Cybersecurity: Causality and Ontological Parsimony

Mike Surridge[a], Samuel M. Senior[a], Duncan Guthrie[a]

[a]*IT Innovation Centre, University of Southampton, Southampton, SO17 1BJ, U.K.*

Abstract

Cybersecurity risk assessment using standards like ISO 27005 is hard, especially for complex target systems. The main challenges are to identify threats, estimate their likelihood, and determine their consequences. One source of difficulty is the presence of (system-specific) dependencies, whereby a threat to one system component can lead indirectly to consequences in other system components via (system-specific) attack paths and secondary-effect cascades. This paper postulates that part of the problem is that threat paths require an analysis of causes and effects in the context of a specific system, but tools and knowledge bases used for this are not based on causal models. Existing cybersecurity knowledge bases may help identify threats but do not allow the likelihood of adverse outcomes including indirect effects to be found in a target system. This paper proposes to address some of these challenges by using a simple causal model. Such a model provides at least three attractive benefits for knowledge capture and reuse: parsimony (the number of distinct concepts is small compared to existing knowledge bases), generality (it is feasible to capture these concepts in a way that does not require assumptions about target systems), and utility (it is easy to create simulations of cybersecurity threats in target systems and determine the presence and likelihood of attack paths and secondary effects). This has implications for the development of current and future cybersecurity knowledge bases. For example, the high cost of analysing new vulnerabilities in the NVD catalogue seems related to the fact that CVSS (in its current form) hides some causal relationships. This suggests that aligning CVSS with a causal model would reduce the cost of NVD and make it more useful in risk assessment. Mapping CVSS and other cybersecurity knowledge bases to a simple causal model would also make it easier to integrate them in a way that supports application in practical risk assessments.

Keywords: Cybersecurity, Risk Assessment, Semantics

1. Introduction

Standards like ISO 27005 [1] provide a recipe for analysing information security risks in terms of the likelihood and impact of threats affecting a target system. However, identifying cybersecurity risks and determining their likelihood is not easy. ISO 27005 proposes two alternative approaches to help overcome these difficulties: asset-based and event-based analysis. In event-based analysis, the system is treated as an ecosystem and uses a top-down approach to find risk sources (agents with motivation and ability to cause harm) and harmful events they can cause. In the asset-based approach, the system is described in terms of business (or primary) assets and supporting (or secondary) assets. Then threats are identified that would affect those assets in operational scenarios, their likelihood and impact are estimated, and the associated risks are determined. ISO 27005 claims that using the asset-based approach 'if all valid combinations of assets, threats and vulnerabilities can be enumerated within the scope of the ISMS, then, in theory, all the risks would be identified'. This is a bold claim, but depends on being able to identify all these elements. This creates a requirement to develop and maintain cybersecurity knowledge bases to support risk analysts, examples of which include the US National Vulnerability Database (NVD) [2] and the ATT&CK and D3FEND frameworks from Mitre [3, 4].

Many threats do not directly compromise the system, but do so indirectly in a manner best described by a sequence of threats known as a *threat path*. There are two main types of threat path: attack paths and secondary effect cascades. An attack path arises when the effects of one threat enable exploitation of a system weakness by another threat, so an attacker can harm a target system through a series of steps. A secondary effect cascade occurs when the effects of one threat cause further effects due to a system dependency, with no attacker involvement. In both cases, effects can be caused that are far from the first threat. This makes it difficult to recognise that they are indirect effects of the first threat and may lead to underestimation of their likelihood.

Some cybersecurity knowledge bases now attempt to address these difficulties by including indirect effects when describing threats or related aspects, such as vulnerabilities. However, threat paths arise because of dependencies within systems, so they are naturally system-specific. Because those depen-

dencies may or may not exist in a given target system, a risk analyst must decide whether or not each risk described by such a knowledge base is relevant to their system. If they cannot do this accurately, risks may be overestimated leading to excessive precautions, or underestimated leaving systems exposed. Developers of such a knowledge base must also decide what assumptions to make concerning system dependencies, and may need to cover several combinations of assumptions. This makes development more complex and leads to a completeness problem: which combinations of assumptions are needed to fulfil the ISO 27005 promise that 'all the risks would be identified'?

This work described here addresses these challenges by recognising that threat paths and threats are inherently causal mechanisms. If threats or related concepts are described in terms of causes and effects, then threat paths in a given target system can be found and analysed more easily by constructing a causal model of the system. A knowledge base describing such threats becomes simpler and more self-contained, and it is easier to recognise when these threats arise in the target system. It is still necessary to model dependencies within the target system, but this is relatively easy if the knowledge base and tools used are designed to support it.

This approach was used in the implementation of the prototype risk analysis software Spyderisk [5]. This uses a knowledge base that describes system components, threats, and their causes and effects. Here, the focus is not on the software prototype, but on the approach used to create the knowledge base from a causal model, and the implications and opportunities for development of new and existing cybersecurity knowledge bases.

2. Related work

2.1. Cybersecurity knowledge capture

Cybersecurity terminology has been developed over decades and is very rich. This is a strength, but also a weakness, as there are many overlaps and ambiguities. RFC 4949 [6] captures a large number of terms used in Internet Security, but many terms have multiple subtly or not so subtly distinct meanings. Standards in this area must carefully define their terms, and the ISO 27000 series of standards for information security risk management begins with the overview and vocabulary of ISO 27000 [7]. However, this, like RFC 4949, is not a true taxonomy, but a dictionary that provides definitions of terms that are often overlapping or ambiguous.

Ontologies provide a way to resolve some of these difficulties. Herzog et al. [8] and others produced examples of cybersecurity ontologies, and subsequent developments include the well-known models published by Fenz and Ekelhart of ISO 27001 and the German IT Grundschutz Manual [9]. Several reviews of such ontologies have also been published, by Blanco et al. [10] and more recently by Meriah and Rabai [11], which highlights the use of a recognised ontological development methodology, derived from well-known and/or standardised sources and providing complete coverage of cybersecurity terms like 'risk', 'threat', 'vulnerability', 'security attribute', and so forth. This allows cybersecurity information to be digitised and shared, but retains some of the ambiguities and overlaps of cybersecurity language, making it less suitable for machine inference and the analytical procedures of risk assessment.

Foundational ontologies provide a rigorous conceptual baseline on which other ontologies can be based, which in principle helps to resolve ambiguity, reduce duplication, and integrate ontologies through the common foundation. Some have been proposed for use in cybersecurity. The Basic Formal Ontology (BFO) [12] was used in the Common Core Cyber Ontology (C3O) [13], which models cyber assets and events and can be used to describe cyber attacks (which form a subclass of events) although it does not include models of specific types of cyber attack. The C3O has been proposed as a way to model cybersecurity in the Internet of Things [14]. The Universal Foundational Ontology (UFO) [15] has been used by Oliveira et al. as a starting point for a security (not cybersecurity) ontology from a risk treatment perspective [16]. This ontology is discussed in more detail later in Section 5.1.

However, few cybersecurity researchers have used foundational ontologies to capture cybersecurity knowledge. Early work by Herzog, Fenz and others predates the development of foundational ontologies, but they are still not widely used today. The starting point is still cybersecurity terminology, and increasingly previous knowledge bases, and the main challenge is still the presence of ambiguity and overlaps. For example, the D3FEND ontology was bootstrapped by manual analysis of existing knowledge bases using domain expertise to ensure a level of consistency that could not be obtained from machine learning or by importing existing sources [17]. Even efforts to unify disparate knowledge repositories do not rely on foundational ontologies. The Unified Cyber Ontology (UCO) proposed by Syed et al. became a community effort to create a mid-level ontology, arguing that providing or choosing a foundational ontology would limit opportunities to integrate and

exploit knowledge bases with disparate or no such foundations [18]. UCO can be overlayed on a foundational ontology, allowing users to choose to use one, although presumably this would diminish the range of existing knowledge bases that they could then integrate. The proposed Unified Ontology for Cyber Security published by NIST [19] uses a relationship overlay called WAVED to relate concepts from different knowledge bases, without recourse to a foundational ontology. These integration approaches allow one to query multiple knowledge bases to find information, but do not directly support an ISO 27005 risk analysis procedure.

2.2. Cybersecurity knowledge and causality

The idea that threat paths are fundamental to cybersecurity risk assessment is also not new. They were proposed by Phillips and Swiler as a way to perform a cybersecurity risk assessment using methods then established in safety analysis [20], and used in Bruce Schneier's article showing how attack trees must be considered when evaluating threats to a target system [21]. These early developments focused on attack paths, but secondary effect cascades are also important in risk assessment. In 2008, this was raised in an EC Directive [22] concerning cyberphysical critical infrastructures, where the dependencies between physical and IT elements complicate risk analysis and can lead to risk amplification.

