# Discovering "Unknown Known" Security Requirements

Awais Rashid
Security Lancaster Research Centre
Lancaster University, UK
a.rashid@lancaster.ac.uk

Syed Asad Ali Naqvi
Security Lancaster Research Centre
Lancaster University, UK
s.naqvi@lancaster.ac.uk

Rajiv Ramdhany
Security Lancaster Research Centre
Lancaster University, UK
nirish777@gmail.com

Matthew Edwards
Security Lancaster Research Centre
Lancaster University, UK
m.edwards7@lancaster.ac.uk

Ruzanna Chitchyan
Dept. of Computer Science
University of Leicester, UK
rc256@leicester.ac.uk

M. Ali Babar
School of Computer Science
University of Adelaide, Australia
alibabar.m@gmail.com

## ABSTRACT

Security is one of the biggest challenges facing organisations in the modern hyper-connected world. A number of theoretical security models are available that provide best practice security guidelines and are widely utilised as a basis to identify and operationalise security requirements. Such models often capture high-level security concepts (e.g., whitelisting, secure configurations, wireless access control, data recovery, etc.), strategies for operationalising such concepts through specific security controls, and relationships between the various concepts and controls. The threat landscape, however, evolves leading to new tacit knowledge that is embedded in or across a variety of security incidents. These *unknown knowns* alter, or at least demand reconsideration of the theoretical security models underpinning security requirements. In this paper, we present an approach to discover such unknown knowns through multi-incident analysis. The approach is based on a novel combination of grounded theory and incident fault trees. We demonstrate the effectiveness of the approach through its application to identify revisions to a theoretical security model widely used in industry.

## Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications; K6.5 [**Management of Computing and Information Systems**]: Security and Protection

## Keywords

Security requirements, incident analysis, grounded theory

## 1. INTRODUCTION

Modern organisations operate as part of a complex, hyper-connected eco-system comprising other organisations and a diverse range of third-party technologies and services. Such an eco-system poses challenging security requirements. For instance, opening up the infrastructure through Internet- and web-based interfaces to support employee mobility and interactions with other organisations (e.g., through Software as a Service) results in many additional entry points that can be attacked by mal-actors. Similarly, 'bring your own device' cultures, whereby end users utilise new personal technologies or software services (from partially-trusted third parties) in their day-to-day working practice, introduce new security challenges and attack vectors. Furthermore, an organisation's IT infrastructure is often a patchwork of systems, software, services and technologies procured from third-party providers which can be harbingers of latent vulnerabilities.

Identifying and operationalising security requirements in such a complex landscape is non-trivial. Consequently, security models exist, e.g., [1], that provide best practice guidelines to identify and operationalise security requirements. Such models offer high-level security concepts that can be used as a basis for security requirements. Examples include: whitelisting of devices and software; secure configurations for hardware and software for mobile and desktop devices; continuous vulnerability assessment and remediation; data recovery planning; controlled use of administrative privileges, etc. Such models also include detailed guidelines on how to operationalise such requirements through particular security controls. Thus they represent our theoretical understanding of the current threat landscape and the resultant security requirements.

However, the threat landscape is increasingly dynamic, leading to emergence of new security requirements that demand revision, or at least reconsideration, of these theoretical models. Such emergent requirements are often implicit within or across a variety of security incidents. We refer to such requirements as *unknown knowns*[1] [16]—they represent knowledge that is "unknown" to the requirements engineer, and so does not make its way into requirements, but is "known" in that it exists in known security breaches.

Discovering such unknown knowns is challenging because incidents are usually separated from one another in terms of



*ICSE '16, May 14 - 22, 2016, Austin, TX, USA*



ISBN 978-1-4503-3900-1/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2884781.2884785

[1]Sawyer et al. adapt Donald Rumsfeld's classification of *known knowns*, *known unknowns* and *unknown unknowns* to requirements and argue that there is a fourth category of *unknown known* requirements.

both time and space. It is rare for two or more incidents to be closely co-located in time and space, investigated jointly and a common causal analysis conducted. Quantitative approaches such as statistical and data-mining analyses, most commonly used for knowledge aggregation, reveal the correlations and patterns only within the data attributes. These correlations and patterns tell us *what* is going on in the data, but not *how* or *why* these patterns appear. The analysts still have to use their domain knowledge to hypothesise about the *how* and *why*. In the absence of aggregation and internalisation of common cause-effect knowledge across incidents, these hypotheses often remain conjectural.

We present an approach to discover such unknown knowns through qualitative analysis of security incidents across space and time. Our approach is based on a novel combination of the grounded theory method [5] and incident fault trees [8] to identify new concepts and relationships that need to be integrated into existing theoretical security models – leading to updated or new models that provide a more effective basis for identifying and operationalising security requirements. The novel contributions of our work are as follows:

- We propose a novel synthesis of incident fault trees and the grounded theory method to aggregate and internalise common cause-effect knowledge (across incidents) pertaining to unknown known security requirements. While the grounded theory method enables the iterative analysis and synthesis of patterns across incidents separated by space and time, use of incident fault trees ensures that explanation is not traded off in favour of documentation—the analyst has to actively search for relevant events, their causes, enablers, consequences and mitigation measures.
- Our approach is not based on a static structure or process. Instead, it provides the means to control the depth of the analysis through explicit procedures for deriving a hierarchy of concepts across different levels of abstraction. This allows the analyst to express and explore in detail all three—*what*, *how* and *why*—levels of understanding.
- We demonstrate the effectiveness of our approach through its application to identify revisions to a theoretical security model [1] widely used in industry.

We discuss related work in Section 2. Section 3 briefly introduces grounded theory and incident fault trees. Section 4 presents our approach for discovering unknown knowns. Section 5 evaluates our approach through analysis of 11 major security incidents and identifies unknown knowns with respect to the Twenty Critical Security Controls [1] widely recommended in industry. Section 6 discusses insights from the analysis while section 7 concludes the paper.

## 2. RELATED WORK

Several researchers have focused on *identification of security requirements* before the system has been implemented using, for instance, misuse cases [18], goal and anti-goal analysis [20], abuse frames [10], and patterns of security goals [22]. Others have proposed the use of creativity techniques, such as workshops [6, 11] and goal analysis [7] to identify potential threats and incidents at the outset of system development. However, these various works have not addressed the consequences of an evolving threat landscape and related requirements change. Researchers have also highlighted the limitations of *security-modelling languages* (e.g., UMLSec, SecureTropos, KOAS and i*) with respect to relating security requirements and dynamic elements of a system [2].

Consequently, more recently, *adaptive security* has become a lively research topic. For instance, Franquera et al. [4] advocate the need for an agile security evaluation framework due to expected changes in security requirements throughout the development process. Yet, they do not detail construction of such a framework. Tsigkanos et al. [19] focus on security threats that arise due to physical proximity of potentially malicious agents and valuable assets. Such proximity requirements are known ahead. When the physical locations of threats/assets change at runtime, the pre-specified requirements on security are used to adjust the security models. Here only the physical topology changes, not the security requirements themselves.

