinting where necessary to the tool that we have implemented to support the execution of the steps.

Table 1: Key differences in safety and security.

	Safety	Security
Data Collection & Investigation	There is an established process to collect structured evidence for root cause analysis.	This process is not well-established; data are often unstructured and the information is often scattered across multiple actors.
	There are no malicious actors. Incidents happen because of general malfunctioning.	Incidents are caused by attackers whose skills and capabilities may be subtle and even unknown.
	Accidents to be investigated usually take place in well-known and well-defined settings	We face much more heterogeneous contexts, furthermore the incidents can still be unfolding at the time of analysis.
Analysis	RCA techniques are widely used and the human component is a central part of practices	The use of RCA methods is often advocated but lacks human-related insights.
	A human error is a well defined concept that can be the starting point of an analysis.	The human error is considered a systems failure mode that does not call for investigations.
	There is always some root cause that can be isolated for an incident.	The root cause of the success of an attack is always the attacker, therefore we are interested in all the factors that contribute to the success of attacks.
	The analysis begins from the terminal point of failure: the observable incident.	An attack/incident can be an intermediate step leading to other attacks/incidents. Therefore we might not be able to observe the factual consequences of an attack/incident on a system.
Recommendation Generation	Removing the root cause prevents the incident from reoccurring.	Since the root cause is the attacker, technical controls can be applied on to reduce the attacker's capabilities. Socio-technical controls can be applied on the human contributors.
	An adverse event, being coincidental, may never reoccur on similar systems.	An attack incident will re-occur because attackers actively probe similar systems to recreate it. The sharing of recommendations is thus critical.
Implementation	People involved in incidents are mostly trained professionals (e.g., pilots, air traffic controllers, power plant operator).	People are much more diverse with regard to their relevant skills and knowledge (e.g., children, bank employees, elderly people, medical doctor). Furthermore they can have motives and concerns unrelated to security.
	Root causes are identified and controlled.	It may be impossible to control all identified contributors that will be actively manipulated by the attacher

Data Collection & Investigation. This step remains, in its goal, as it were in safety. It is about gathering factual information about an attack, usually though digital forensics, live monitoring or in-person investigation. However, to solve challenge C_1, we need a description of an attack that is expressed in an appropriate structured format. This is missing in the socio-technical security counterpart. We define an *attack description scheme*, a structured description of fields as: *Attackers actions*, their *Effects on the systems security*, the *Declared and Imitated Identities*, the *Command* the user is asked to execute, the *Medium* used to launch the attack, and the attack's *Prerequisites*. *Prerequisites* are the capabilities that an attacker is required to possess in order to perform the attack. This choice of a structure makes also possible to generalize the attack's description, abstracting from the specific context and system. This is required if we intend to reuse the knowledge gained about this attack in the prospective analysis.

Retrospective Analysis. In this steps the analyst first builds a list of *Error Modes*, then he searches for *Contributors* for them. For this transposition from safety to security to work, we consider a human error as being 'an action or decision that results in one or more unintended negative outcomes' (quoted from (Strauch, 2004)).

An Error Mode (EM) is the analogue of a Failing Mode in technological failure analysis. An EM describes a users action that, in judgment of the analyst's observation, should have not happened, or not at that time, or not onto that object.

A Contributor is a characteristic pertaining the system's functioning that has facilitated the attacks success. For instance, 'Habits and Expectations' is a contributor to an e-mail phishing attack if the malicious message is received when a genuine message is expected (e.g., a monthly email reminding your subscriptions).

We have implemented this step by customizing an existing technique called CREAM. In so doing we

address challenges C_2 and C_3. In CREAMs original retrospective analysis, the analyst follows a cause-consequent process which is represented by tables. By traversing the tables, the analyst searches for antecedents of each Failing Mode. This process is recursive: intermediate 'generic' antecedent can be justified by other antecedents until the analyst finds antecedents that are 'sufficient in themselves. They are called 'specific' and are the most likely cause of the inspected incident. In Figure 4, left side, this is the 'Yes' branch.