Causal models provide a way to calculate the effects of dependencies in threat paths composed of simpler threat models, but most cybersecurity applications have focused on probabilistic analysis, as proposed by Phillips and Swiler, and not on the development of reusable knowledge bases. For example, Guariniello and DeLaurentis showed how to determine risks using links between systems described in terms of their strength and criticality [23]. This is a very high-level approach, where the links are system-specific, which does not lead to reusable (system-independent) elements that could be included in a knowledge base. Poolsappasit et al. [24] used a set of attack templates involving transitions between states, but each template may include multiple transitions, and the states can refer to anything in the target system.

Schneier formulates attack trees for a specific target system, but points out that an attack tree against one system will also work against similar systems. This idea was used in the ATT&CK knowledge base, which is formulated in terms of tactics (adversarial goals), techniques, and procedures (TTP). Schneier's claim that attack trees are reusable is clearly valid at

some level, as demonstrated by the creation of ATT&CK, but its elements are complex and expressed using STIX 2.1 [25], which is designed for information sharing rather than reasoning. ATT&CK is primarily a source of threat intelligence, describing potential adversaries and their *modus operandi*. D3FEND is a complementary knowledge base describing defensive cybersecurity measures and their relationship to ATT&CK techniques. D3FEND supports reasoning to determine whether a set of defensive technologies addresses potential attacks, but not the causal analysis required in risk assessment.

The Common Vulnerability Scoring System CVSS represents threat paths in a different way. Version 2, published in 2007 [26] was adopted as a schema to classify software vulnerabilities by (NVD). Each vulnerability is modelled using descriptive metrics, whose values can be used to calculate a severity score, which became seen as a proxy for the risk posed by each vulnerability. This was criticised by authors such as Schoenfield and Quiroga [27], on the grounds that risk depends on all direct and indirect effects of exploiting a vulnerability. This was addressed in CVSS v3 [28] by extending the impact metrics to include indirect effects. However, this depends on assumptions about the target system, making it more difficult to determine the correct metric values for a vulnerability or to decide how relevant the results are to a given target system (see Section 5.2).

Most cybersecurity knowledge bases have a concept of threat which has (causes) consequences, but efforts to create knowledge bases for modelling threat paths are limited. In most cases, threat effects are explicitly included, but threat causes are not. Fenz et al included a causal relationship between different threats and used this in a Bayesian network analysis, but other threat causes are not included except to distinguish whether threats are accidental or deliberate of natural or human origin. Some authors of this paper created a knowledge base designed to capture secondary effect cascades and used it to create a Bayesian network to support root cause analysis in a given target system [29]. However, a Bayesian network contains many parameters, including conditional causation probabilities. These are difficult to obtain and system-specific, so the generic threat models from the knowledge base could not provide the information needed to address arbitrary target systems.

To address these limitations and better align with ISO 27005, the Bayesian formulation used in [29] was replaced by a qualitative approach, leading to the Spyderisk software described by Phillips et al. [30]. This simplifies risk calculation and eliminates many system-specific parameters, allowing a knowledge

base of reusable threat models to be created. This is quite different from either ATT&CK or CVSS. The focus here is to describe the insights gained from this transformation, including potential implications for risk analysts and developers of cybersecurity (or other risk-related) knowledge bases.

3. Causal Model Formulation

3.1. Asset behaviours and risks

Causal models have been extensively studied in the literature in fields ranging from the design of clinical trials, to data driven knowledge discovery using machine learning algorithms. There are three main approaches, usually attributed to Campbell [31], Rubin [32] and Pearl [33]. Pearl's approach is based on causal relationship graphs, making it naturally suited to model cybersecurity risks including threat graphs. This starts from a structural causal model, defined by Pearl as an ordered triple $< U, V, E >$, where:

- U is a set of exogenous variables whose values are determined by factors outside the model,
- V is a set of endogenous variables whose values are determined by factors within the model, and
- E is a set of structural equations that express the value of each endogenous variable in terms of the other variables.

The term *asset* is not defined explicitly in ISO 27000, but the overview says *information, and related processes, systems, networks and people are important assets for achieving organisation objectives.* A threat is *a potential cause of an unwanted incident, which can result in harm to a system or organisation.* ISO 27005 says that an asset is *anything that has value to the organisation and therefore requires protection.* These definitions imply that threats cause harm to the target system by compromising assets. They cause a change in the status or behaviour of assets, here referred to for brevity as a threat-induced *behaviour.*

Information system assets include information (data), processes (running software), hosts (devices that can store data and run software), and networks (over which connected hosts can communicate). Threat-induced behaviours include loss of confidentiality, integrity, or availability (CIA). People are also

important; although strictly speaking, when a human contributes to a system, the asset is not the human, but their fulfilment of a system role. Behaviours associated with human roles include loss of availability (nobody can fulfil the role), impersonation (an unauthorised person fulfils the role), and subversion (a person in the role is persuaded to harm the system).

Other assets may be even less tangible. For example, if two processes have a client-service relationship, a client impersonation threat would cause a loss of client authenticity. This cannot be a behaviour of the client or the service because it does not affect their relationships with other processes. The threat affects only their mutual relationship, which should therefore be modelled as an asset that requires protection.

Let the set of system assets be $A = \{A_a\}$, where a is a unique label for each asset, such as a URI. Let $B = \{B_a^b\}$ be the set of potential threat-induced behaviours in the system, where each asset A_a can have several behaviours whose types are distinguished by a second URI b. Each type of behaviour may occur for some or all of the system assets. The state of behaviour B_a^b can be modelled by a binary variable $V_a^b \in V$, where $V_a^b = \top$ (true) means that B_a^b occurs.

In ISO 27005, each potential threat is assigned an *impact* representing the amount of harm caused by its effects. The *risk* then depends on the impact and the *likelihood* of occurrence. A causal model can be used to find the likelihood of each behaviour $L(V_a^b = \top)$, henceforth written as $L(B_a^b)$. In Pearl's formulation, likelihood is represented as a mathematical probability, but this is not the only option. ISO 27005 allows for qualitative or quantitative determination of likelihood and impact, and notes that the cost of quantifying likelihoods may be high and 'it can be sufficient to use initial and rough estimates of likelihood'. Qualitative approaches are therefore widely used in system-level risk analysis.

This paper, like ISO 27005, does not specify the representation of likelihood. How it is represented and determined may depend on how the model is used and with what inputs. This is discussed further in Section 3.5.

3.2. Threat effects

The status of a behaviour is driven by the presence or absence of threats, so each V_a^b is an endogenous variable, whose value is given by a structural function $E_a^b \in E$. Let $T = \{T_i\}$ be the set of potential threats in the system, whose status is modelled by a further binary variable $V_i \in V$, where $V_i = \top$ means that the threat occurs.

Let $T(b,a) \subset T$ be the set of threats that cause behaviour B_a^b. Each threat is a *sufficient* cause of its effects, so the structural function that gives V_a^b should be a logical union of the threat status variables. If $T(b,a) = \{T_{i_1}, T_{i_2}, ..., T_{i_n}\}$

$$V_a^b = V_{i_1} \vee V_{i_2} ... \vee V_{i_n}$$

where $\vee$ represents the logical union of its arguments. This can be written more succinctly

$$V_a^b = \bigvee_{i:T_i \in T(b,a)} V_i \tag{1}$$

Equation (1) describes the effects of threats, but not their causes. These should be described by structural functions E_i that give the values V_i in terms of the presence or absence of causes. These functions model how a threat arises and how it can be prevented.

3.3. Threat causes

3.3.1. Secondary threat causes

Secondary threats model the propagation of threat effects due to system dependencies with no need for a malicious agent. They are caused by the effects of other threats, which means that an asset behaviour must also be a type of threat cause. In principle, several behaviours may be needed, each being *necessary* to cause the threat. The status V_i of T_i therefore involves a logical intersection over their status. If $B(i) = \{(b_1, a_1), ..., (b_n, a_n)\} \subset B$ are the behaviours that cause a secondary threat T_i, then

$$\begin{aligned} V_i &= B_{a_1}^{b_1} \wedge ... \wedge B_{a_n}^{b_n} \\ &= \bigwedge_{b,a:B_a^b \in B(i)} V_a^b \end{aligned} \tag{2}$$

where the operator $\wedge$ represents the logical intersection of its arguments.