Elahi et al. [3] propose to extend the i* modelling technique with vulnerability concepts in order to reason about vulnerability and threat propagation in requirements models. This framework is much more considerate of the change in threats and security requirements. However, it does not provide any guidelines for finding emergent threats and vulnerabilities. Instead it focuses on linking such knowledge to the requirements models. In contrast, our work provides such guidelines along with clearly identifiable strategies for revising existing models underpinning security requirements.

Salehie et al. [15] present a qualitative approach (using a causal network) to evaluate the current state of a system's security and adapt the relevant countermeasures when secured assets change. Since assets are linked into the causal network, changes to the asset model propagate to change in the security preferences inferred from the network. All models are constructed a-priori with assumptions about the intended application and its environment. Our approach complements this by providing a-posteriori evidenced information on the origin of changes to the asset model and rich content for threat model construction.

*Multi-incident analysis* tools and approaches have been developed in the context of specific industries to undertake post-hoc analysis of faults and failures. Such methods often take a quantitative approach, either based on statistical correlational techniques (e.g., [14]), or more recently, data mining approaches (e.g., [21]). Such approaches suffer from *inappropriate abstraction*, i.e., each incident is described in terms of a finite number of specified attributes, which have to be reasonably generic so as to accommodate a large number of incidents. The distinct details of each incident are thus abstracted away. Such abstraction also does not allow for the representation of semantic differences owing to the changing definitions of attributes across time and space and also the different purposes of the data and collection methods used. Consequently, the correlations and patterns identified can only reveal what is going on in the data, but not how or why these patterns appear.

A number of general purpose *single incident analysis* approaches can also facilitate multi-incident analysis. Approaches such as MORT [9] and HPIP [12] often express the accumulated insights about hazards in a particular field by means of a limited number of concepts that can be used to describe an incident. These concepts may include taxonomies of incident types, their potential causes, and pre-

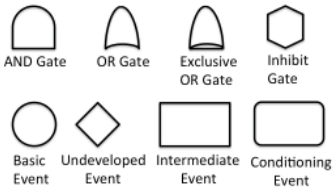


**Figure 1: Symbols used in Incident Fault Trees**

scribed questions to guide the incident analysis. This enables the results from different incident analyses to be integrated by means of a common incident vocabulary. However, this integrative ability generally comes at the cost of accuracy, as these techniques struggle to accurately represent the peculiar and idiosyncratic characteristics of each incident. Similar to quantitative multi-incident analysis approaches, these techniques also cannot mitigate the problem of semantic differences between various incident attributes across time and space. Thus, there is a need for approaches, such as ours, that capture and integrate new cause-effect knowledge into an evolving security model while preserving the contextual richness of incidents.

# 3. BACKGROUND CONCEPTS

## 3.1 Grounded Theory Method

The Grounded Theory Method (GTM) [5] is an approach for deriving theories based entirely on data. It works by breaking the data down into small portions, such as individual lines, and then assigning labels, called *codes*, to each portion. Each code should represent the concept expressed in its corresponding portion of data. The codes are then compared to one another and, based on similarities and differences, grouped under more abstract concepts called *categories*. This constant comparison between and across codes and categories continues, even as new codes are integrated into the analysis and new categories discovered. The derived categories, their relationships and dependencies, and reflections on the analysis are recorded in *memos*. Through this iterative abductive approach, where data analysis and collection are both simultaneous and interdependent, a theory emerges which is fully grounded in the available data. GTM is used in our approach to aggregate the results of the various individual security incident analyses, augmented through incident fault trees (IFTs), to produce an abductive theory that captures unknown known security requirements.

## 3.2 Incident Fault Trees

An Incident Fault Tree (IFT) [8] is a modelling tool for the retrospective documentation and analysis of incidents and accidents. It is used to explain an incident in terms of the various causes that contributed to it. IFT is based on and, thus, uses the graphical symbols (see Fig. 1) of the Fault Tree Analysis (FTA) method. An IFT comprises of two broad types of notations: events and gates. Events may describe specific discrete events or other conditions that are relevant in terms of causing an incident. Gates are symbols that represent various ways in which the events need to combine in order to cause another event. We next describe those symbols that are used for IFT modelling in our approach:

- *Basic Event:* cannot be explained in terms of other causal events, or has no known causes.
- *Intermediate Event:* can be explained in terms of other causal events, or there are known causes for it.
- *Undeveloped Event:* can be explained in terms of other causal events but we have chosen not to do so because such causal elaboration serves no analytical purpose.
- *AND Gate* and *OR Gate:* respectively represent the conjunction or disjunction of causal events.
- *Inhibit Gate:* is placed between an event and its causes to stop an event from occurring unless the condition in the *Conditioning Event* is satisfied. In practice, this combination is used in our approach to model mitigating or remedial actions that may act as barriers to the progress towards a security breach.

IFTs support our GTM-based approach by providing a framework for explicitly structuring the codes/categories and their causal relationships. Such a framework is well suited to GTM-based qualitative analysis as it is descriptive and scalable—it can grow in detail as the investigation proceeds and more information comes to light. Further, the process of deriving a basic IFT (that describes an incident) may be augmented through the change analysis and barrier analysis methods [8]. These methods encourage the analyst to think about the specific departures from normal operating procedures and the failed barriers that led to the incident. This leads to consideration of not only the errors of commission, i.e., factors that contributed to an incident, but also the errors of omission, i.e., what was ***not*** done that led to the security breach. IFTs facilitate flexibility in analysis because the analyst can explore different hypotheses by elaborating and developing different paths in the tree to investigate various causal dependencies. Furthermore, conventional GTM can be effort intensive as all information is indiscriminately coded and categorised. In contrast, in our approach, this coding is facilitated and directed by IFTs.

# 4. APPROACH

Our method for discovering unknown known security requirements is qualitative – it is a synthesis of GTM with IFTs. It is both an analysis and an investigation method. The method evolved from our desire to find an effective way to deal with the implicitness of domain knowledge when conducting a multi-incident analysis of security breaches[2]. In order to manage the large amount of unstructured and semi-structured data, we started organising it using IFTs. Eventually the IFT became the pivotal model in the method, which honed and guided both the analysis and investigation aspects of the method. Just like conventional grounded theorists use interviews to extract the relevant pieces of information for their analysis from the life story of the subject/interviewee, our method uses the IFT model to extract the relevant pieces of information for the purpose of causal analysis from the documentation of an incident. The nature of IFT means that the analyst is actively prompted to search for the breach events, their enabling contexts, resulting consequences and preventative measures. Fig. 2 presents an overview of our method. The ovals represent processes and

[2]We use the term incident and breach interchangably to maintain consistency with the terminology of IFTs.