To adapt this process in security, we need to define a less restrictive stop rule to yield Contributors. We have to avoid pointing invariably to the attackers actions and continue to investigate additional contributing antecedents.

Our method's stop rule has been redefined in such way that all likely specific antecedents for the event are presented to the analyst together with the specific antecedents that are contained into sibling generic antecedents. Figure 4, right side, shows this process.

The tool implements the process. It shows the analyst with a tree-like structure that the analyst can traverse opening new nodes until he finds a stop condition (see Figure 5). The analyst uses the description of the attack to define the security-critical actions carried out by the victim and the associated EMs. Additional EMs may have to be analysed in the course of events that lead to the critical action, for instance if the victim first encounters the attacker and misidentifies him/her as being trustworthy. Considering each antecedent with the attacks description in hand, the analyst follows the stop rule to build the list a Contributors of the attack under scrutiny.

Generalization. Generalisation partially addresses challenges C_4 and C_5. It is a new and sophisticated step that requires several successful steps of Data Collection & Investigations and Retrospective Analysis steps before it can produce valuable outputs. The output is a list of *Attack Modes (AMs)* compiled by grouping the output of several previous steps and organized into a catalogue. An AM is a link between an attacker's capability and the effects that it can produce on a system's security. For instance, sending a message that nudges a user to click on a malicious link (an attacker capability) is what allows the attacker to execute code on the system (effect produced on the system). Once an initial list of AMss is set (and we have bootstrapped our tools with several of them, see later), an analyst can use the catalogue to probe, prospectively, a given system for socio-technical vulnerabilities given a threat model (step (iv)).

To implement this step in our tool, we were in the need to bootstrap the tool's with an initial catalogue of AMs. Instead of waiting for a sufficient amount of real attacks to be observed, described and then analysed for their root causes, we resorted to look into fifteen *Attack Patterns* among those described in the Common Attack Pattern Enumeration and Classification (CAPEC) library (MITRE, 2014). We are here referring to the library as it were in January 2016. Then we run step (i) and (ii) on them using the tool. For reason of space we cannot give more details of our reasoning as analysts, but we resorted at our experience of computer security specialists and to common sense and knowledge in security analysis when it was necessary to take a decision.

From the CAPEC library we selected the attacks where the user is at the source of the success of the attack; after processing them, we populated the AM catalogue with 29 contributors to two socio-technical capabilities that we identified. These capabilities are intermediate goals of the attacker, peripheral and decoupled from the system on which they are exploited: *Identify spoofing* which is the capability to usurp an identity, and *Action spoofing* which is the capability to deceive the user into thinking that an action he performs will behave as he expects, whereas another action, which is harmful for the system, is executed in its place. Some of these AMs appear in Figure 5.

Prospective Analysis (Security Analysis). The last step is that of the prospective *security analysis*. It also addresses partially challenges C_4 and C_5. Assuming that a catalogue of AMs has been bootstrapped, the implementation of this step consists of querying the catalogue for the Contributors linked to a given potential threats. The tool implements this step in a semi-automated manner: the analyst describes the attacker's capabilities against which he wants to test a system's security (i.e., he specifies the *threat model*) while the tool filters and displays the corresponding AMs.

The analyst is supposed to follow a stress-test driven Security Analysis, as if he were attacking the system. To ensure that the potential attacks are indeed feasible on the system, our tool filters out the attacks that exceed the attackers capabilities that are not part of the threat model. From the tool's viewpoint, the attacks that the analyst imagines to launch on the system are no different from regular attacks and are therefore investigated in the same way (i.e., by steps Data Collection & Investigations then Retrospective Analysis). Contributors that links to the analyst's attacks are displayed along with the AMs with the difference that now the AMs now instruct the analyst about possible ways a similar attacker than him