Secondary effect cascades arise when the effects of one secondary threat cause another secondary threat, whose effects cause another secondary threat, etc. Equations (1) and (2) capture this and allow secondary effect cascades to be determined from causal relationships. It is not necessary to identify the indirect effects of each threat in advance. They are found by the model.

Where redundancy mitigates against the propagation of threat effects, this is modelled by the intersection operator in equation (2). For example, if a process cannot be executed without an input, then loss of availability

at the input causes a loss of availability at the process (it cannot start). If the process has several sources of input, then the status of each input is a distinct secondary cause of the threat to the process, and from equation (2) they must all be unavailable for the process to be affected.

However, not all secondary threats involve redundancy. If the input was incorrect and the process had no way to detect this, the incorrect input would cause errors in the process. With several sources of input, errors in any source would cause errors in the process. Equation (2) does not cover this, but it can be modelled by making each data source the cause of a separate threat to the process. The fact that an error in any source is sufficient to affect the process is modelled by the logical union operator in equation (1).

Together, equations (2) and (1) can model any secondary effect cascade.

3.3.2. Primary threat causes

The only other threat cause that could exist must be one that is not an effect of any threat. Such a threat cause must be one of two types:

- causes that are equivalent to asset behaviours, even though not caused by a system threat, which are the focus in this section, and
- causes that are not equivalent to asset behaviours, which are discussed later in Section 3.3.3.

Threat causes equivalent to asset behaviours arise because in ISO 27005 a risk assessment must have a defined scope. An asset is considered part of the target system if it contributes to that system. Other assets are considered external and are not included explicitly in the risk analysis. During the lifetime of an asset, it may join a system, leave again, or never join. It may interact with external assets before or after joining the system, and these interactions are a source of external influences on the system. If the target system were widened to include those external assets, those influences would become threat effects, but being external, they are not caused by any threat in the causal model. Note that moving the system boundary to include external assets does not eliminate this type of threat cause because external influences on the previously external assets must then be added.

Threats with causes of this type are called primary threats because they can occur in the absence of other threats. Primary causes also provide a way to model agency, where the threat cause embodies a choice. This choice depends on the agent's prior experiences, which are by nature external to the

target system. The agent is usually outside the system and not represented explicitly in the causal model, but they may be the asset that causes the threat, in which case the threat is an insider threat.

Because the effect of an external influence is equivalent to an asset behaviour, primary threats can (like secondary threats) form threat paths, known as attack paths. Consider a system that has a server connected to a LAN provided by a router connected to the Internet. Its users are people using client devices to access and use the server remotely from the Internet. When the system is working as expected, they connect via the router to use the server as intended. An incomplete list of threats in the system would be:

1. If the legitimate user of a client device is untrustworthy, it has an untrustworthy user.
2. Untrustworthy users of a client device can access the Internet.
3. Untrustworthy users of the Internet may gain access to the router.
4. Untrustworthy users of the router can access the LAN.
5. Untrustworthy users of the LAN may gain access to the server.
6. Untrustworthy users of the server may be able to crash the server.
7. If the server is not available, it will be inaccessible to legitimate users.

Threat 7 is a secondary threat of the type discussed in Section 3.3.1. The rest are primary threats expressed in terms of causes and effects.

Threat 1 represents an insider attack by a legitimate user, which is a primary threat because it involves agency. The effect of threat 1 is a cause of threat 2 whose effect is to give untrustworthy users access to the Internet, a cause of threat 3. The attack path continues through threats 4, 5 and 6, leading to loss of availability on the server. This causes secondary threat 7, which disables access for legitimate users.

Threat 1 is not the only one with external causes. Threat 3 is caused by untrustworthy users of the Internet. In any reasonable target system, this is a shared asset with external users, many of whom are not trustworthy. Threat 3 can therefore occur even if threats 1 and 2 do not, so there is a shorter threat path that models an external attack to disable the system, with root-cause threat 3. The router, LAN, server, or client devices may be dedicated to the system, but since they may join other systems at any time, the possibility of external users cannot be excluded. Threats 2, 4, 5 and 6 therefore also have primary causes.

This example shows that a primary threat cause is really a pair of causes: a behaviour B_a^b and an equivalent external influence W_a^b. This can be mod-

elled by a binary exogenous variable U_a^b, where $U_a^b = \top$ signifies that the external influence made the associated asset A_a act as a threat cause as though it was affected by behaviour B_a^b. Either B_a^b or W_a^b is sufficient to make A_a do this, so the presence of a primary cause is modelled by the union $V_a^b \vee U_a^b$. When a primary threat cause is due to W_a^b, the threat is a root cause of an attack path. However, when it is due to B_a^b, the primary threat must have a predecessor threat in the attack path.

A primary threat may have several primary causes, all of which must be present for the threat to occur. If threat T_i has primary causes equivalent to the set of behaviours $W(i) \subset B$, the function for V_i is

$$V_i = \bigwedge_{b,a:B_a^b \in W(i)} (V_a^b \vee U_a^b) \tag{3}$$

It is helpful to think of W_a^b as a *trustworthiness attribute* undermined by the behaviour B_a^b on the same asset A_a. It represents the extent to which A_a can be trusted not to act as a threat cause even when it is not affected by the effects of other threats. In the example, threat 3 causes loss of user trustworthiness on the router, which causes threat 4. However, threat 4 is not caused by loss of trustworthiness, but by lack of trustworthiness. W_a^b models the possibility that A_a lacks trustworthiness even when not affected by other threats.

3.3.3. Exploitation of weaknesses

The other possibility identified at the start of Section 3.3.2 is a threat cause that is not caused by any threat and is not equivalent to any asset behaviour. This type of cause arises from the presence of a weakness in the system that can be exploited by the associated threat.

Weaknesses arise when a system contains one or more assets whose nature and relationships provide ways for something to go wrong. In most cases, exploitation can be prevented, so weakness can be ascribed to the absence of security measures. For example, if two processes have a client-service relationship, there is a weakness arising from the fact that the client could be impersonated to the service, unless client authentication is implemented. This weakness is exploited by a threat of unauthenticated access whose effect is impersonation of the client to the service. There may be several different security *control strategies* that address a system weakness. In this example, one might block access except from a pass list of client IP addresses, or

use X.509 client certification, or require each client to have a username and password, etc. The absence of each control strategy then acts as a necessary threat cause, as the weakness and hence the threat would be prevented if any of these control strategies were implemented.

Expressing exploitable weaknesses in terms of the absence of control strategies is a more robust approach for risk assessment purposes than modelling weaknesses explicitly. If a risk analyst omits a control strategy from their model, the risks will be overestimated, not underestimated. When the threat poses an unexpected risk, the analyst will be prompted to investigate and will likely correct the oversight. If the threat is considered to exist only if there is a system weakness, overlooking the weakness will mean that the threat is omitted, the risks will be underestimated, and the analyst will never be prompted to check anything.

Let $G = \{G^g\}$ be the set of control strategies that could be implemented in the target system. The status of each control strategy can be modelled by a variable V^g, where $V^g = \top$ means that G^g is absent (or faulty). Although control strategies are not affected by threats, this is an endogenous variable for reasons explained below. Let $G(i) \subset G$ be the set of control strategies that could address the weakness exploited by the threat T_i. The absence of each $G^g \in G(i)$ is a necessary cause of T_i, so

$$V_i = V_i^{(0)} \wedge \bigwedge_{g:G^g \in G(i)} V^g \tag{4}$$

where $V_i^{(0)}$ is the status of the threat if no control strategy could prevent exploitation of the weakness, found by equation (2) or (3).

3.4. Security controls

Control strategies are properties of the system, not (as with behaviours and trustworthiness attributes) properties of individual assets. However, each threat involves specific assets and is prevented by specific control strategies, so a relationship must exist between control strategies and assets. If a system has two services, each with its own client, one can implement client authentication between one client and service, but not the other. Relationships to relevant assets must be included in the model of a control strategy.

This can be done using *controls* representing security control measures that may be implemented in the system. Let $C = \{C_a^c\}$ be the set of possible controls in the system. Each asset may have several controls distinguished

by the label c representing the type of control, each of which may be implemented at a subset of system assets. Controls are not affected by threats, so the status of C_a^c can be modelled by an exogenous variable U_a^c, where $U_a^c = \top$ means that the control of type c is not implemented (or faulty) at A_a.

