Computer Science Review 35 (2020) 100219

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

A review of attack graph and attack tree visual syntax in cyber security

Harjinder Singh Lallie [a,*], Kurt Debattista [b], Jay Bal [b]

[a] Cyber Security Centre, WMG, University of Warwick, Coventry, UK
[b] WMG, University of Warwick, Coventry, UK

ARTICLE INFO

Article history:
Received 26 March 2019
Received in revised form 30 September 2019
Accepted 12 December 2019
Available online 11 January 2020

Keywords:
Attack graph
Attack tree
Visual syntax
Cyber-attack

ABSTRACT

Perceiving and understanding cyber-attacks can be a difficult task, and more effective techniques are needed to aid cyber-attack perception. Attack modelling techniques (AMTs) - such as attack graphs, attack trees and fault trees, are a popular method of mathematically and visually representing the sequence of events that lead to a successful cyber-attack. These methods are useful visual aids that can aid cyber-attack perception.

This survey paper describes the fundamental theory of cyber-attack before describing how important elements of a cyber-attack are represented in attack graphs and attack trees. The key focus of the paper is to present empirical research aimed at analysing more than 180 attack graphs and attack trees to identify how attack graphs and attack trees present cyber-attacks in terms of their visual syntax.

There is little empirical or comparative research which evaluates the effectiveness of these methods. Furthermore, despite their popularity, there is no standardised attack graph visual syntax configuration, and more than seventy self-nominated attack graph and twenty attack tree configurations have been described in the literature — each of which presents attributes such as preconditions and exploits in a different way. The survey demonstrates that there is no standard method of representing attack graphs or attack trees and that more research is needed to standardise the representation.

Contents

* Corresponding author.
E-mail addresses: HL@warwick.ac.uk (H.S. Lallie), K.Debattista@warwick.ac.uk (K. Debattista), jay.bal@warwick.ac.uk (J. Bal).

https://doi.org/10.1016/j.cosrev.2019.100219
1574-0137/

1. Introduction

Cyber security pervades many aspects of social, political and business life and has huge implications for the online and personal safety of individuals and families. In 2017, the average cost of a data breach was reported globally as $3.86 million [1]. Although many data breaches result in the compromise of personal data, a number of well publicised attacks against vehicular [2], medical [3], and industrial control systems [4] have demonstrated that cyber security breaches can have serious impacts on personal safety.

Quite often mitigation strategies focus on protecting systems from perpetrators intent on causing deliberate damage to a system and/or data, however, many cyber security failures occur simply as a result of user behaviour — often because of unintentional errors brought about by a failure to fully understand security mechanisms [5].

User error and behaviour is an important factor in a cyber-attack [6,7]. Perhaps one of the most important mechanisms in improving user behaviour is to make it easier to perceive cyber security and make it more 'useable'. Research into cyber security usability aims to design systems which help to understand how users perceive and understand cyber security [8–10] by taking *"into account the perceptions, characteristics, needs, abilities and behaviours of users"* [11].

Cyber-attack perception is an important research problem [12–14] which demands better techniques and methods to aid the perception and assessment of cyber-attacks. Quite often, observers find the analysis and understanding of complex patterns difficult to visualise [15,16]. Well-designed diagrams and graphical systems can aid this process [17,18].

This paper describes how attack graphs and attack trees represent cyber-attacks in terms of their visual syntax. The paper demonstrates that although there are numerous benefits to presenting cyber-attacks as attack graphs or attack trees, there are inconsistencies regarding the way cyber-attacks are represented in attack graphs and attack trees and in doing so, outlines the need to standardise their visual syntax. This is the first paper to present such a detailed critical analysis of the visual syntax of attack modelling techniques.

Collectively these shortcomings outline the need for further research and they also define some of the parameters that comparative evaluations should consider. The research presented in Lallie et al. [19,20] aims to address these shortcomings and propose an attack graph visual syntax which is cognitively effective and also preferred by practitioners — thereby increasing the likelihood of adoption.