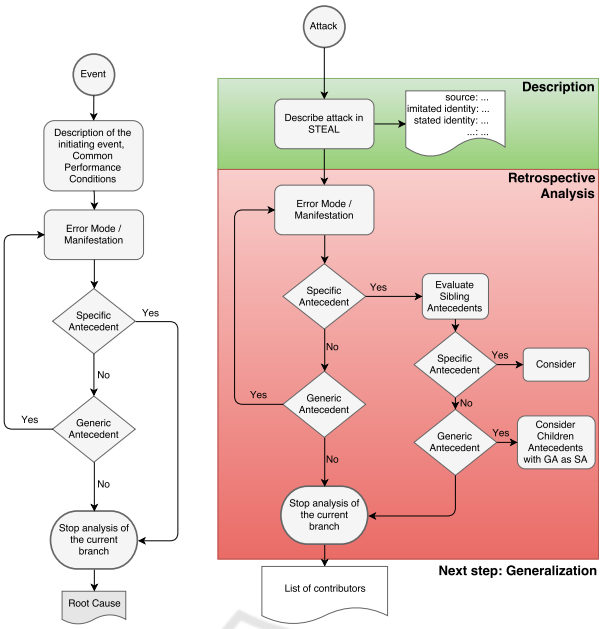

Figure 4: CREAM process (left) and its adaptation to security (right).

could intrude the system, whereas the Contributors yielded from the analyst-driven Security Analysis informs the analyst how an attacker could realize the attacks adverse effect on the system. It is then the analyst that evaluates whether the system is already sufficiently protected against the risk of exploitation of each identified Contributor. If the analyst consider that the risk is too high, he is left with the duty to provide controls that reduce the likelihood of a successful exploitation of this Contributor to an acceptable level.

5.1 The Tool: *S·CREAM Assistant*

We developed a S·CREAM's companion tool, the *S·CREAM Assistant* which is available at (Huynen, 2016). We wanted the application to be multi-platform, portable, and stand-alone in the first iterations, while still being able to transpose it to a client-server model, or even to a desktop application if we later decide so. Consequently, we chose to implement *S·CREAM Assistant* in JavaScript, storing all data in the web browser's Local Storage. *S·CREAM Assistant* uses several frameworks. AngularJS (Google, 2016) manages the Views and the Controllers, js-data (Js-data Development Team, 2016) handles the Models and provides an Object-Relation Mapping, and D3.js (Bostock et al., 2011) displays the interactive tree used to visualise S·CREAM's *Retrospective Analysis* step. *S·CREAM Assistant* allows the analyst to customize CREAM's original tables used by the S·CREAM methodogy via XLST stylesheets. The analyst can also import and export his analyses stored in the web browser's Local Storage into a JSON representation.

6 USE CASE

We put ourselves in the shoes of a security practitioners who after having assessed risks posed to one of its organization's application decides to implement OTPs to authenticate users on this application. He chooses the Yubikeys nano USB security token.

A YubiKey is a multi-purpose security token in the form of a USB dongle. A YubiKey is versatile as it can present itself as a keyboard or a two-factor authentication device to a computer (or via NFC to a smart phone). A YubiKey can be used to generate and store a 64 characters password, generate OTPs, or to play different challenge-response protocols (yubico AB, 2015). YubiKey's user interface consists of only one button and a LED.

YubiKeys have peculiar user interface and user interactions as they are not doted of a screen. The absence of a screen has for consequence to shift the duty of providing feedback to the user to a LED light. The LED's behaviors (e.g., flashing rapidly, being on or off, *et cetera*) have different meanings and are explained in the user's manual (yubico AB, 2015).

One aspect of Yubikeys is that they support two configuration slots on one device. To use these configurations, the user touches the button of the device for