Each control strategy G^g can then be modelled in terms of its contributing controls $C(g) \subset G$, which is why the status of G^g was modelled by an endogenous variable V^g. G^g is implemented if and only if all its controls are implemented, so the status of each control strategy is given by

$$V^g = \bigvee_{c,a:C_a^c \in C(g)} U_a^c \tag{5}$$

For example, one way to implement client authentication is to have the client present a username and password to the service. Control properties can be used on the client and the service to represent the ability of the client to send and the service to verify a shared secret password, respectively. If a control strategy consists of these two controls, and both are present ($C_a^c = \bot$), then equation (5) says the control strategy is present ($V^g = \bot$), so (4) gives $V_i = \bot$ (the threat is prevented) even if $T_i^0 = \top$.

Depending on the type of control, the association with A_a means either that the control is implemented by A_a or that it protects A_a. The client-service weakness could be prevented by restricting connections to IP addresses of legitimate clients. This could be as an IP pass-list enforced by a router with no changes in the implementation of the service. This strategy consists of a pass-list policy and an enforcement point. Both are features implemented on the router, but only the enforcement point can be represented as a property on the router. The pass-list protects a specific service, so it must be modelled as a property of that service.

Where trustworthiness attributes allow the system model to have a bounded scope, control attributes allow bounds on its level of detail. Most security controls are of value to a system, so their implementation can be considered as system assets. Representing controls by asset properties allows these implementation assets to be omitted, simplifying the system model. For example, a service can implement password verification by redirecting log-in attempts to a single-sign-on (SSO) service. One could add this to a system model as a new service related to the client and the service. Its presence would negate the inherent weakness in a simple client-service relationship and with it the threat of unauthenticated access, so the original control strategy would not

be needed. However, as with trustworthiness attributes, control strategies and properties cannot be eliminated by including more assets in a system model. The dependence on the SSO service creates a new weakness and new threats. For example, if a client is impersonated in the client-SSO relationship, it can be impersonated to the original service. Control strategies are then needed for threats to the client-SSO relationship.

This example makes it seem that control properties should be modelled as endogenous variables because they are affected by threats. However,

- the original threat is still prevented, here unauthenticated access to the first service, and
- the original control may still apply to other threats, here threats involving other clients that do not use the SSO service.

The new threats bypass the controls but do not change their status.

Finally, since control properties are modelled using exogenous variables, they do not depend on each other. If a control protects many assets, the same control property should be used in all relevant control strategies. This is possible only if the implementing asset is included in the model, as in the router example. If a single point of failure is not included, it should be added, as in the SSO service example. Conversely, one can simplify the model by removing assets if their only purpose is to implement security measures that are not single points of failure.

3.4.1. Control weaknesses and side effects

Some system weaknesses arise from the presence rather than the absence of security controls measures. In the client-service scenario, the use of a password creates weaknesses that allow other threats:

- the password may leak, enabling a threat of access using stolen credentials that bypasses or exploits the control, or
- the user may forget their password and become locked out of the service, which is a side effect of the control.

If the password were modelled as a system asset, such threats would be caused by a lack of confidentiality or availability of the password. Other threats with those effects could then form threat paths, enabling the new threats. For example, if the client is a browser used on a device where the

user also reads email, a phishing attack may cause a loss of confidentiality in the password, thus allowing the stolen credential attack.

If controls were modelled as asset properties, these threat paths would be overlooked. They should be modelled as a single threat whose causes include the presence of the control strategy that *triggers* the threat. The effects of the triggered threat are the effects of the missing threat path. This version of the phishing threat would have the same causes as before, plus the presence of the password authentication control strategy. It cannot affect the password because it is no longer an asset, so the effect of the phishing threat is the end-effect: loss of client authenticity in the client-service relationship.

If threat T_i is triggered, there must be a set $X(i) \subset G$ of trigger control strategies. If any $C^g \in X(i)$ is present ($V^g = \bot$), then T_i is triggered, but if none are present, T_i cannot occur, so

$$V_i = V_i^{(1)} \wedge \bigvee_{g:G^g \in X(i)} \neg V^g \tag{6}$$

where $V_i^{(1)}$ is the result of equation (4).

Combining equations (2), (3), (4), (5) and (6) gives the structural equation for the state V_i of any threat T_i

$$V_i = \left(\bigwedge_{b,a:B_a^b \in B(i)} V_a^b \right) \wedge \left(\bigwedge_{b,a:B_a^b \in W(i)} (V_a^b \vee U_a^b) \right) \wedge \left(\bigwedge_{g:G^g \in G(i)} \bigvee_{c,a:C_a^c \in C(g)} U_a^c \right) \wedge \left(\bigvee_{g:G^g \in X(i)} \bigwedge_{c,a:C_a^c \in C(g)} \neg U_a^c \right) \tag{7}$$

If any of $B(i)$, $W(i)$, $G(i)$ or $X(i)$ is empty, then the union or intersection over its members should be omitted from this equation, equivalent to replacing that term by $\top$ (true). For example, a secondary threat that is not triggered or blocked by any control strategy has $W(i) = G(i) = X(i) = \emptyset$, so equation (7) reduces to equation (2).

3.5. Determining likelihood and risk

As mentioned in Section 3.1, for consistency with ISO 27005, equations (1) and (7) do not specify the representation or determination of likelihood. Any approach may be used, given a way to calculate $L(\neg x)$, $L(x \wedge y)$, and

$L(x \vee y)$ that is consistent with the properties of logical negation, intersection, and union.

In a classic ISO 27005 risk assessment, one can specify likelihoods for the exogenous variables:

- $L(W_a^b)$ represents trustworthiness assumptions about external influences on system assets,
- $L(C_a^c)$ represents assumptions about controls implemented at or for system assets.

and use e Equations (1) and (7) can be used with the rules for the likelihood of logical negation, intersection, or union to find the likelihoods $L(T_i)$ and $L(B_a^b)$ of each threat and threat effect.

If likelihood is represented in terms of probability, equations (7) and (1) can be used to create a Bayesian network to find $L(T_i)$ and $L(B_a^b)$. This may not be easy, since a Bayesian network uses many parameters to describe conditional probabilities, and (as noted in ISO 27005) it can be difficult to obtain accurate estimates for these or for the inputs $L(W_a^b)$. Another quantitative approach suggested in ISO 27005 uses the frequency of events or state changes to represent likelihood, in which case $L(T_i)$ and $L(B_a^b)$ could be found using a stochastic event simulation algorithm.

In a qualitative approach, likelihood is described using an ordered set of N terms such as {'low', 'medium', or 'high'}, labelled in order by integers in $[0, N-1]$. The likelihood $L(x) = k$ means that the likelihood of x is described by the k-th term. $L(T_i)$ and $L(B_a^b)$ can be found by initialising them to zero and repeatedly using equations (1) and (7), with logical operators given by $L(\neg x) = N - 1 - L(x)$, $L(x \wedge y) = \min(L(x), L(y))$, and $L(x \vee y) = \max(L(x), L(y))$. This procedure is computationally inexpensive and is guaranteed to converge.

The risk R_a^b of each threat effect B_a^b is then determined from its likelihood and impact I_a^b, an input that represents the degree of harm caused if that threat effect were to occur. This can be expressed as

$$R_a^b = I_a^b \circ L(B_a^b) \tag{8}$$

where the form of the function depends on how likelihood and impact are represented. In a quantitative approach, the impact can be expressed in terms of financial or human cost, whose expectation value becomes the measure

of risk. In a qualitative approach, impact and risk are usually represented using sets of terms, and equation (8) is often implemented as a look-up table. Note that equation (8) is a post-processing step, so I_a^b and R_a^b are not causal model variables.

A causal model can, of course, be used in other ways. For example, in a root cause analysis one starts with known values for a subset of threat effects V_a^b, and solves to find the likelihoods for potential root cause threats that may indicate the presence of an external attacker or the absence of a control measure. If likelihood is represented using probabilities, this can be done using a Bayesian search algorithm. A qualitative approach is less useful for such an analysis because there are few distinct values for each likelihood and there may be many equally likely threat paths consistent with the known input values.

3.6. Knowledge representation

The causal model described by equations (1) and (7) is defined in terms of assets with behaviours, trustworthiness attributes, and controls, plus threats and control strategies, as shown in Figure 1. The labels a, b, c, i and g in this figure have the same meaning as in the causal model equations and the arrowheads follow the conventions of UML indicating inheritance, aggregation, composition or a simple association.