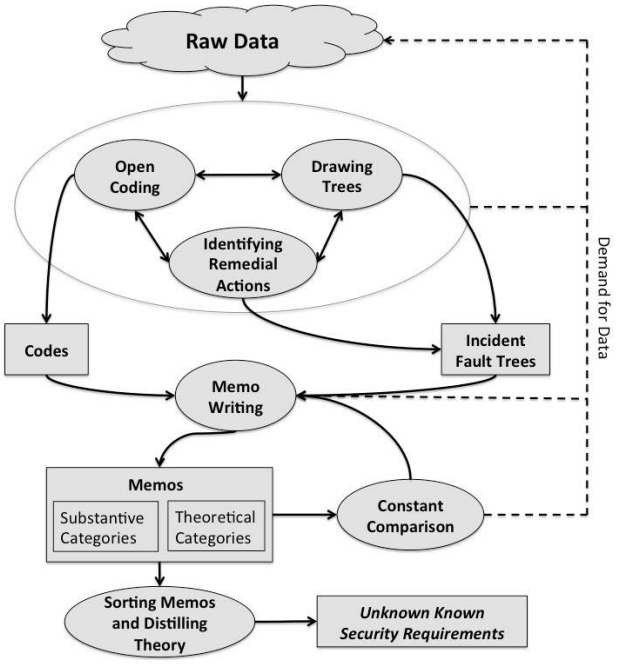


**Figure 2: Overview of the method**

the rectangles represent the artefacts or products of these processes. The method is aimed at analysts (e.g., in organisations such as CERTs) interested in distilling new classes of security problems through analysis of otherwise isolated security incidents. We next discuss the various steps of the method. Note that, for retrospective analysis, as is the case with our method, raw data mostly consists of second hand data in the form of breach incident reports, studies, analyses and other archived information.

***Drawing Trees:*** The analysis usually begins by first reading the incident investigation document(s) to identify the interesting events or conditions that we would like to analyse and explain. After one or more of these events or conditions have been identified, each of them may become root nodes for IFTs. Next, we search for and identify concepts and factors, from the documentation, that caused the root incident. This process continues recursively, with further searches, to identify the causes of each of the previously identified causal factors. This search may be informed by *barrier analysis* (i.e., actions and countermeasures that were either unused or inadequate in preventing the incident) and *change analysis* (i.e., deviations from normal operating procedures or abnormal working practices that contributed to the incident). Fig. 3 represents the fault tree as produced by our analysis for one of the incidents we analysed.

***Open Coding:*** We can see in Fig. 2 that the *Drawing Trees* process is part of a larger process that also contains *Open Coding* and *Identifying Remedial Actions*. This larger process signifies that the three activities are interdependent and essentially happen in parallel. Open coding is the process of annotating small portions of text with codes that indicate important concepts from these texts. When we identify an interesting event from the documentation that we would like to analyse, we also select the text related to that event and open code that text. The choice of the relevant events, texts, and the coding itself can be carried out by two or more independent researchers to counter potential bias. The codes identified from the coding process may in turn inform the tree building process by indicating the causes for the event under consideration. This process can become cyclical where building the IFT may lead to selecting the material for open coding, and open coding may lead to building and modifying the IFT.

The data selected for open coding, though centred on the IFT, may contain some additional details as well. Thus the codes created during open coding may not be strictly restricted to the event that we were trying to explain. These additional codes may be valuable in that they may produce insights that would result in modifying or adding data to the IFT in ways that we had not anticipated before. They also provide information about the context in which incidents take place. Consequently, these peripheral codes may lead to the discovery of important concepts and new directions for investigation during the constant comparison process because they may also be present in other incidents.

***Identifying Remedial Actions:*** The structure of the IFTs provides us with the opportunity to identify the areas where remedial actions might be taken. The areas are identified in the form of the paths going from the leaf nodes to the root of the tree. The remedial actions are essentially those activities that try to disrupt the progression of causes from the leaves to the root. A remedial action on a causal path is represented by a combination of an inhibit gate and a conditioning event. It describes the actions that can mitigate the effects of the events described along the causal path below the inhibit gate. These solutions may come from recommendations in the incident report, subsequent follow up reports, or from news sources after the incident describing how the state has changed and improved since the incident. The process of deriving remedies may also help to refine IFTs. While thinking about barriers and recommendations, associations may appear between different causal paths allowing us to aggregate them under a common parent node or category. During constant comparison, the remedial measures from different failures are compared and can result in identification of unknown known security requirements.

***Memo Writing:*** Generalities in the data, in the form of concepts and relationships, are discovered and recorded during memo writing. The processes of Drawing Trees, Open Coding, and Identifying Remedial Actions will produce an IFT and a set of codes related to that tree. During memo writing these codes are compared with one another and, based on similarities and differences, aggregated into more abstract concepts called categories. Further comparisons between and across codes and categories may lead to the discovery of more abstract categories. These categories and their relationships are recorded in memos. All hypotheses and suppositions are also recorded in memos. Memos can be written at any stage of the analysis. The aggregation and extrapolation of concepts during memo writing, informed by the causal relationships in the IFTs, may point to gaps in existing analysis and thus provide direction to the emerging theory. For example, a memo may reveal a new possible causal factor about an incident thus prompting a search for evidence to substantiate this hypothesis. If this evidence is found then the revealed causal factor may be added to the IFT for the incident, thus enriching it. New insights may also be discovered because writing memos can expli-

cate tacit ideas. Figs. 4, 5, 6 show excerpts from various memos, summarising the derivation of several categories in terms of finer grained concepts. During the constant comparison process where faults[3] are compared to other faults, their memos are also compared to one another in order to discover unknown known security requirements.

***Constant Comparison:*** In this phase the insights obtained from an incident analysis are integrated into the evolving theory by constantly comparing its artefacts (IFT and memos describing categories and their relationships) with similar artefacts from previously analysed incidents, as well as the concepts of the evolving theory. This comparison integrates knowledge across various incidents, where both causes and remedies identified for one incident may inform and instigate further analysis of other incidents in order to discover unknown known security requirements.

***Sorting Memos and Distilling Theory:*** The sorting and writing process mainly consists of selecting and arranging the relevant information from the memos and comparing it against the existing theoretical security model in order to distill a theory representing unknown known security requirements. This theory, as finally presented, comprises a number of security concepts and their associations. The remedial measures are also classified into categories and related to causal factors with the aim of providing guidance for operationalising the security requirements.

# 5. EVALUATION

In order to evaluate the effectiveness of our approach, we apply it to discover unknown known security requirements with respect to a real-world security model: the Council on Cyber Security top 20 Critical Security Controls (CSC20) [1]. We focus on a particular class of security threats: *data exfiltration* (i.e. data theft). This is because, as systems become increasingly open, security models need to evolve beyond traditional 'castle defences' of firewalls and intrusion detection systems. Furthermore, the increasing frequency of data breaches – despite awareness of and compliance with models such as CSC20 – alludes to the possibility of unknown known security requirements with respect to such models.

By applying our approach to a corpus of 11 security incidents that exhibit a diversity of attack traits, we show that it is possible to keep the corpus of incidents relatively small and still saturate data exfiltration attack concepts and countermeasures. We then present a comparative analysis of our discovered security requirements with CSC20. The analysis was conducted with reference to Version 5.0 of CSC20, the latest version at the time of the analysis. This analysis reveals disparities—unknown known security requirements—in the coverage of data exfiltration threats in the theoretical security model represented by CSC20. We note that the unknown known security requirements discovered through our analysis were passed on to the Council for Cyber Security for consideration in revisions to CSC20. Version 6.0, subsequently released, reflects a number of our recommendations. Though the Council for Cyber Security does not provide direct traceability between any recommendations submitted and revisions released, this nevertheless demonstrates the utility of our approach.

[3]We use the term 'faults' to refer to vulnerabilities or actions leading to security violations in order to maintain consistency with the terminology of IFTs.