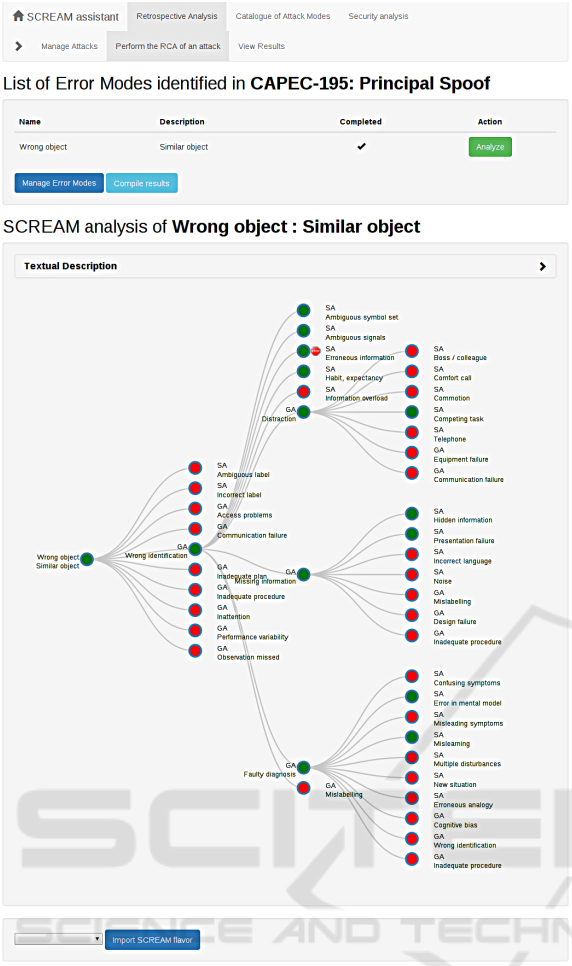

Figure 5: Screen-shot of the retrospective analysis as performed by using our tool. 'GA' are generic antecedent, 'SA' specific antecedent. Red antecedent cannot be further expanded, they denote a stop condition in the search for contributors.

different periods of time. These slots can be configured to generate OTPs or a static password. Quoting from the yubico's YubiKeys security evaluation document (yubico AB, 2012) this functionality assumes a few security implications:

> The YubiKey 2.0 introduces a mechanism where the user can use two separate credentials. We call the storage for these credentials 'slot 1' and 'slot 2'. To generate a credential from slot 1, the user touches the button for a short period of time (e.g., well below 2 seconds). To generate a credential from slot 2, the user touches the button for a long period of time (e.g., well above 3 seconds). With proper user education, we believe this does not add any additional security problems so we continue to evaluate the YubiKey configured with just the slot 1 credential.

It is worth noting that YubiKeys (now in version 4) come in two forms. One is the *standard YubiKey*, which is $18 \times 45 \times 3$mm. It has a round-shaped button with a LED at the top. The other is the *YubiKey 'nano'*. Much smaller, it completely disappears into a USB port when plugged in, and it has button and LED on its edge.

Without more details about the organization's context, we only investigate the consequences of configuration and functional choices related to the Yubikey itself. How the different configuration settings can impact the token's operations and the provided security when used by the organization's employees?

We perform our analysis on the basic operation of a YubiKey with the 'Dual configuration' functionality enabled. We set a YubiKey nano to yield an OTP on slot 1, and a static password on slot 2. The security practitioner's rationale being that it would be a waste not to propose to improve employees' passwords with a long random string of characters while the Yubikey provides this feature.

6.1 Security Analysis

Threat Model. The main assumptions for this system are that the attacker can read and write on the Internet. This Threat Model implies that the attacker is free to send messages on the web medium to the user before and after the operation of the Yubikey by touching its button. More specifically, we consider that the user is visiting a website under the control of the attacker.

Semi-automatic Security Analysis. We consider that this system's Threat Model allows the attacker to control the source, the declared identity, the imitated identity, the command, and that it can write on the web medium. As the attacker has no control over the YubiKey, he cannot spoof the action the user is about to perform. The attacker has control of the sequence of communication with the user. In consequence, by using S·CREAM, we find that the reachable Socio-Technical Capability (STC) is *Identity spoofing*.

Analyst-driven Security Analysis. To find likely potential attacks on this system, our strategy is to formulate hypotheses about the consequences of the user's actions in consideration of the attacker's extended capabilities.

There are two actions that a user has to carry out when using a YubiKey on a computer: *plugging* the YubiKey into a usb port, and *operating* the YubiKey

by touching its button according to the authentication scheme of the application. On the YubiKey nano, both actions are critical from a security point of view.