The entities in such a model are divided into two main types:

- Structural entities, which model the composition of the target system. They are not part of the causal model, but provide a frame of reference for its composition and interpretation.

- Causal entities, which model threats and their causes and effects. They form the causal model, having causal model variables and likelihoods. They are related to and can be inferred from the structural entities.

To capture knowledge of these entities in a reusable form, it must be independent of any specific target system. This can be done by specifying:

- the types of structural entities (assets and relationships) from which target systems can be composed,

- the types of properties (behaviours, trustworthiness attributes and controls) possessed by each type of asset,

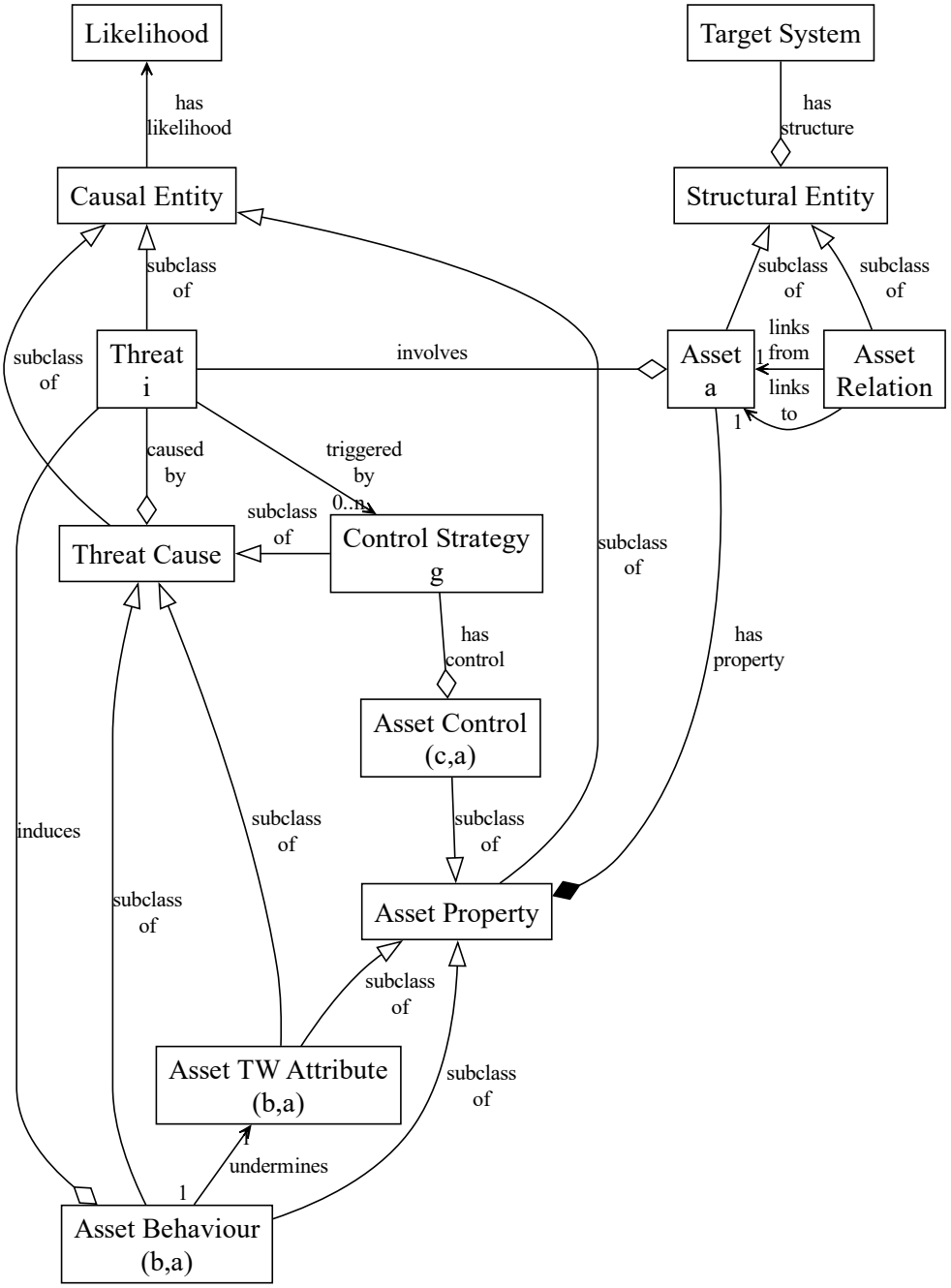

Figure 1: System model structure

- the types of threats that can arise, the roles played in them by assets, and the roles and properties of assets that act as primary or secondary causes or threat effects,

- the types of control strategies, the types of threats they block and/or trigger, and the asset roles and control properties that represent their implementation.

Each threat model is based on a set of related assets, whose presence implies the potential presence of a system weakness exploited by the threat. To make the threat model independent of any target system, one can specify everything in terms of the roles played by these assets. Each role is defined in terms of the types and relationships of assets that fulfil that role and whether the role is unique (filled by one asset in each threat) or not (filled by multiple assets), the latter case supporting redundancy models, as described in Section 3.3.1. Roles or relationships may also be prohibited, where the presence of assets or relationships negates system weaknesses so there is no threat, as discussed in Section 3.3.3. The causes and effects of the threat can be specified in terms of the properties of assets in specific roles, including the control properties that make up control strategies that would negate or trigger the threat.

3.7. Semi-automated risk assessment

Given a knowledge base formulated in this way and a model of a target system structure in terms of its asset types and relationships, pattern matching can be used to add threats with their causes and effects. The result is a causal model of the target system that can be used to calculate the likelihood and risk of each threat effect. To carry out a classic ISO 27005 risk assessment:

1. A structural model of the target system is created using related assets.
2. A causal model of the system is generated from this, adding asset properties, threats, and control strategies and their causal relationships.
3. A set of assumptions is added, specifying the likelihoods $L(W_a^b)$ that external influences will cause threats to the system and $L(C_a^c)$ that controls are implemented in the system.
4. Equations (1) and (7) are used to determine the likelihoods $L(T_i)$ and $L(B_a^b)$ of threats and threat effects.
5. Equation (8) is used to calculate the risk for each threat effect.

If risks are not acceptable, one can return to step 3 and change known likelihoods to model the effect of adding controls to the system, or one can return to step 1 and alter the structural model to reflect changes intended to eliminate weaknesses.

A key feature of this process is that the most difficult steps can be automated: step 2 using rules from the knowledge base described in Section 3.6, and step 4 using the solution procedure as described in Section 3.5. Step 5 can also be automated, although that step is not difficult.

This approach was used in the Spyderisk project, which produced the software described in [30], and recently published an ontology corresponding to Figure 1 [34]. The software uses equations (1), (7) and (8) to determine risks, although it uses classes to model populations of system assets, threats, etc., so the equations are modified accordingly. The knowledge base used for steps 1 and 2 [35] is an evolution of that used in [29] which was itself inspired by the work of Herzhog, Fenz and Ekelhart. It is interesting to reflect on how it was created, starting with the structure in Figure 1 and not with the terminology used in cybersecurity.

4. Creating a knowledge base

4.1. Overall procedure

To create a knowledge base aligned with a causal model, one starts by deciding the types of target system it should support, e.g., cyber-physical systems. An iterative process can then be used:

1. Identify types of assets and relationships from which the target systems in this domain of interest will be composed.
2. Identify types of behaviours applicable to each type of asset, or desirable trustworthiness attributes, which should have an equivalent 'loss of trustworthiness' behaviour.
3. Identify different ways unwanted asset behaviours could be caused. Each becomes a type of threat that involves the affected type of asset and other related assets that together embody a system weakness.
4. Identify causes of each threat in terms of the properties of assets with roles in the threat. These include primary or secondary causes and control strategies comprising combinations of control properties.
5. Identify new types of asset or relationship needed to model each threat.
6. Define rules to infer the presence of intangible assets.

The asset and relationship types from step 1 provide the vocabulary for defining the structural model of a target system. Step 2 provides the vocabulary for b in Figure 1. Step 3 leads to threat models, defined in terms of related asset roles, so they are independent of any specific target system. Step 4 provides the vocabulary for c in Figure 1 by finding control properties in control strategies that trigger or block these threats. Step 4 may also add new trustworthiness attributes or behaviours needed to model primary or secondary causes.