We next illustrate our analysis in detail, initially describing the analysis process for the first incident in our corpus in terms of its IFT and GTM analysis artefacts (Sec. 5.1). We then describe the various intermediate stages in our analysis process and show how the various security concepts and requirements were discovered (Sec. 5.2). Finally, we summarise the unknown known security requirements with respect to CSC20 (Sec. 5.3). The full analysis and the comprehensive IFTs are available at: http://dx.doi.org/10.17635/lancaster/researchdata/60.

## 5.1 Analysis of Nitro Attacks

The Nitro attacks were a targeted campaign (Jul-Sept. 2011) against 29 Fortune 500 companies in the chemical sector and another 19 in various other sectors, primarily the defence sector. The attackers sought to steal intellectual property such as design documents, chemical formulae and manufacturing processes for chemicals and advanced materials. We model the chain of events leading to the final outcome using an IFT. For the sake of brevity, a simplified IFT is shown in Fig. 3.

The tree offers insight into the way the exfiltration attack panned out and where security controls can be deployed to detect, prevent or mitigate the attack. As illustrated (Event 1.1), the rootkit program (i.e., the malware), authenticated with a Command and Control (C2) server via TCP port 80, and upon success, received binary code containing Poison Ivy, a common Remote Access Trojan (RAT). Using the C2 server, the attackers then instructed the RAT instance (Event 1.3) to provide the infected computer's IP address, the names of all other computers in the workgroup or domain, and dumps of Windows cached password hashes. Using various tactics such as pass-the-hash attacks or cracked Windows password hashes, the attackers proceeded to gain access to other computers with the same Network Logon user rights (Event 2.1). As attackers needed more elevated access rights, they performed privilege escalation (Event 2.2) on non-administrative users and then moved on to gain access to key high value targets, which included process experts and server administrators. This enabled them to traverse the network with ease and find servers hosting the desired intellectual property and gain access to the sensitive materials. Once the attackers identified the desired intellectual property, they established access to staging servers at key aggregation points on the network; this was done to prepare the data for exfiltration. The data was copied to these internal staging servers where it was aggregated, compressed and encrypted for exfiltration (events 4.1 and 4.2). Depending on the volume of data to be exfiltrated and the degree of stealth required, the attackers used different exfiltration channels, such as FTP, HTTP and email – all of which are high bandwidth channels available to users on most networks.

Open coding and the constant comparison of codes enables the requirements analyst to organise the above understanding of the causes of the incident and produces insights that result in refining the IFT. Codes are grouped into hierarchies leading to the formation of categories. The comparison of codes and categories for the Nitro attacks produced the following main categories: Target Identification, Reconnaissance, Attack Staging, Network Intrusion, Concealment & Persistence, Data Exfiltration, Attack Consequences, Countermeasures. From this refined understanding, the IFT was modified to include these super nodes. Us-

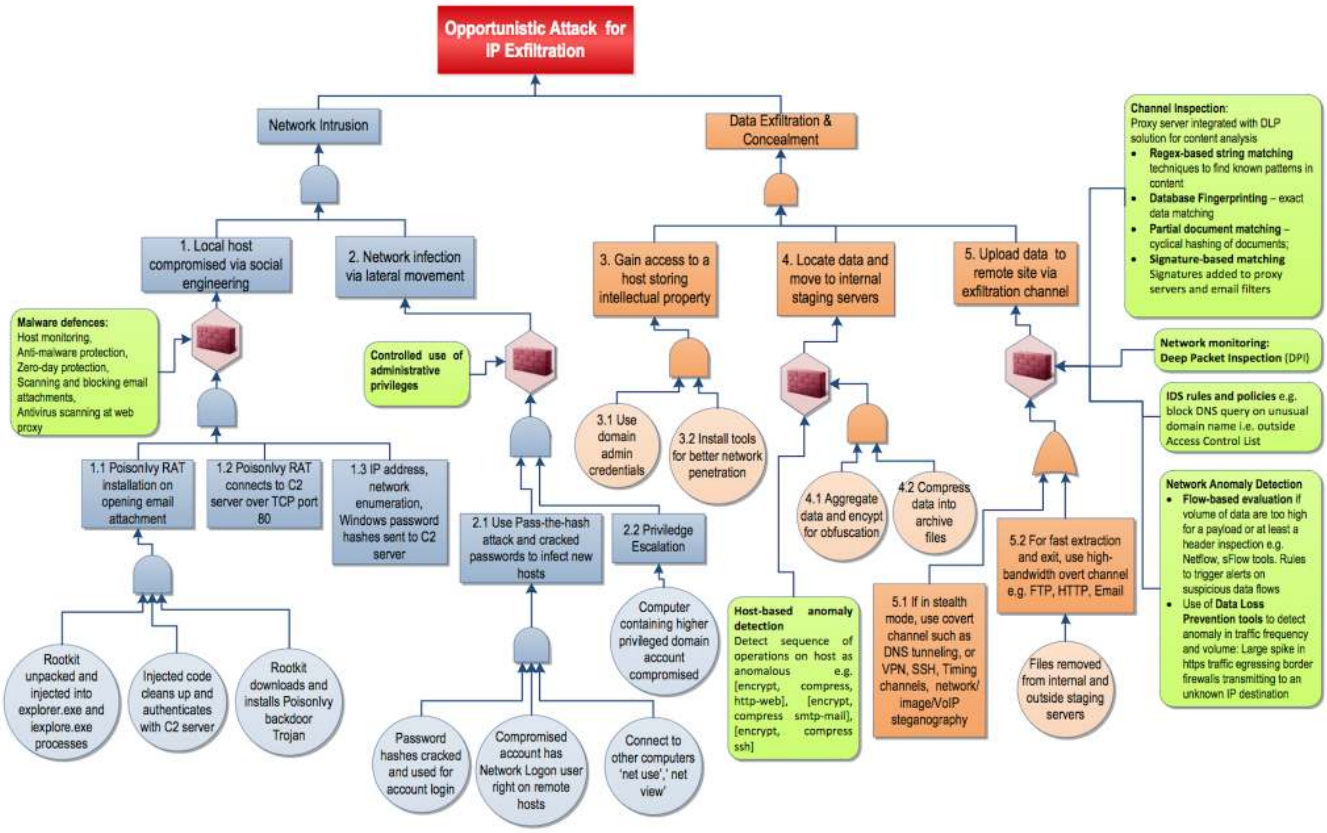


**Figure 3: Simplified IFT for Nitro attacks**

ing the IFT, the data exfiltration concept was recursively developed into concepts and factors that may have contributed to it. Our refinement of the data exfiltration concept led to the selection of new material about data exfiltration schemes utilised within the incident. This new material was open-coded and the codes constantly compared to introduce new concepts (sub-categories) recorded in memos. As seen in Fig. 4, the Data Exfiltration concept is broken further into the concepts: 1) Data Capture, 2) Data Staging & Preparation, 3) Exfiltration Channels, and 4) C2 Infrastructure/Communications. Further iterations of this analytical process enabled, for example, the different modalities of the 'Data Capture' concept to be defined, and the identification of overt/covert types of exfiltration channels. Whenever possible, insights gained from the memos were used to modify the tree to introduce new causal factors (e.g., events 5.1 and 5.2). Additional insights about countermeasures were obtained through barrier analysis. Fig. 5 shows a memo excerpt, summarising such remedial actions uncovered.