- *plugging* the Yubikey nano in a computer can accidentally produce an OTP because of the location of the button at the edge of the device. *Plugging* or *unplugging* a YubiKey nano can lead to a loss of confidentiality of the OTP code located in the first slot. As the YubiKey operates after the touching event is finished we consider that the Error Mode to investigate is 'Sequence-Wrong action', and that the user appends an irrelevant action to the sequence of actions.
- *operating* the YubiKey nano has two important dimensions: the action's duration (i.e., less than 2 seconds or more than 3 seconds) and the action's location (i.e., which user interface element has the focus at the time of the action). The user needs to touch the device within the right amount of time while being in communication with the correct entity; otherwise, there can be a loss of confidentiality. As location-based attacks are already covered by the *Identity spoofing* (i.e., the user misidentifies the attacker for another entity), we focus on the duration. In particular, we investigate the EM 'Duration-Too long'.

Table 2 sums up the results of this investigation.

Discussion on the Results and the Possible Remediations. Regarding *Identity spoofing*, the attacker has a lot of options when it comes to impersonate another entity (see the *Identity spoofing*'s column in Table 2). A prominent example of such attack is the Man In the Browser attack: the attacker, in control of the web browser, redirects the user to a website he controls when the user attempts to go to his bank's website. The attacker then asks for the credentials (including two-factors Authentication credentials as the one provided by a YubiKey) and logs into the bank's website in place of the user. The key result of this analysis is that there is little that can be done to thwart the attack, given the number of Contributors. The results of this analysis come to the same conclusion as the security evaluation made by yubico (yubico AB, 2012), which states, 'We conclude that the system does not provide good defence against a real-time man-in-the-middle or phishing attack.'

Regarding potential attacks on the 'Dual configuration' functionality, Table 2 shows that there are three Contributors that an attacker can manipulate to foster the occurrence of the 'Sequence-Wrong action' EM during the *plugging* critical action. The attacker, in control of the webpage can emit sounds or noises to apply pressure on the user, and he can also create a competing task. We see little practical application of this attack.

Finally, we turn to the case of the *operating* critical action. Investigating this *critical action* with S·CREAM yielded more Contributors than the *plugging* critical action, and therefore, it appears more likely to observe potential attacks that exploit the *operating* action as opposed to the *plugging* action. Table 2 lists the Contributors that we reckon can be used to trigger to the 'Duration-Too long' EM. For instance, we select 'SA-Confusing symptoms' because the attacker can attempt an attack in the same fashion as Social-Engineering attacks in which the attacker sends a 'bad authentication' message as sole feedback after each login attempt, nudging users to give away every password they know while trying to authenticate. The difference being that, in our use case, the user would try every possible action on the YubiKey instead of entering passwords. This kind of attack is very well possible given the fact that the YubiKey provides little feedback when a slot is yielded and no feedback about which slot is yielded. Furthermore, the user might be unsure how he configured his YubiKey (and someone may have configured it for him).

In the light of this analysis, we consider that the choice of the Yubikey for implementing OTPs is not a bad choice, but that the security practitioners should refrain from using the second slot as it poses some socio-technical security issues.

7 DISCUSSION & CONCLUSION

We developed S·CREAM with the ambitious aim to help security practitioners design effectively secure systems. The toolkit can be used at different points of the ISM cycle and allows its users to prospectively (as in the use case presented in § 6) and to retrospectively identify socio-technical factors that could lead or could have led to a security breach. Used with discernment these insights can potentially be transformed in requirements or strategies that improve a system's security, but one needs to be aware of the limitations of S·CREAM before using the toolkit.

The first limitation is that our methodology has to be further developped and properly validated. Indeed, both the tables inherited from CREAM —the RCA in safety from which S·CREAM is inspired— and the catalogue of Attack Modes require care and maintenance before the methodology can reach its full potential. Clearly, the catalogue that we bootstrapped from the CAPEC library is, for the need of S·CREAM, still rudimentary and has to be re-

Table 2: Contributors yielded by the *Security Analysis*.