Steps 3 to 5 are sometimes impossible because previously identified asset and relationship types cannot capture the associated system weakness and model the causes of the threat. This may happen for several reasons:

- The threat does not affect all assets in a class, implying that further refinement of the asset class hierarchy is needed to distinguish those that are affected from those that are not.
- The threat should only affect assets of an existing type that have (or lack) relationships not yet included in the knowledge base.
- The causes or effects of the threat do not relate to assets in a single role, which implies the need for an intangible asset of a type that may need to be added, whose role is related to multiple existing roles.

The last of these corresponds to the discussion in Section 3.1, where a threat that exploits a weakness arising from a related client and service affects only their relationship, implying that the relationship should also be an asset. Risk analysts may overlook the presence of intangible assets, which should ideally be added automatically, hence the need to create inference rules for this in step 6.

Where new types of assets or relationships are added in Step 5, one must then return to Step 2 and determine what types of behaviour and trustworthiness attributes the new assets have. Work can then resume with Step 3, finishing incomplete threat models that involve the new assets or relationships, and adding threats that cause the new asset behaviours, bearing in mind that these may be hitherto ignored effects of existing threats.

Extensions of an existing knowledge base to cover new areas usually start at step 5. For example, one might have a cybersecurity knowledge base in which device theft or destruction is ignored. Later, cyberattacks might be added that require physical access. At that point, new asset classes are

needed to describe physical spaces, and one must model a whole new subdomain: how physical spaces are related, how they can be reached by intruders, and how this can be prevented. Ideally, such subdomains should be modular, so they can be used or not in each risk assessment depending on the scope.

4.2. Baseline threats

The best way to bootstrap this process is to start with *baseline* threats. These can be divided into three main groups, which model:

- normal system functions that change system behaviour or privileges
- secondary effect propagation due to internal system dependencies
- processes for starting up system elements unless they are disabled

In cybersecurity, the first group is important because it describes the use of system privileges. For example, if one has admin rights on a host, one can control who has user rights, start or stop processes, access stored data, and so forth. Each such action should be modelled as a threat whose causes include a trustworthiness attribute that represents the trustworthiness of those holding that privilege. This must have an equivalent undermining behaviour that can be caused by other threats to model ways of gaining that privilege. Attack paths often start with such a threat, followed by baseline threats that represent exploitation of privileges using normal system functions. Baseline threats and trustworthiness attributes can also be used to model the rights of unprivileged users, of specific processes running on hosts, or having physical access to spaces, etc.

The second group models aspects such as loss of availability propagating from a host to hosted processes, overload propagating from a process to its host, or incorrect process input leading to processing errors and thence to incorrect output. Although these threats do not model attacks, they may still be related to control strategies. For example, overload at a service would normally propagate to its host, but this can be prevented by reducing the priority of the service (a control strategy). In that case, the host is protected, but the control strategy triggers a different threat, whereby the overload causes the service to become unavailable, even though its host is neither overloaded nor unavailable.

Start-up processes cover actions such as switching on a host. If a host is switched off, it can only be attacked physically. Switching it on enables

other types of attack, so it makes sense to model switching on as a threat whose effect enables those other attacks. This effect is the behaviour of being *in service*, with a corresponding trustworthiness attribute of being *out of service*. This can be added as a necessary cause in many other threats to a host, so they occur only if it is in service. Modelling such processes as threats allows dependencies between components to be included (e.g., a process cannot be in service unless its host is in service), and disabling an asset to protect it can be modelled by a control strategy, the same as other security measures. The disablement strategy would protect the disabled asset and dependent assets by preventing them from being in service. However, it would also trigger a threat that causes a loss of availability in the disabled asset, which would spread through secondary threats to dependent assets.

Baseline threats are important because they, along with their causes and effects, allow a target system to be modelled in the absence of offensive threats. Their causes and effects provide a starting vocabulary to use when describing other threats.

4.3. Classification issues

It is not always easy to decide whether a concept should be modelled as a trustworthiness attribute or a control. In practice, this depends on which threat models are included in the knowledge base. For example, one way to model a software vulnerability is as a weakness exploited by threats. This means that the vulnerability is assumed to exist in an asset and be exploitable unless countered by a control strategy. Such a control strategy might represent measures such as penetration tests to find and remove vulnerabilities before the asset is deployed in a target system, or a software patching procedure to remove vulnerabilities after deployment. Note that in a risk assessment one might set $L(C_a^c) > 0$ for these controls to model the chance that $C_a^c = \top$ because a test failed to detect the vulnerability, or a patch was applied too late to prevent its exploitation.

An alternative is to assume that all software contains vulnerabilities, but they only become exploitable when they are discovered by attackers. With this approach, a threat is used to model the efforts of attackers to discover exploitable vulnerabilities in a software asset. Its effect is a behaviour that represents the discovery of a vulnerability in this asset. The equivalent trustworthiness attribute represents the absence of exploitable vulnerabilities, which can be combined with the behaviour as a primary cause of threats

that represent the exploitation of such a vulnerability. The absence of control strategies such as software patching or penetration testing is then a cause of the discovery threat, not the exploitation threat.

The advantage of the second approach is that if a vulnerability is known to be present in an asset, it can be modelled by setting $U_a^b = \top$ (or $L(W_a^b)$ to represent certainty) for the corresponding trustworthiness attribute. This will then cause exploitation threats even if vulnerability discovery is still prevented by control strategies. The first approach is simpler, which may make it more appropriate in some applications, but the presence of a known vulnerability can only be modelled by disabling all control strategies against the exploitation threat. Either approach leads to a valid causal model if it is consistent with the threats used in the knowledge base.

5. Existing cybersecurity knowledge bases

5.1. Terminological parsimony

The main benefits of a causal model grounding come from the fact that there are very few mathematically distinct entities and only three types of system asset properties: behaviours, trustworthiness attributes, and control properties. This is not the result of a design decision (it is intrinsic to the causal model), but this parsimony increases the potential benefits of machine reasoning. Using fewer distinct first-order concepts to describe something means that (all else being equal), fewer asserted facts are needed as input to infer useful conclusions. Less parsimony leads to a more complex input model from which it is more difficult to extract useful information, if it can be done at all. A good example is the inclusion in many ontologies of both system weaknesses and control measures as first-order concepts. This can make it harder to infer the presence of threats, as discussed in Section 3.3.3.

The language of cybersecurity is perhaps too rich for its own good. The lack of alignment with a simple model of causality makes it difficult to use this language when analysing risks. Different terms are used for concepts that are mathematically similar (in a causal sense), and the same terms are sometimes used for concepts that are mathematically distinct. For example, RFC 4949 lists five definitions for the term 'access control': two correspond to control strategies, referring to controls such as access policies and enforcement mechanisms related to assets or asset types, two are general control objective statements related only to a broad class of threats, one one is a formal modelling definition that does not refer to any threat. It is easy to

get confused, which can lead to inconsistent or incorrect interpretations and even to inappropriate schema developments. One such development is the story of CVSS, mentioned in Section 2.2.

The terminological inexactitude found in cybersecurity and the failure to reflect the causal nature of cybersecurity threats and associated risks seem to be linked. In Section 2.1 the use of foundational ontologies like BFO and UFO were briefly discussed, noting that they have not been widely exploited to model cybersecurity. It is worth noting that these foundational ontologies support a form of causal modelling. For example, BFO was developed to model geospatial information and is well suited to represent temporal and spatial entities and their relationships. BFO classifies entities as continuants (which persist over time) or occurrents (which occur over time). C3O builds on this, modelling threats as a type of occurrent and adding causal relationships between occurrents. This allows threat paths to be represented, but conceals the causation mechanism. SofIoTS [14] uses C3O as a foundation for a model of IoT security, where the relationship to space is an important aspect that is well supported by BFO. However, SofIoTS finds some difficulties in mapping cybersecurity terminology to BFO concepts. For example, value and risk are associated with possible futures, which should only be associated with types in BFO, and vulnerability is context dependent, which is inconsistent with treating it as a disposition (a characteristic of an endurant bearer that must be intrinsic to the bearer).

UFO classifies entities as endurants (which are wholly present when they exist in time), events (which occur over time), or situations (collections of endurants that may exist in time). An event is then a transition between situations, elements of which can be regarded as causes or effects of the event. These ideas were used to develop ontologies for risks [36] and security [16]. The last of these is interesting because, although not a cybersecurity ontology, it aligns quite well with the causal model described in Section 3. Threats are a type of event that arises in a situation containing certain objects (existentially independent endurants) with certain dispositions (capabilities). However, to remain consistent with UFO risks are described in terms of the intentions of stakeholders, including attackers, and security in terms of intervention events. This is formally correct, but difficult to use in a risk analysis. It is also far removed from the terminology commonly used in cyber security. This is elaborated by Oliveira et al. in an analysis of Mitre's D3FEND ontology [37], which concludes that D3FEND is self-inconsistent and should be improved by reformulation using UFO as a foundation. Another way

to interpret this (and some of the concerns raised by SofIoTS) is that the terminology of cybersecurity is not entirely fit for purpose.