The rest of this paper is structured as follows. Section 2 begins by describing the concept of AMTs and the use of visual syntax to represent cyber-attacks. Section 3 proceeds to describe the underlying theory of cyber-attacks. This section describes the relationship between the primary cyber-attack constructs which are: *precondition, exploit* and *precondition logic*, and also explains the secondary constructs which are variants of primary constructs. Collectively, the primary and secondary constructs correspond to the concepts that AMTs should represent.

Section 4 outlines some of the graph theory relating to attack graphs and attack trees and then Section 5 introduces important visual syntax theoretical concepts. Section 6 provides a detailed analysis of the presentation of attack graphs and attack trees.

2. Attack modelling techniques (AMTs)

Attack modelling techniques (AMT) are used to model and visualise the sequence and/or combination of events that enable a successful cyber-attack on a computer or network. AMTs can be broadly divided into three categories: methods that are based on the *use case* framework, methods that present a cyber-attack from a *temporal* perspective, and *graph* based methods. These methods are highlighted in Fig. 2. Of the methods outlined in Fig. 2, attack graphs and attack trees are the most popular method of representing cyber-attacks — at least as far as the academic literature is concerned. These two methods form the basis of the present research.

AMTs enable observers to evaluate the salient information in a diagram [16,21–23] and help remove the intellectual burden from security experts — who have to evaluate cyber-attack scenarios

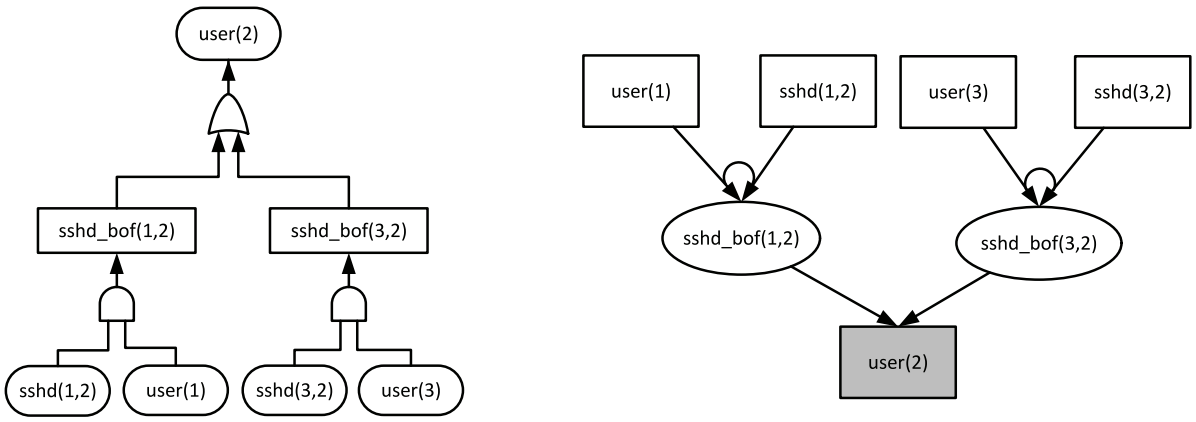

Fig. 1. Two attack models. A fault tree (left) and an attack graph (right).

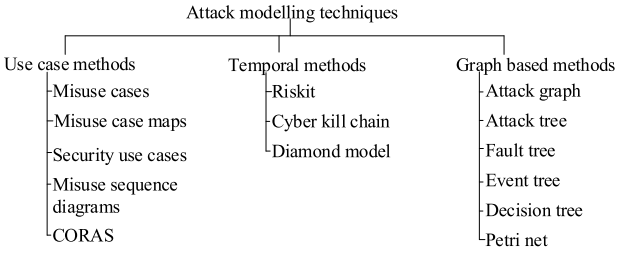

Fig. 2. Attack modelling techniques.

and evaluate potential mitigations [24]. Consequently, security problems can be presented in a manner that enables a decision maker — whether an expert or non-expert, to more quickly grasp the problem [22], better perceive risk landscapes [25], and easily perceive complex concepts [26]. In such circumstances, AMTs provide effective tools and workspaces [27], they make this process clearer and simpler and thereby facilitate easier discussion and debate [23], and can aid the perception of cyber-attacks with little reference to logical models [28].