STC: *Identity spoofing*	Attack: Foster 'Sequence-Wrong action' EM on **plugging**	Attack: Foster 'Duration-Too long' EM on **operating**
GA-Faulty diagnosis	GA: Sound	GA: Adverse ambient conditions
GA-Inadequate quality control	SA: Competing task	SA: Confusing symptoms
GA-Inattention	SA: Design	SA: Inadequate training
GA-Insufficient knowledge	SA: Noise	SA: Information overload
GA-Mislabelling		SA: Mislearning
GA-Missing information		SA: Multiple signals
GA-Wrong reasoning		SA: New situation
SA-Ambiguous label		SA: Noise
SA-Ambiguous signals		SA: Overlook side consequent
SA-Ambiguous symbol set		SA: Too short planning horizon
SA-Competing task		SA: Trapping error
SA-Erroneous information		
SA-Error in mental model		
SA-Habit, expectancy		
SA-Hidden information		
SA-Inadequate training		
SA-Incorrect label		
SA-Mislearning		
SA-Model error		
SA-Overlook side consequent		
SA-Presentation failure		
SA-Too short planning horizon		

fined. Furthermore, the antecedent-consequent tables we use are still not dedicated to security and therefore the Contributors yielded by S·CREAM's *Security Analysis* are currently more generic than what we think they should be. Because of it sometimes they are difficult to be interpreted, but we expect the toolkit's relevance to grow over time through its use and with the adjustments that we will make along the way. Therefore, we consider the current state of the toolkit as a proof of concept that as future work we will challenge through the analysis of different security incidents and systems, and that we will tune to improve its accuracy.

Another critical point, is that the analyst is not necessarily an expert on all aspects upon which S·CREAM can shed light. This shortcoming is even more salient when we consider the possible additions we can make to its tables. For instance, there is a lot of literature on warnings and how warnings can have a negative impact on user's decisions if not implemented properly. If this literature were to make its way into S·CREAM's tables with new possible antecedents, the analyst would be expected to be able to decide whether the warnings that are presented to the user fulfill their mission.

A final warning is that S·CREAM's results can be misused by the analyst. Indeed, it needs to be clear that the results obtained through the use of the toolkit are *potential* not *verified* causes for an attack. The toolkit produces a list of potential factors that an attacker may exploit to perform an attack on a system, but is the analyst's duty to ponder on whether these factors should be controlled on the system or not.

7.1 Future Work

We intend to validate the toolkit. By validating we mean in particular ensuring that the S·CREAM methodology yields as often as possible sound results for its security analysis.

While we have assumed the CREAM's tables also work for security still we intend to identify the Contributors that are the most often discarded by analysts, and for this we plan to challenge their relevance experimentally.

Other improvements concerns the *S·CREAM Assistant*. We intend to design helpers, such as checklists, to offer guidance to the analyst who run the tool and add the possibility to share schemes, catalogues of AMs, and sets of attacks. The tool works better if analysts of different companies cooperate and share their knowledge.

The antecedent-consequent tables inherited from CREAM should be specialized for security. We intend to provide up-to-date tables of antecedents that reflect the current state of the research on factors that influence security-related behavior

However, there is a main obstacle to overcome before reaching these milestones: we need to find a sufficient number of documented attacks to analyse.

These attacks and the corresponding attacker's traces are in the hand of security practitioners. Therefore, we need to solve very practical issues in order to make our toolset usable and used by them before starting to improve its retrospective and prospective prediction: we have to find a way to access and then to extract from raw logs the information useful to correlate socio-technical attacks with user actions and attacker activities.

We believe that the shortcomings we identified can be fixed, and that by improving S·CREAM's tables, by maintaining our toolset's knowledge of security and human-related factors, and by fostering its use and the sharing of experiences, our toolkit can be a useful addition to a security practitioner's toolbox, but we need to fully implement such ameliorations.

REFERENCES

Adams, A. and Sasse, A. (1999). Users Are Not the Enemy. *Comm. ACM*, 42:40–46.

Anderson, R. J. (2008). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley.

Beautement, A., Becker, I., Parkin, S., Krol, K., and Sasse, M. A. (2016). Productive Security: A Scalable Methodology for Analysing Employee Security Behaviours. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS) 2016. USENIX Association: Denver, CO, USA*. in press.