Our memos not only recorded the concepts representing various stages and remedies of attacks, produced through the comparisons of codes and categories. They also identified and summarised the classes of exploits used during the attack, the conditions within the organisation that made the use of these exploits favourable and the types of channels used to exfiltrate data in the incident. They were also used to hypothesise on various other possible data exfiltration channels that could have been used in the attack. Since our IFT analysis indicated a combination of network intrusion and exfiltration causes that led to the theft of intellectual property, the memos reflected both the intrusion defences to defeat the exploits used in the attack as well as post-intrusion measures to prevent data exfiltration.

## 5.2 Iteratively Building the Theory

Our method for discovering unknown known security requirements involves refining our theory by iteratively applying the analysis approach to several other incidents. This involves integrating the insights obtained from the current analysis with those from previous analyses by constantly comparing their IFTs as well as categories recorded in memos. As the new categories identified in the current analysis are compared with the extant set of categories, existing categories are potentially modified or aggregated or new categories created to augment existing ones. Such new categories indicate new patterns of attacks, new classes of data exfiltration channels and countermeasures identified from the barrier analysis within the IFT.

We used insights obtained from our first analysis to select the next incidents for analysis. In particular, after the first incident analysis we became cognisant of the different modalities in which data assets are prevalent in organisations, namely (see Fig. 6(a)): *data at rest* (i.e., 'inactive' data stored in databases, file servers, archives, etc. that is not accessed or changed frequently), *data in use* (i.e., 'active' data processed by applications and held in computer memory, CPU caches/registers, or operational tables in databases) and *data in motion* (i.e., data in transit in

| *Categories* | *Description* |
|---|---|
| **Data Capture** | • Data in motion<br>• Data at rest<br>– Intellectual property such as design documents, chemical formulae and manufacturing processes for chemicals and advanced materials data residing in file systems, databases and other storage – copied to staging servers by attackers<br>• Data in use |
| **Data Staging and Preparation** | • Establish access from key high-value target machines to staging servers at key aggregation points on the network<br>• Copy data to staging servers<br>• Prepare data for exfiltration<br>– Perform in-situ data analysis and extraction before exfiltration |
| **Exfiltration Channels** | • High bandwidth overt channels<br>– FTP or HTTP<br>– Design documents copied to remote servers via HTTP<br>• Covert channels<br>– Using SSH or DNS tunneling channels for transmitting data to external staging servers is less likely to be detected<br>• High bandwidth covert channels<br>– Dropbox, Box file servers<br>• Exfiltration channel level of covertness and bandwidth choices<br>– Exfiltration channel used depends on the level of stealth used in the attack, the volume of data to be exfiltrated and the likelihood of detection by security systems |
| **C2 Infrastructure and Communication** | • RAT malware connection modes<br>– Reverse TCP mode RAT malware<br>– Socket reuse mode RAT malware<br>• C2 server connection via unblocked ports<br>• C2 communication protocols are encrypted<br>• Covert C2 communication<br>– Box/Dropbox implants: command and control via filename changes. |

**Figure 4: Summary of the subcategories of the data exfiltration category from the Nitro attacks**

| *Categories* | *Description* |
|---|---|
| **Data at rest discovery scanning** | • Identify the location of sensitive data in the organisation<br>• Perform content inspection on documents to look for sensitive content<br>• Generate (e.g., regular expressions) rules for content matching of files containing sensitive structured content<br>• Generate data fingerprints, hashes or file signatures that can be used by filters to detect exfiltration of this type of data |
| **Channel inspection and network monitoring** | • Content matching at proxy servers and filters<br>• Signature-based matching in DLP (data loss prevention) modules at proxy servers for common overt channels<br>– Add signatures at proxy servers to block documents from leaving the organisation<br>– Content analysers to look for matches in keywords, personally identifiable information, hashes, defined patterns or files flagged by signatures<br>• Subject all out-going traffic to deep packet inspection (DPI) |
| **Host-based data encryption and anomaly detection** | • Approved hard-drive encryption software deployed on hosts<br>• Host-based monitoring via host-based intrusion detection systems to recognise malware or backdoor activity<br>• Utilise file integrity checking tools to ensure that critical system files (including sensitive system and application executables, libraries and configurations) have not been altered |
| **Network anomaly detection** | Perform continuous monitoring on all inbound and outbound traffic Deploy Netflow, sFlow or IPFIX tools to determine baseline 'normal' netflows between servers and workstations and detect anomalous flows |

**Figure 5: Summary of the countermeasures categories derived from the Nitro attacks**

the form of packets transported by communication protocols and data traversing the network, temporarily residing in memory buffers to be forwarded). We used this insight to guide our incident selection process to: (i) discover concepts across the breadth of incident classes, and (ii) within each

| **Intrusion Detection & Prevention** | **Data at Rest Exfiltration Countermeasures** |
|---|---|
| 1. Countermeasures for SQL injection<br>2. Countermeasures for buffer overflow exploits<br>3. Hardened OS configurations with<br>a. Address space layout randomization (ASLR)<br>b. Stack Canaries<br>c. Executable Space Protection<br>4. Malware Defences<br>5. Continuous vulnerability assessment of deployed software on company servers<br>6. A policy for subscribing to CVS alerts and applying security updates.<br>7. Keep up with the latest bug reports for web and application server products and other products in company's Internet infrastructure.<br>8. Apply the latest patches to these products.<br>9. Periodically scan company web site with one or more of the commonly available scanners that look for buffer overflow flaws in server products and custom web applications.<br>10. Code review for custom applications that accepts input from users via the HTTP request to prevent buffer overflow exploits | 1. Data in motion, data in use discovery scanning<br>2. Countermeasures for memory-scraping for track2 data<br>a. Prevent core dumps on POS/ payment processing computers<br>b. End-to-end encryption of payment card data when it must be shared among payment network participants (data in transit) and when it must be stored in proprietary systems (data at rest)<br>c. Chip technology<br>3. Content matching at proxy servers and filters<br>4. Channel Inspection for repeated byte count patterns<br>5. Network Monitoring to detect flows pertaining to exfiltration channels or C2 behaviour<br>6. POS protection rules,e,g.<br>a. Adopt a 'no Web surfing on the POS computer' policy |

**Figure 7: Sample countermeasures identified through the iterations**

class, to attempt to saturate concepts by considering further variants of the exfiltration attacks of the same class.

To select data exfiltration incidents that exhibited the 'Data in Motion' and 'Data in Use' modalities, we considered the range of attacks that targeted the consumer-payment chain, which includes actors such as merchants and their payment-processing partners. For data-in-motion-class exfiltration incidents, we selected attack campaigns that used various point-of-sale (POS) memory-scraping malware such as the Dexter malware-based breaches of POS systems in retail, hospitality and fast food outlets and the Target data breach incident. Amongst the incidents we considered, the Heartland Payment Systems data breach embodied exfiltration strategies where attackers targeted both data in motion (card data at the merchants' POS terminals) and data in use (card data within payment processing systems).