An example of two AMTs - a fault tree and attack graph, is provided in Fig. 1. The example in Fig. 1 demonstrates how a perpetrator is able to conduct a series of exploits (`sshd_bof`) on a sequence of host computing devices (denoted in brackets), and by doing so, acquire user privileges (`user`) on each one. The example also shows one of the preconditions (`sshd`) that are necessary for the attack to be successful. This example demonstrates how a sequence of exploits can be visualised to aid cyber-attack perception.

The cyber security domain suffers from two specific problems relevant to the present paper: inconsistency in the ontological terms, vocabulary, and definitions used to describe the domain [29], and a problem of unclear and meaningless semantics [30].

Multiple terms are used to mean the same thing. For example, *exploits* are also referred to as *actions* or *attacks*, the *perpetrators* are also referred to as *attackers*, *malefactors*, *actors* and *adversaries*, the *preconditions* to an exploit are also referred to as *requires*, *predicates* or *prerequisites*, and the *postconditions* of the exploit are also referred to as *provides* or *effects*. Further examples are provided in Table 1. This problem of inconsistency extends to multiple areas in cyber security, and is particularly acute for attack graphs and attack trees.

This paper uses the terms *exploits*, *preconditions* and *perpetrators*. While the use of precondition and exploit may appear straightforward, the selection of the term *perpetrator* requires more justification. Terms such as *attacker*, *adversary*, *crook*, *bad guy*, *assailant* and *hacker* suggest that the source of a cyber-attack is acting deliberately, when in fact the exploit might be effected inadvertently. Terms such as *mis-user* and *mis-actor* are very uncommon and unlikely to resonate easily with readers. The term *perpetrator* does not suggest either a deliberated or inadvertent motive. The use of this term does not remove anything in terms of the descriptive power of the term.

Table 1
Terms used to describe the fundamental cyber-attack constructs.

Term	Term used & supporting references
Precondition	*Prerequisite*: [31–37]; *Precondition*: [38–47]; *Prerequisite and precondition*: [46,48–51]; *Predicate*; [29,32,52–56]; *Requires*: [57–59]
Postcondition	*Postcondition*: [39,41,43,45,60]; *Effect*: [61]; *Provides*: [57–59]; *Consequence*: [31,32,50,62,63]; *Impact*: [28,59,64–67];
Perpetrator	*Perpetrator*: [68–70]; *Attacker*: [34,35,71–74]; *Adversary*: [42,75–77]; *Malefactor*: [60,78,79]; *Actor*: [29,80–82]; *Hacker*: [47,48,68,77,81,83–108]; *Crook*: [49] [82] [109]; *Mis-actor*: [49][2]; *Assailant*: [106,110–113]; *Misuser*: [49,94,114–117]; *Bad guy*: [113,118–122]

[1] Wu et al. use the terms *attacker*, *adversary*, *malefactor* and *perpetrator* interchangeably in the same paper [123].
[2] Sindre and Opdahl use the terms *crook* and *mis-actor* in the same paper to refer to a perpetrator. The term *mis-actor* is used specifically as an *'inverted actor'* in other words the opposite to *actor* which is the term generically used in use cases to refer to the user of a service or function [49].

Attack models are constructed using a combination of two-dimensional shapes — such as squares, rectangles, ellipses; one-dimensional shapes — such as lines; and textual syntax to represent cyber-attack constructs. This is referred to as a visual syntax [124], visual rhetoric [125] or visual grammar [126]. The visual syntax of modelling systems such as fault trees [127] and Petri nets [128] - both also used occasionally to model cyber-attacks, is standardised — resulting in a common understanding of the framework. This is not the case for attack graphs and attack trees.

Attack graphs and attack trees suffer from a distinct lack of standards, prescriptive methodologies and common approaches in terms of their visual syntax [129,130]. Authors use self-nominated and untested visual syntax to model the attack — referred to by Alexander [131] as the *unselfconscious design approach* [131].