Alignment with (or mapping to) a causal model seems a necessary step if cybersecurity knowledge is to be used in a practical way for risk assessment, but the underlying models from BFO or UFO come with ontological baggage that makes mapping difficult and complicates the picture for cybersecurity practitioners. The causal model from 1 seems like a better target for such a mapping. Its simplicity comes primarily from the representation of system model boundaries using exogenous variables, so fine details and external entities can be omitted without creating inconsistencies. These come with a natural mapping to a subset of terms used in cybersecurity, although these terms must be understood in terms of the underlying simple mathematical model. If existing cybersecurity knowledge bases can be mapped onto such a model, it should become much easier to integrate and use them together.

5.2. Causal misalignment

Some existing cybersecurity knowledge bases or schemas cannot be easily related to a causal model because they use incompatible methods to model causal aspects. A good example is CVSS, which provides two sets of metrics used to classify specific software vulnerabilities:

- exploitability metrics, which describe what is necessary to provoke execution of the vulnerable code, etc.
- impact metrics, which describe the consequences if the bugs in vulnerable code can be exploited.

This seems to align with a causal model. Exploitation of vulnerable code is a threat, with causes described by exploitability metrics and effects described by impact metrics. However, while the CVSS exploitability metrics encode the privileges needed to access vulnerable code, the impact metrics only encode effects on the confidentiality and integrity of the data or the availability of system resources. These effects normally occur at the end of an attack path, whereas the vulnerable code exploit usually occurs at the start. CVSS is self-inconsistent, using a privilege model for causes, but not for effects. There is a gap in the model that the vulnerability assessor must bridge, during which causal dependencies are lost.

In CVSS v2, this is partially addressed by scoring tip #9. This is a convention for using impact metrics to encode whether an exploit allows user

or root access to the vulnerable component host. As discussed in Section 4.2, these privileges are related to baseline threats that play an important role in the formation of threat paths. Scoring tip #9 allows the main privileges gained by an exploit to be deduced, so the exploit can be modelled as a threat whose threat paths can be found from its causal dependencies.

The value of scoring tip #9 appears to have been underappreciated, possibly because most users do not create causal risk models. Since CVSS v3, scoring tip #9 was removed and the impact metrics were revised to include indirect impacts, with a scope metric indicating whether the impact is in the vulnerable component or a dependent component. This addressed criticisms about the need to model indirect effects (see Section 2), but without using a causal model. As a result, some causal relationships become hidden, making impact metrics less usable and less useful. Vulnerability assessors must decide which dependent components are present in some assumed target system, and in effect do their own threat path analysis to determine the impact metrics. This must make it harder to analyse new vulnerabilities and probably contributed to the backlog in classifying vulnerabilities in the NVD [38]. It also makes it even more difficult to know whether the CVSS scores for a given vulnerability are relevant to a given target system.

It is proposed that CVSS could easily be updated by realigning with a causal model. The exploitation of a given vulnerability can be modelled using a depth 2 threat graph. Each path in this graph will contain:

1. a vulnerable code access threat, whose causes correspond to the privileges needed to access vulnerable code, and whose effect is a behaviour denoting that vulnerable code is accessible,
2. a vulnerable code impact threat, caused by the effect of the access threat and having effects that represent the gaining of privileges, or the compromise of confidentiality, integrity or availability.

Generic forms of these threats can be included in a knowledge base to provide a specification and support the interpretation of metrics. The causes of access threats will represent the presence of a vulnerability that is accessible given certain privileges and the possession by attackers of those privileges. Impact threats will have effects that represent loss of confidentiality, integrity, or availability, or gain of privileges needed for further attacks. Expressing the exploit as two threats means that fewer threat models are needed to cover all possible combinations of exploitability and impact metric values.

The exploitability metrics in CVSS align quite well with this approach, but (as discussed) the impact metrics do not. Metrics should be added corresponding to causes of baseline threats such as root or user access to a host, access to networks, etc. Specialised privileges causing new threats can be added as necessary, such as the ability to inject or alter requests or responses sent by the vulnerable component, as happens in query injection or cross-site scripting vulnerabilities.

With this approach, the metrics relate to the two-step threat path, which does not depend on which other components are present in the target system. It would be easier to describe a vulnerability using these metrics based only on an analysis of the vulnerable code. The relevance of a vulnerability to a given target system could be determined automatically by embedding the two-step threat path into a causal model of that system.

A prototype set of access and impact threats with privilege-related causes and effects is included in the current Spyderisk knowledge base. A mapping was also created to the relevant trustworthiness attributes from CVSS v2 by using scoring tip #9 but this is not reliable for CVSS v3 and v4, where the metrics relate to longer threat paths with hidden causal relationships.

5.3. Causal integration

The Unified Ontology for Cyber Security mentioned in Section 2.1 includes both D3FEND and NVD. Integration is achieved through a relationship overlay called WAVED, used to link concepts in different integrated knowledge bases. The most significant links relate offensive techniques (the ATT&CK part of D3FEND) with CVE entries from the NVD. WAVED can be used to find (say) offensive or defensive techniques related to a specific vulnerability, which is clearly useful, but does not directly support risk assessment.

It is proposed that a more pragmatic integration approach would be to combine cybersecurity knowledge from different sources using causal models as a common reference point. Aligning D3FEND with a causal model should be easier than CVSS. D3FEND is a control applicability model, in which some (not all) aspects result from underlying causal relationships. Although these are not encoded explicitly in D3FEND, they are not hidden in a way that contradicts Figure 1. D3FEND describes:

- offensive techniques derived from ATT&CK, analogous to threats,
- defensive techniques, and

- digital artefacts used by offensive or defensive cybersecurity actors.

This is somewhat similar to the causal model ontology in Figure 1: offensive techniques correspond to threats, and defensive techniques to control strategies or control properties. Like the assets in Figure 1, the digital artefacts provide the reference frame to describe these techniques, although in D3FEND they represent a subset of the assets in a target system.

It should be relatively easy to create a relationship overlay, similar to WAVED but linking offensive techniques with causal threat models and defensive techniques to control strategies. If threat models were also related to NVD using an updated or alternative form of CVSS, as described in Section 5.2, then the relationships of D3FEND with NVD would be revealed, in a form that would be more directly usable in an ISO 27005 risk assessment. This would also allow the risk assessment to be related to other aspects covered by D3FEND such as response and recovery analysis.

The authors of this article have not tried to do this because both D3FEND and NVD contain a large number of entries and causal relationships are hidden in NVD by the existing CVSS v3 and v4 annotations so considerable reanalysis would be necessary. It may be possible to automate this using AI techniques, with training and test data created using NVD entries that have CVSS v2 annotations. This would make possible the proposed integration.

6. Conclusions and future work

This paper describes a causal model capable of supporting cybersecurity threat path analysis and risk assessment and shows that such a model can provide a useful basis for capturing, integrating and using knowledge about cybersecurity threats, risks, and related concepts. The main virtues of this approach for modelling knowledge are:

- Grounding: there is a well-defined mathematical basis for classifying concepts that does not depend on a precise use of common language.
- Parsimony: the number of distinct concepts, and the number of facts that must be asserted or mapped is small.
- Generality: concepts like threats and vulnerabilities can be modelled without making assumptions about the nature of the target systems.

- Utility: such models can be used to automate difficult steps in a risk assessment using ISO 27005.

- Flexibility: the approach does not restrict how related concepts such as likelihood and risk are represented and determined.

The Spyderisk project has published a minimal supporting ontology, along with a qualitative risk estimation tool and a cybersecurity knowledge base designed around the same causal modelling concepts. The development of this knowledge base provides insight into how the causal structure can be used to capture cybersecurity concepts and iteratively extend the knowledge base. It also sheds light on what is good about several existing cybersecurity taxonomies and knowledge bases, what works less well, and how improvements can be made while avoiding the complexity of an abstract foundational ontology. One such insight is that the development of the CVSS schema since 2017 may have caused problems for curators and users of the NVD, which can be alleviated by realigning CVSS with a causal model.