As in our previous iteration, incident trees were created to facilitate the analysis of causal factors; barrier analysis used to determine the effectiveness of mitigation measures; open-coding and constant comparison used to discover exfiltration concepts and refine our collection of categories. The addition of new categories to the 'Data Capture' subcategory (see Fig. 6(a)) shows our improved understanding of the different data capture modalities that can be exercised in a data breach incident. New exfiltration concepts uncovered in our analysis led to a reorganisation of channels into overt, overt encrypted, covert, and covert encrypted sub-categories (Fig. 6(b)). In addition, by probing further into the identified exfiltration channels, our understanding about the inherent characteristics of exfiltration channels such as bandwidth and covertness was reinforced. By comparing the exfiltration channel concepts across the incidents already analysed, we also started to develop insight into the recurrence of such channels. Finally, memos were written to elaborate on possible other preventive/remedial techniques. Fig. 7 presents an excerpt from a memo summarising some of the countermeasures identified through this process.

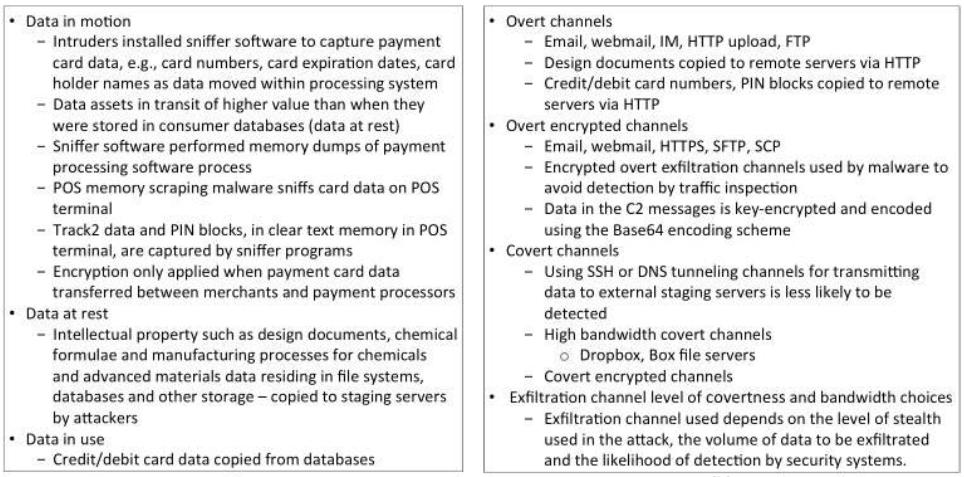


**Figure 6: (a) Sample of refined data capture categories during iterations (b) Sample of data exfiltration channel categories during iterations**

## 5.3 Identified Unknown Knowns

The analysis of data exfiltration incidents and re-selection of new attack incidents was continued till no new incident considered exhibited major variances to properties identified in previous iterations. This saturation was clear during the analysis of the 9th incident, with the 10th and 11th incidents producing categories that were minor variances on existing categories. The body of memos reified the tacit knowledge embedded within every incident about the causes of the incident, classes of exploits used during the attack, conditions within the organisation that made the use of those exploits favourable, the types of data channel used, and other possible data exfiltration methods. Crucially, the summative memos also contained, in the form of specific measures to detect and mitigate data exfiltration attacks, a crude representation of our identified security unknown knowns. To develop a theory encapsulating the data exfiltration security requirements, the memos were sorted and reorganised in terms of data exfiltration method types, and for each method, in terms of measures for detection and mitigation, and associations to organisational conditions, practices and relative unpreparedness, that increased susceptibility to breaches. In doing so, we formulated our theory to first include a taxonomy of data exfiltration methods grounded in the security concepts empirically collected from the data breach incidents. The countermeasures were then associated with these classes and subclasses of data exfiltration methods. This enabled particular groups of security requirements to be readily selected when relevant traits were identified in a data exfiltration incident under analysis.

In order to facilitate the operationalisation of the security requirements, the security model (based on our theory) was formulated to include controls in a fashion similar to CSC20. These controls represent a prioritised and empirically-grounded set of security actions that organisations can take to assess and improve their current security state i.e. their resilience to data exfiltration. Our model refinement produced 12 Data Exfiltration Detection and Mitigation Controls (listed in Fig. 8) and 51 sub-controls (refinement of the security model represented by the 12 controls). A small subset of the security requirements associated with traffic restriction, known channel inspection and channel traffic filtering concepts is reproduced in Fig. 9. A further refinement described importance of the control in blocking or identifying presence of attacks and classified security requirements on the same lines as used in CSC20: i) *Quick-wins*, ii) *Visibility and attribution measures* (to improve the security capabilities within an organisation), iii) *Security configuration and hygiene* (i.e., best practice guidelines) and iv) *Advanced sub-controls* (i.e., detailed guidelines representing latest breakthroughs in countermeasures). Such an alignment enabled comparison of our derived security model with CSC20 in order to identify unknown known security requirements.

| Data Exfiltration Detection and Mitigation Controls |
|---|
| 1. Data at rest, in motion and in use discovery and protection |
| 2. Secure host configuration, host access control, protection and monitoring |
| 3. Traffic restriction, known channel inspection, channel traffic filtering |
| 4. Steganographic channels: protocol tunnelling detection |
| 5. Network monitoring and anomaly detection |
| 6. Boundary defences |
| 7. Logging, log review and auditing |
| 8. Malware defences |
| 9. Protecting credentials with administrative privileges |
| 10. Security provisions for Virtualisation, Virtual Desktop Infrastructures and Cloud Computing |
| 11. Self-protecting data |
| 12. Post hoc data leakage analysis |

**Figure 8: Exfiltration detection and mitigation controls**

To evaluate the pertinence of our security model, the security requirements were compared and contrasted with CSC20. Each requirement was assessed as to whether it was fully or partially covered by CSC20 or if it were a new requirement. As shown in Fig. 9, traffic restriction through perimeter firewalls, proxy servers, and protocol whitelisting is fully covered by the following controls within CSC20: CSC 13-6, CSC 11-1. Requirements 3.3.2, 3.3.5, 3.3.6 and 3.3.7 are, on the other hand, partially or not supported at all by CSC20.

Fig. 10 shows the various CSC20 for which revisions were identified through our analysis. i.e., they provided only partial coverage of the security requirements identified through

| | Control Type | Recommended Security Control/ Exfiltration countermeasure | Relevant CSC | Coverage |
|---|---|---|---|---|
| 3.3.1 | Traffic restriction (proxies/protocol whitelisting) | Configure perimeter and internal firewall to ensure that all outbound traffic go through a proxy.<br>Restrict traffic to protocols that are business critical. | CSC 13-5, CSC 11-1 | Full |
| 3.3.2 | Known Channel inspection | Deploy high-risk channel proxy (e.g. Squid web proxy, email filters) with DLP modules for content analysis, file-hash, regex strings, signature matching, e.g. for HTTP, SMTP, IM, and FTP channels | CSC-17-5 | Partial |
| 3.3.5 | Known Channel inspection | Use steganalysis modules for proxy filters to detect steganographic channels | CSC-17-5 | None |
| 3.3.6 | Known Channel inspection | Use a reverse SSL proxy to analyse contents of SSL/TLS transactions. This raises issues about privacy but a whitelist of web sites using HTTPS can be created for inspection and employees informed about the whitelist. | CSC-17-5 | None |
| 3.3.7 | Actuated Detection Systems | Configure rules on IDS to detect DNS queries on unusual domain names (e.g., outside the Access Control List) or known malicious C2 domains.<br>The computer sending the anomalous DNS request (detected by above rule) should then be automatically quarantined to a VLAN | CSC 5-11 | Partial |

**Figure 9: Traffic restriction, channel inspection and traffic filtering requirements**

our analysis. The figures in paranthesis indicate the number of revisions identified. The revisions are summarised below:

- *Deployment of automated tools to monitor network perimeters (CSC 17-5)*: CSC 17-5 is overly simplistic as it does not define the notion of typical exfiltration channels. Nor does it describe the provision of content analysers for specific channels. In contrast, a Known Channel Inspection sub-control (from our security model; Req. 3.3.2 in Fig. 9) identifies the need for proxy servers in high-risk overt exfiltration channels to inspect the content of messages for sensitive information.