There are more than seventy five attack graph visual syntax configurations and more than twenty attack tree configurations — as demonstrated in Tables 22 and 23 which describe the visual

Table 2
Cyber-attack terms.

Term	Explanation
Vulnerability	An exploitable unplanned system weakness which exists because of the existence of one or more preconditions
Precondition	A system state that is necessary for an exploit to be successful
Initial precondition	The first precondition in a cyber-attack
Perpetrator capability	The tools, knowledge and/or access/privilege levels that a perpetrator needs to be able to run an exploit
Postcondition	The conditions/states created by a successful exploit
Goal	The ultimate target of a cyber-attack
Exploit	A set of steps — executed as code or manual steps, which take advantage of one or more vulnerabilities in a target system and provide specific capabilities to the perpetrator
Non-intrusive event	An event which aids and supports a cyber-attack but does not alter the state of the target system

syntax of more than 180 attack graph and attack tree visual syntaxes. The visual syntax in these examples differ in terms of the shapes used to represent constructs such as preconditions and exploits. The availability of numerous opposing proposals can give rise to confusion for researchers and practitioners in deciding which to use and is evidence of an *"immature research field"* resulting in a *"fragmentation of research efforts"* [132].

3. Cyber-attacks

A number of recent high-profile attacks have exemplified the need to better understand cyber-attacks. These include the Stuxnet virus [4,95,133] the Jeep Cherokee hack [2], the heartbleed attack [104,134,135], the Sony hack [106,136–138] and the Ukrainian power grid attack [108,139].

The fundamental components of all these cyber-attacks were the same and comprised of elements such as *exploits, vulnerabilities*, and *postconditions*.

The terms described in this paper are summarised in Table 2 and the rest of this section describes these terms in further detail.

3.1. Vulnerability, precondition and postcondition

3.1.1. Vulnerability

A *vulnerability* is an exploitable weakness in the design, implementation or management of a system [140]. A vulnerability comprises of a combination of one or more system states referred to as *preconditions*.

3.1.2. Precondition

Preconditions are a set of system properties that must exist for an exploit to be successful. An *initial precondition* is a system property which exists inherently in a system and which did not arise as a consequence of exploitation [40]. Addressing these could make all further steps in an attack null and void.

There are at least three types of precondition:

1. *Statuses/services*. The target holds or advertises particular versions of operating systems, systems software/applications, services [141,142], or is in a particular hardware/software state.
2. *Reachability*. The target is reachable.
3. *Perpetrator capability*. The perpetrator has particular capabilities such as the ability to run a process on a target, access to tools, or privilege levels [143] and/or is in possession of the tools to conduct an attack and has the necessary skill level [141].

Of the precondition types discussed above, *statuses/services* are commonly represented in AMTs such as attack graphs and attack trees, however, *reachability* and *perpetrator capability* are less commonly represented [29,141,144].

Perpetrator capability is an important element in the analysis of an attack and it can be important for a security analyst to understand exploits that require greater perpetrator capability versus those that require less [145]. As an example of perpetrator capability, consider the example of the *rcp* (*remote call procedure*) exploit given by [40]. The *rcp* exploit requires the following capabilities:

- The *rcp* service must be available to the perpetrator.
- The target host must trust the source host.
- The perpetrator must have local shell access.

3.1.3. Postcondition

The successful perpetration of an exploit results in one or more *postconditions*. Although the result of an exploit is technically referred to as a postcondition, these can also form the preconditions of further exploits, therefore, most researchers refer to postconditions simply as precondition with the term *goal* being used to identify the final postcondition.

3.2. Precondition logic

For an exploit to be successful, one or more preconditions must be satisfied. The combination of preconditions can be represented using precondition logic. So, given two preconditions: pr_1 and pr_2, if both the preconditions have to be satisfied, this can be represented as a conjunction of the form: $pr_1 \wedge pr_2$ - where the symbol $\wedge$ represents conjunction. Conversely, if any one of the two must be satisfied, then this can be represented as a disjunction of the form: $pr_1 \vee pr_2$ - where the symbol $\vee$ represents disjunction.