Bianco, D. (2014). The pyramid of pain. Available at http://detect-respond.blogspot.lu/2013/03/the-pyramid-of-pain.html.

Boring, R. L. (2012). Fifty Years of THERP and Human Reliability Analysis. *Proceedings of PSAM11*.

Bostock, M., Ogievetsky, V., and Heer, J. (2011). D3: Data-driven documents. Available at http://vis.stanford.edu/papers/d3. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*.

Boyd, J. (1995). The essence of winning and losing.

Brumfield, J. (2015). 2015 Data Breach Investigations Report. Technical report, Verizon.

Caralli, R., Stevens, J., Young, L., and Wilson, W. (2007). Introducing octave allegro: Improving the information security risk assessment process. Technical Report CMU/SEI-2007-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Cotroneo, D., Paudice, A., and Pecchia, A. (2016). Automated root cause identification of security alerts: Evaluation in a SaaS Cloud. *Future Generation Computer Systems*, 56:375 – 387.

ENISA (2016). Annual Incident Reports 2015. Technical Report October, ENISA - European Union Agency for Network and Information Security.

Ferreira, A., Huynen, J., Koenig, V., and Lenzini, G. (2015). In Cyber-Space No One Can Hear You S·CREAM - A Root Cause Analysis for Socio-Technical Security. In *STM*, volume 9331 of *Lecture Notes in Computer Science*, pages 255–264. Springer.

Google (2016). AngularJS. Available at https://angularjs.org/.

Huynen, J. (2016). S·CREAM Assistant, a tool to support S·CREAM analyses. Available at https://github.com/gallypette/SCREAM-Assistant.

International Organization for Standardization, Geneva, S. (2005). ISO/IEC 27001:2005 - Information technology – Security techniques – Information security management systems – Requirements. Technical report.

Ishikawa, K. and Ishikawa, K. (1988). *What is Total Quality Control? the Japanese Way*. Prentice Hall.

Js-data Development Team (2016). Js-data. Available at http://www.js-data.io/.

Kasikci, B., Schubert, B., Pereira, C., Pokam, G., and Candea, G. (2015). Failure sketching: A technique for automated root cause diagnosis of in-production failures. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 344–360, New York, NY, USA. ACM.

Kirlappos, I., Parkin, S., and Sasse, M. A. (2014). Learning from "shadow security:" why understanding non-compliant behaviors provides the basis for effective security. In *Proceedings 2014 Workshop on Usable Security*. Internet Society.

MITRE (2014). CAPEC - Common Attack Pattern Enumeration and Classification. Available at https://capec.mitre.org/.

Noureddine, M., Keefe, K., Sanders, W. H., and Bashir, M. (2015). Quantitative security metrics with human in the loop. In *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, HotSoS '15, pages 21:1–21:2, New York, NY, USA. ACM.

Reason, J. (1990). *Human Error*. Cambridge University Press.

Schneier, B. (2014). The future of incident response.

Schoenfisch, J., von Stülpnagel, J., Ortmann, J., Meilicke, C., and Stuckenschmidt, H. (2015). Using abduction in markov logic networks for root cause analysis. *CoRR*, abs/1511.05719.

Strauch, B. (2004). *Investigating Human Error: Incidents, Accidents, and Complex Systems*. Ashgate Pub Ltd.

Swain, A., of Nuclear Regulatory Research, U. N. R. C. O., and Guttmann, H. (1980). *Handbook of Human Reliability Analysis With Emphasis on Nuclear Power Plant Applications - Draft Report For Interim Use and Comment*. NUREG/CR. U.S. Nuclear Regulatory Commission.

yubico AB (2012). Yubikey security evaluation: Discussion of security properties and best practices. Available at https://www.yubico.com/wp-content/uploads/2012/10/Security-Evaluation-v2.0.1.pdf.

yubico AB (2015). The yubikey manual: Usage, configuration and introduction of basic concepts. Available at https://www.yubico.com/wp-content/uploads/2015/03/YubiKeyManual_v3.4.pdf.