- *DNS query logging mechanisms (CSC 5-11)*: CSC 5-11, in its current form, focuses only on logging and log review as a detection mechanism. Our partially-matched Actuated Detection System sub-control (Req. 3.3.7, Fig. 9), in contrast, proposes more advanced measures, e.g., detecting anomalous DNS queries (outside an access control list) through rules specified in the perimeter Intrusion Detection System (IDS) and automatic quarantining of the query sender to a VLAN.

- *Use of data loss prevention (DLP) systems (CSC 17-9)*: CSC 17-9 currently recommends using network-based DLP tools to monitor traffic patterns without any precision about the type of patterns to detect C2 activity. Our revision proposes new requirements such as detecting disparities between the volume of inbound and outbound packets. We also suggest measures to reduce false positives (legitimate traffic also falls into the data-in-out-disparity category), e.g., analysing historical network flows to create a baseline.

- *Cloud provider security practices for data protection (CSC 17-4)*: CSC 17-4 is vague and simply states that cloud provider security practices should be considered. Our revision to CSC 17-4 identifies requirements to review provisions for the isolation of virtual machines and mechanisms to restrict interactions between co-hosted virtual machines.

Fig. 10 also shows that nine new requirements were identified: 8 for CSC 17 (which focuses on data protection) and 1 for CSC 3 (which pertains to secure configurations for hardware and software). Examples include the need for:

- countermeasures against use of steganographic techniques and protocol tunnels for covert exfiltration;

| | CCS Critical Security Control |
|---|---|
| **No. of revisions proposed** | CSC 17-5 (1), CSC 5-11 (1), CSC 17-9 (1), CSC 17-4 (1) |
| **No. of new requirements** | CSC 17 (8), CSC 3 (1) |
| **Total Amendments** | 13 |

**Figure 10: Proposed amendments to the CCS Critical Security Controls v5.0**

- monitoring connection patterns between endpoints for anomalous behaviour (e.g., suspect endpoint access, or hosts acting as staging servers during attack);

- preventing memory-scraping operation of malware by adopting solutions for rapid erasure of data from memory or timely encryption of sensitive data (e.g., adding diversity or randomisation to data address spaces);

- data-at-rest protection via automatic encryption in file systems/databases and through negative databases [13].

# 6. DISCUSSION

## 6.1 Towards more effective security models

Security models such as CSC20 embody a theoretical understanding of the threat landscape and the resultant security requirements at a particular instant in time. They provide key domain knowledge about the *known known* security threats and associated requirements, and can reasonably extrapolate to the *known unknowns* based on this knowledge. The constantly mutating nature of security threats, however, poses a problem to the pertinence of such security models. While *unknown unknowns*, such as zero days, will always pose a key security threat, unknown knowns result in the security models being superseded by the relative mutation pace of existing threats and the sophistication of new threats. This presents a key challenge as what is unknown known from the perspective of the requirements engineer may not be so from the perspective of the attacker. Furthermore, while the attacker only needs to identify one unknown known, the requirements engineer must seek to understand all possible unknown knowns or, at least, as many as possible. While we do not claim completeness of the unknown knowns identified through our method – such completeness is impossible given the diversity of security challenges in contemporary settings – our analysis shows that we can saturate the identification of unknown knowns in a particular class of threats (in our case data exfiltration) with a manageble number of incidents (11). Our analysis identified 4 partial and 9 fully unknown known security requirements in a widely adopted security model. This demonstrates both the effectiveness of our approach and the scale of the challenge in order to keep security models in step with tacit knowledge embedded in incidents across space and time. Our approach provides a means for security models to evolve as this tacit knowledge does – leading to richer models that aggregate and internalise cause-effect patterns across incidents.

## 6.2 Partial vs. fully unknown known

Our experience highlights a further categorisation of unknown known security requirements. Partially unknown knowns capture mutations of existing threats, leading to refinement

or revision of existing domain knowledge underpinning security requirements. The evolution of the covertness of C2 communication channels provides a good example of mutations of existing threats. In response to C2 server domain names being blacklisted and C2 traffic showing as outliers on traffic flow analysis, attackers are adapting their approaches, e.g., using social networks as a C2 server resolution service or blending C2 traffic with packets destined for cloud services.

Fully unknown knowns, on the other hand, capture new and emergent threats and represent new domain knowledge that needs to be integrated into existing security models. For instance, novel exfiltration methods are continuously making the transition from the lab to the wild and the gestation period of a vulnerability exploit from its inception to its availability in popular exploit kits is now relatively short. For example, the once-novel DNS tunneling tools were made available in popular exploit kits such as Metasploit and Blackhole within months of their discovery. New exfiltration schemes such as VoIP Steganography (concealing secret messages/data within VoIP streams without severely degrading the quality of calls) [23] and use of timing channels [17] are emerging threats for which existing security models have no control in place.

Our approach also enables the identification of relevant points within an existing security model where such updates need to be applied. Its flexible nature enabled us to structure the security model emerging from our analysis along the same classification as that utilised in CSC20. This, in turn, allowed a direct comparison across key concepts and controls in the two models to identify the various points at which refinements/revisions or additions are needed.

## 6.3 IFT as a linchpin for knowledge capture and aggregation

Because of the wealth of data contained within incident documentation, the conventional grounded theory guidelines of describing and elaborating concepts in terms of their properties put the analysis process at risk of drifting too far into the description of an incident, at the cost of its explanation. IFTs, as the *atomic model*, facilitate the causal explanations of incidents by enabling the selection and organisation of pertinent information, which may otherwise be scattered in the documentation. Use of IFT requires the analyst to actively search for the relevant incident event, its causes, enablers, consequences and preventions. This ensures that the analysis remains focused on drawing out explanations for the incident rather than documenting the incident's description. The emphasis on causal relationships within an IFT ensures that explanation is not traded off in favour of documentation. Furthermore, the populated IFTs provide a source of richly expressed domain knowledge otherwise scattered across raw data from a variety of incidents. For instance, the *inhibit* gates make it possible to pinpoint potential requirements with respect to specific threats. These, in turn, can be mapped to misuse cases, describing sequence of actions taken by the attacker with respective remedial actions. Similarly, the higher levels of an IFT can be mapped to anti-goals, with lower levels mapping to goal refinements and tasks. The remedial actions thus correspond to solutions to these anti-goals tasks.