The shape or symbol used to represent precondition logic in an AMT is referred to as the *precondition operator*. In the examples given above, the precondition operators are $\wedge$ and $\vee$.

Although the representation of precondition logic is critical in helping to identify cyber-attack mitigations and countermeasures, very few attack graphs represent precondition logic [65,88,88,146–149].

To illustrate the benefit of presenting precondition logic, consider the attack graph representation in Fig. 3 [150]. In this example, `sshd(3,1)` and `user(3)` are two preconditions. In the formal expressions presented by Barik and Mazumdar, `sshd(3,1)` means that the `sshd` (*secure shell daemon/service*) is running on host `1` and is accessible by host 3, and `user(3)` means that the perpetrator has user access privileges on host 3 which is the host from which the exploit will be launched.

This attack graph does not represent the precondition logic necessary for each exploit, consequently, the attack graph could be interpreted in a number of ways. For example, for a successful `sshd buffer overflow` attack (`sshd_bof(3,1)`), one of the following two preconditions might apply:

1. `sshd(3,1)` $\wedge$ `user(3)`
2. `sshd(3,1)` $\vee$ `user(3)`

As it happens, both preconditions (`sshd(3,1)` $\wedge$ `user(3)`) must be satisfied for the exploit to be successful.

The representation of precondition logic in attack graphs is highlighted in Table 22, and the data shows that there are numerous attack graphs which do not represent precondition logic.

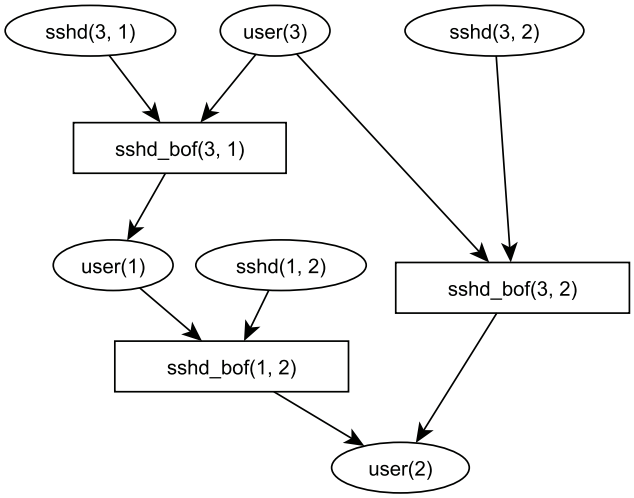

Fig. 3. An example attack graph.
Source: Adapted from [144].

Table 3
IDS signatures, preconditions & postconditions [59].

Signature (Exploit)	Precondition	Postcondition
`RPC sadmind UDP PING`	Disclosure of host data	Disclosure of running service; system access (non-intrusive)
`RPC portmap sadmind request UDP`	Disclosure of host data	Disclosure of port number; disclosure of running service; system access; remote access (non-intrusive)
`RPC sadmind UDP NET-MGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt`	Disclosure of host; disclosure of port number; disclosure of running service	System access; remote access; admin access (intrusive)
`RPC sadmind query with root credentials attempt UDP`	Disclosure of host; disclosure of port number; disclosure of running service; system access; remote access	Remote access; admin access (intrusive)

3.3. Exploits

An exploit is a set of steps — executed as code or manual steps, which take advantage of one or more vulnerabilities in a target system and provide specific capabilities to the perpetrator [29].

Exploits can be defined as a formal expression in the form:

`exploit(vulnerability, postcondition, source, target)`

where:

- `vulnerability` is the vulnerability being exploited. As outlined earlier, the vulnerability can be broken down into a number of *preconditions*, all or a combination of which must exist for the vulnerability to be exploited.
- `postcondition` is the outcome of the exploit.
- `source` is the source host i.e., the host that commits the exploit.
- `target` is the target of the exploit.