There are several possible directions for further work, some of which are now being investigated by the authors and their collaborators:

- Standardisation: the current Spyderisk code uses its own bespoke reasoner and knowledge base syntax. Mapping the knowledge base to a common foundational ontology using a standard syntax would enable integration with other tools and knowledge bases.

- Automation: the parsimonious nature of the causal model makes it more feasible to capture knowledge in a more automated manner using AI methods. One target could be the NVD catalogue, to provide a vulnerability classification schema that allows better integration with resources such as D3FEND.

- Integration: the causal model can be used to integrate other cybersecurity knowledge bases, making them more usable in risk assessment. The existing D3FEND and NVD catalogues could be integrated using this approach, given a causal schema to classify vulnerabilities.

The ultimate goal is to exploit cybersecurity knowledge of all kinds in the analysis of potential risks, using automation (including AI where appropriate), to make risk assessment easier and more affordable.

Acknowledgement

This work is within the Horizon Europe TELEMETRY (Trustworthy mEthodologies, open knowledge & autoMated tools for sEcurity Testing of IoT software, haRdware & ecosYstems) project, supported by EC funding under grant number 101119747, and UKRI under grant number 10087006.

References

[1] International Organization for Standardization, ISO/IEC 27005:2022: Information security, cybersecurity and privacy protection — Guidance on managing information security risks (2022).

[2] National Institute of Standards, Technology, National Vulnerability Database, `https://nvd.nist.gov/`, accessed: 4 Nov 2024 (2019).

[3] The MITRE Corporation, ATT&CK® Adversarial Tactics, Techniques, and Common Knowledge, `https://attack.mitre.org/`, accessed: 7 Aug 2024 (2013).

[4] The MITRE Corporation, D3DEND® Detection, Denial, and Disruption Framework Empowering Network Defense, `https://d3fend.mitre.org/`, accessed: 13 Nov 2024 (2021).

[5] Spyderisk: Risk assessment and risk modeling tool, `https://github.com/Spyderisk`, accessed: 4 Nov 2024 (2023).

[6] I. N. W. Group, Internet Security Glossary, Version 2 (2007).

[7] International Organization for Standardization, ISO/IEC 27000:2020: Information technology — Security techniques — Information security management systems — Overview and vocabulary (2020).

[8] A. Herzog, N. Shahmehri, C. Duma, An ontology of information security, International Journal of Information Security and Privacy (IJISP) 1 (4) (2007) 1–23.

[9] S. Fenz, A. Ekelhart, Formalizing information security knowledge, in: Proceedings of the 4th international Symposium on information, Computer, and Communications Security, 2009, pp. 183–194.

[10] C. Blanco, J. Lasheras, E. Fernández-Medina, R. Valencia-García, A. Toval, Basis for an integrated security ontology according to a systematic review of existing proposals, Computer Standards & Interfaces 33 (4) (2011) 372–388.

[11] I. Meriah, L. B. A. Rabai, Comparative study of ontologies based iso 27000 series security standards, Procedia Computer Science 160 (2019) 85–92.

[12] R. Arp, B. Smith, A. D. Spear, Building ontologies with Basic Formal Ontology, Mit Press, 2015.

[13] B. Donohue, M. Jensen, A. P. Cox, R. Rudnicki, A common core-based cyber ontology in support of cross-domain situational awareness, in: Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX, Vol. 10635, SPIE, 2018, pp. 65–74.

[14] P. Smart, M. Boniface, M. A. Jarwar, J. Watson, Sofiots: Ontological framework, demonstration outcomes, and recommendations for further work, https://eprints.soton.ac.uk/490073/1/SOfIoTS_D4_Reportv2.pdf, accessed: 29 Jan 2025 (2023).

[15] G. Guizzardi, A. Botti Benevides, C. M. Fonseca, D. Porello, J. P. A. Almeida, T. Prince Sales, UFO: Unified foundational ontology, Applied ontology 17 (1) (2022) 167–210.

[16] Í. Oliveira, T. P. Sales, R. Baratella, M. Fumagalli, G. Guizzardi, An ontology of security from a risk treatment perspective, in: International conference on conceptual modeling, Springer, 2022, pp. 365–379.

[17] P. E. Kaloroumakis, M. J. Smith, Toward a knowledge graph of cybersecurity countermeasures, The MITRE Corporation 11 (2021) 2021.

[18] Z. Syed, A. Padia, M. L. Mathews, T. Finin, A. Joshi, et al., Uco: A unified cybersecurity ontology, in: Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security, 2016, pp. 195–202.

[19] K. A. Akbar, F. I. Rahman, A. Singhal, L. Khan, B. Thuraisingham, The design and application of a unified ontology for cyber security, in: International Conference on Information Systems Security, Springer, 2023, pp. 23–41.

[20] C. Phillips, L. P. Swiler, A graph-based system for network-vulnerability analysis, in: Proceedings of the 1998 workshop on New security paradigms, 1998, pp. 71–79.

[21] B. Schneier, Attack trees, Dr. Dobb's journal 24 (12) (1999) 21–29, https://www.schneier.com/academic/archives/1999/12/attack_trees.html.

[22] Council of European Union, Council directive 2008/114/ec on the identification and designation of European critical infrastructures and the assessment of the need to improve their protection, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32008L0114 (2008).

[23] C. Guariniello, D. DeLaurentis, Communications, information, and cyber security in systems-of-systems: Assessing the impact of attacks through interdependency analysis, Procedia Computer Science 28 (2014) 720–727.

[24] N. Poolsappasit, R. Dewri, I. Ray, Dynamic security risk management using Bayesian attack graphs, IEEE Transactions on Dependable and Secure Computing 9 (1) (2011) 61–74.

[25] O. Cyber Threat Intelligence (CTI Technical Committee), Structured Threat Information Expression (STIX™), https://oasis-open.github.io/cti-documentation/stix/intro, accessed: 13 Nov 2024 (2020).

[26] P. Mell, K. Scarfone, S. Romanosky, CVSS: A Complete Guide to the Common Vulnerability Scoring System, version 2.0, https://www.first.org/cvss/v2/guide, accessed: 4 Nov 2024 (2007).

[27] B. Schoenfield, D. Quiroga, Don't Substitute CVSS for Risk: Scoring System Inflates Importance of CVE-2017-3735, https://www.mcafee.com/blogs/other-blogs/mcafee-labs/dont-substitute-cvss-for-risk-scoring-system-inflates-importance-of-cve-2017-3735, accessed: 5 Nov 2024 (2017).

[28] Forum of Incident Response, I. Security Teams, CVSS: Common Vulnerability Scoring System, version 3.0: Specification Document,

https://www.first.org/cvss/v3.0/specification-document, accessed: 4 Nov 2024 (2015).

[29] M. Surridge, B. Nasser, X. Chen, A. Chakravarthy, P. Melas, Run-time risk management in adaptive ICT systems, in: 2013 International Conference on Availability, Reliability and Security, IEEE, 2013, pp. 102–110.

[30] S. C. Phillips, S. Taylor, M. Boniface, S. Modafferi, M. Surridge, Automated knowledge-based cybersecurity risk assessment of cyber-physical systems, IEEE Access (2024).

[31] T. D. Cook, D. T. Campbell, Experimental and quasi-experimental designs for generalized causal inference (1986).

[32] G. W. Imbens, D. B. Rubin, Causal inference in statistics, social, and biomedical sciences, Cambridge university press, 2015.

[33] J. Pearl, Causality, Cambridge University Press, 2009.

[34] Spyderisk Core Model, https://github.com/Spyderisk/ontopublish, accessed: 29 Mar 2025 (2025).

[35] Spyderisk: IT Network Model, https://github.com/Spyderisk/domain-network, accessed: 4 June 2025 (2025).

[36] T. P. Sales, F. Baião, G. Guizzardi, J. P. A. Almeida, N. Guarino, J. Mylopoulos, The common ontology of value and risk, in: International conference on conceptual modeling, Springer, 2018, pp. 121–135.

[37] Í. Oliveira, G. Engelberg, P. P. F. Barcelos, T. P. Sales, M. Fumagalli, R. Baratella, D. Klein, G. Guizzardi, Boosting D3FEND: Ontological analysis and recommendations, in: Formal Ontology in Information Systems, IOS Press, 2023, pp. 334–348.

[38] K. Townsend, CVE and NVD - A Weak and Fractured Source of Vulnerability Truth, https://www.securityweek.com/cve-and-nvd-a-weak-and-fractured-source-of-vulnerability-truth/, accessed: 14 Feb 2025 (2024).