## 6.4 Data and model quality

As our work is based on a data-driven method (i.e., the model is first and foremost extracted from the available data), the quality of the input data is clearly critical. Of course, the analyst should not choose a source that is irrelevant, clearly mis-directed, or otherwise unsuitable. The concerns about the quality of a given data source are further mitigated by using a number of different input sources and the key role of the analyst—who systematically processes and questions the data by eliciting cause, action, result and effect and filling in missing information through various iterations. Furthermore, peer-coding and feedback can also be used, whereby more than one analyst independently processes the given data, with the results of such independent work critically reviewed by peers (in the team) and then integrated and harmonised into an agreed upon model.

## 6.5 Where do unknown knowns hide?

Our experience shows that unknown known security requirements are implicit in patterns of cause-effect-prevention relationships across a range of security incidents. Analysis of a single incident using an IFT can reveal tacit knowledge relevant to that particular breach but not the patterns that can lead to identification of gaps in a prevalant conceptual model. While the majority of unknown knowns hide in patterns across incidents, we did identify instances (e.g., the vagueness of CSC 17-4: cloud provider security practices) where typical patterns of imprecision can identify a lack of knowledge or the need to discover further knowledge to plug the gaps in the model. Identification of such gaps can guide the analyst in chosing incidents for analysis and/or the search for particular unknown known security requirements.

# 7. CONCLUSION AND FUTURE WORK

Our method for discovering unknown known security requirements is aimed at refining theoretical security models so that they remain pertinent in the face of a continuously changing threat landscape. IFTs enable the requirements engineer to perform causal analysis of the security incidents under consideration. The rigorous and iterative processes of open-coding and memo writing force the requirements engineer to, firstly, rationalise the creation of categories for new exploit, threat and countermeasure concepts. Secondly, by selecting incidents to analyse in an attempt to saturate the categories recorded in memos, the requirements engineer ensures that requirements derived from these memos are as near-complete as possible. Furthermore, memo-writing enables one to aggregate identified concepts, extrapolate on them (e.g., consider the applicability of countermeasures to different threats) and correlate various patterns of attacks and solutions across the analysed incidents. Thus the method ensures sophistication of coverage afforded by the coded countermeasures. Our evaluation in a real-world context demonstrates that the rich picture resulting from the method indeed leads to identification of security requirements that may be partially or fully unknown known. Our future work will explore this distinction further, aiming to derive a taxonomy of partially and fully unknown known security requirements to guide the analyst. The qualitative nature of our method makes it possible to structure our derived model in a format that should enable comparison with any pre-existing conceptual model to discover unknown knowns with respect to that model. Such investigation is another avenue for future work along with our method's potential to discover unknown knowns in domains other than security.

## 8. REFERENCES


[1] The critical security controls for effective cyber defense, version 5.1. Council on Cyber Security, 2015.

[2] M. Almorsy, A. Ibrahim, and J. Grundy. Adaptive security management in saas applications. In S. Nepal and M. Pathan, editors, *Security, Privacy and Trust in Cloud Systems*, pages 73–102. Springer Berlin Heidelberg, 2014.

[3] G. Elahi, E. S. K. Yu, and N. Zannone. A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. *Requir. Eng.*, 15(1):41–62, 2010.

[4] V. N. L. Franqueira, Z. Bakalova, T. T. Tun, and M. Daneva. Towards agile security risk management in RE and beyond. In *First International Workshop on Empirical Requirements Engineering, EmpiRE 2011, Trento, Italy, August 30, 2011*, pages 33–36, 2011.

[5] B. G. Glaser and A. L. Strauss. *The Discovery of Grounded Theory*. Aldine Publishing Co., New York NY, 1967.

[6] B. Hollis and N. A. M. Maiden. Extending agile processes with creativity techniques. *IEEE Software*, 30(5):78–84, 2013.

[7] J. Horkoff, N. A. M. Maiden, and J. Lockerbie. Creativity and goal modeling for software requirements engineering. In T. Maver and E. Y. Do, editors, *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition, C&C '15, Glasgow, United Kingdom, June 22-25, 2015*, pages 165–168. ACM, 2015.

[8] C. W. Johnson. *A Handbook of Incident and Accident Reporting*. Glasgow University Press, 2003.

[9] N. W. Knox and R. W. Eicher. Mort user's manual for use with the management oversight and risk tree analytical logic diagram. System Safety Development Center, Idaho National Engineering Lab., 1992.

[10] L. Lin, B. Nuseibeh, D. C. Ince, M. Jackson, and J. D. Moffett. Introducing abuse frames for analysing security requirements. In *11th IEEE International Conference on Requirements Engineering (RE 2003), 8-12 September 2003, Monterey Bay, CA, USA.*, pages 371–372, 2003.

[11] N. A. M. Maiden, C. Ncube, and S. Robertson. Can requirements be creative? experiences with an enhanced air space management system. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*, pages 632–641. IEEE Computer Society, 2007.

[12] M. Paradies, L. Unger, P. Haas, and M. Terranova. Development of the nrc's human performance investigation process (hpip). Division of Systems Research, Office of Nuclear Regulatory Commission, NUREG/CR-5455, SI-92-101, Vol 2, 1993.

[13] A. Patel, N. Sharma, and M. Eirinaki. Negative database for data security. In *Proceedings of the 2009 International Conference on Computing, Engineering and Information*, ICC '09, pages 67–70, Washington, DC, USA, 2009. IEEE Computer Society.

[14] P. Ruddick, K. Hannah, C. P. Schade, G. Bellamy, J. Brehm, and D. Lomely. Using root cause analysis to reduce falls in rural health care facilities. In *Advances in Patient Safety: New Directions and Alternative Approaches, Vol. (2)*. 2008.

[15] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh. Requirements-driven adaptive security: Protecting variable assets at runtime. In *2012 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, USA, September 24-28, 2012*, pages 111–120, 2012.

[16] P. Sawyer, V. Gervasi, and B. Nuseibeh. Unknown knowns: Tacit knowledge in requirements engineering. In *RE 2011, 19th IEEE International Requirements Engineering Conference, Trento, Italy, August 29 2011 - September 2, 2011*, page 329, 2011.

[17] G. Shah and A. Molina. Keyboards and covert channels. In *Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 - August 4, 2006*, 2006.

[18] G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requir. Eng.*, 10(1):34–44, 2005.

[19] C. Tsigkanos, L. Pasquale, C. Menghi, C. Ghezzi, and B. Nuseibeh. Engineering topology aware adaptive security: Preventing requirements violations at runtime. In *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, pages 203–212, 2014.

[20] A. van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *26th International Conference on Software Engineering (ICSE 2004), 23-28 May 2004, Edinburgh, United Kingdom*, pages 148–157, 2004.

[21] L. M. Veltman. Incident data analysis using data mining techniques. Master's thesis, Texas A & M University, 2008.

[22] Y. Yu, H. Kaiya, H. Washizaki, Y. Xiong, Z. Hu, and N. Yoshioka. Enforcing a security pattern in stakeholder goal models. In *Proceedings of the 4th ACM Workshop on Quality of Protection, QoP 2008, Alexandria, VA, USA, October 27, 2008*, pages 9–14, 2008.

[23] E. Zielinska, W. Mazurczyk, and K. Szczypiorski. Trends in steganography. *Commun. ACM*, 57(3):86–95, 2014.