Examples of the formal expression of exploits include:

- `ftp_rhosts(ftp, trustedhost,i,j)`: where `i` exploits an `ftp` vulnerability in `j`, and uploads a list of trusted hosts to host `j`.
- `sshd_bof(sshd, root,i,j)`: where `i` exploits a buffer overflow vulnerability in the `ssh` (secure shell) service of `j`. This provides `i` with root access to `j`.
- `local_bof(bofv, root,i,j)`: where `i` exploits a buffer overflow vulnerability (`bofv`) in `j`. This provides `i` with root access to `j`.

It is noteworthy that the formal expression presented above is proposed by the authors, and although numerous proposals have been put forward in the literature [34,151–153], there are no standards or agreements in terms of the formal expressions used to define exploits and vulnerabilities.

3.3.1. Non-intrusive events

Non-intrusive events — also referred to as *secondary exploits* [60] and *suspicious* events [80], are system discovery operations used by perpetrators to discover target system properties and vulnerabilities [72,80]. Unlike exploits — which alter system properties, non-intrusive events do not alter the system properties — unless there is a programmed/manual reaction to the event such as system port blocking, or the creation of new intrusion detection system rules.

Examples of non-intrusive events include `rpcinfo` - which reveals port and service data on the target machine, and probes and scans such as `nmap, netcat, amap, XProbe, P0f` and `X-Scan` - which return a system response but do not alter the system state.

Intrusion detection systems (IDSs) detect and alert both intrusive (exploits) and non-intrusive events. Table 3 shows that some events — such as the `RPC sadmind UDP PING` and the `RPC portmap sadmind request UDP`, are both intrusive and non-intrusive. In other words, `RPC sadmind UDP PING` reveals some information about the host — this is non-intrusive, and `RPC portmap sadmind request UDP` provides system access to the host — which is intrusive. Similarly, an *IP address spoof* results in a non-intrusive outcome.

This paper refers to intrusive and non-intrusive events simply as *exploits*.

Exploits, preconditions and precondition logic are referred to in this research as *primary constructs* because they are the fundamental cyber-attack constructs. Initial preconditions, goals, postconditions and perpetrator capability are referred to as *secondary constructs* because they are variants of primary constructs.

3.4. Cyber-attack

Having outlined the basic cyber-attack constructs, the discussion can now proceed to defining the term itself. A *cyber-attack* comprises of a set of one or more *exploits* which are applied as sequences or in parallel against a target system or systems and which expose *vulnerabilities* in, and alter the state of the targeted system(s) [59].

Once the vulnerability is exploited, a number of *postconditions* are created — each of which could be a precondition for further exploits [29]. The ultimate purpose/aim of an attack is referred to as a *goal* in an attack tree [180] and attack graph, and an *undesirable condition* in a fault tree [181].

It is useful to note that observers are interested in different views of a cyber-attack. This is referred to by Moody [124] as the *principle of cognitive fit* (discussed in Section 5). Analysts are interested in a low level view technical of a cyber-attack and particularly in the preconditions and exploits that contributed to the attack. Occasionally, they have to communicate the cyber-attack to non-experts. Non-experts such as stakeholders, executives and decision makers are interested in a high-level — often non-technical view of the cyber-attack which hides

Table 4
Application of attack graphs in system hardening.

Prediction strategy		Method & citations
Event prediction	*Measuring event likelihood*	*Markov decision process* to calculate the probability of attack success [52], *hyper-alert correlation graph* and the *attack strategy graph* [32,57], *compromised confidence index* [149], *adjacency matrix clustering* [154], *alert correlation matrix* [155], *association rule mining* [50], *rankFutureScenarios* [64], *probability based attack prediction* [35,37], *absorbing Markov chain* for performing exploitability analysis and a *Markov reward model* for aiding likelihood of compromise, [156], *forecasting attack graph* [157], *pwnPr3d* [158], Other contributions include: [32,35,37,50,68,75,83,159–161]
	Analysing multi-Step attacks	*Alert clustering*, *alert merging* and *intent recognition* [80], *alert correlation framework* [31], *causal correlation* [33], *clustered adjacency matrices* [154], *divide and conquer framework* [153], *divisive hierarchical clustering algorithm* [162], *exploit dependency graph* [97], other contributions include: [24,50,151,159,163–167]
	Attack path analysis	*Attack chaining* [151], calculating the difficulty of perpetrating an attack [88], *association rule mining* [50], other contributions include: [145], [159], [47],
Miscellaneous	*Strengthening security weaknesses*	[52,68,144,168] *exploitation graphs* [88], *topological vulnerability analysis* [169]
	Prioritising vulnerability removal	*Skill level analysis* [145], *Minimum Critical Set of Attacks (MCSA)* [52], *predictive graph* [34,61], *network hardening graph* [62], [170], a *game-theoretic approach* [171]
	IDS alert correlation	*Hyper-alert correlation graph* [32], *ADEPTS* [149], *association rule mining algorithm* [50], *topological vulnerability analysis (TVA)* [167], Other contributions include: [31,34,41,59,64,75,152,157,172]
Determining optimal device configuration	*Identifying critical devices*	*AssetRank algorithm* [173], Other contributions include: [156,174,175]
	Determining optimal device placement	*Topological vulnerability analysis* [169], *success measurement model* [176]
	Optimal device configuration	*Graph based network vulnerability analysis* [141], *clustered adjacency matrices* [154], *ranked attack graph* using *PageRank* [177], *dependency attack graphs* [173], *optimal IDS placement* [169,178], *NetSPA* [34], *attack response graph* [179]
	Predicting the impact of configuration settings	*Reverse graph traversal* for identifying critical preconditions and vulnerabilities [62], *ranked attack graph* [177], *adjacency matrix clustering method* and *reachability analysis* [154], *predictive graph* [34,61,169], *divisive hierarchical clustering algorithm* [162]

Table 5
Attack modelling techniques — miscellaneous methods.

Method	Citations
Use case based AMTs	*Misuse cases* [49,82,117,183–187]; *misuse case maps* [94,188]; *security use cases* [189]; CORAS [89,190–193]
Temporal methods	*Diamond model* [194,195]; *cyber kill chain* [105,196–198]; *Riskit* [199–201]
Miscellaneous graph/tree based AMTs[1]	*Event tree* [202]; *OCTAVE* [203]; *the bowtie method* [70,204–207]; *influence diagrams* [44,208–211]; *extended influence diagram* [212,213]

[1]As the title suggests, these are general miscellaneous attack graph/tree based AMTs. Attack graph based examples are given in Table 22, and attack tree based examples are provided in Table 23.

Fig. 4. Examples of vertices and edges. *Source:* Adapted from [45, left], [144, right].

the constituent elements of the attack. However, occasionally it is useful for them to see the cyber-attack from the analyst's viewpoint [182].

3.5. Attack paths

Quite often, there are alternative sequences of exploits — any one of which can result in a successful attack. These are referred to as *attack paths* and are represented in all AMTs. An example of this is provided in the attack graph highlighted in Fig. 6 which comprises of two attack paths which can be described as:

1. `sshd(3,1)` ∧ `user(3)` → `sshd_bof(3,1)`; `user(1)` ∧ `sshd(1,2)` → `sshd_bof(1,2)` → `user(2)` (highlighted in a red/dotted line).
2. `user(3)` ∧ `sshd(3,2)` → `sshd_bof(3,2)` → `user(2)` (highlighted in a green/dashed line).

Techniques such as *attack path analysis* aid the investigation of exploit paths in an attack graph by outlining known and predicted attack event sequences. Such an analysis helps to identify the resources in the path that could be affected, as well as the vulnerabilities that lie along those paths. Examples of the use of attack path analysis combined with weighted edges in aiding the attack prediction include the *Bayesian based attack graph* proposed by Frigault and Wang [46]. The attack graph in this example enables an analyst to calculate the probability of a successful attack path. Each node in the graph represents a potential vulnerability and the preconditions and postconditions associated with the vulnerability. A probability is assigned to each node to represent the likelihood of that vulnerability being exposed. Further examples are provided in Table 4.

Fig. 6 highlights another of the differences in the visual representation of attack graphs and attack trees — that of the event flow. The figure shows that the flow of events is represented top-down. In other words, an observer starts reading at the top

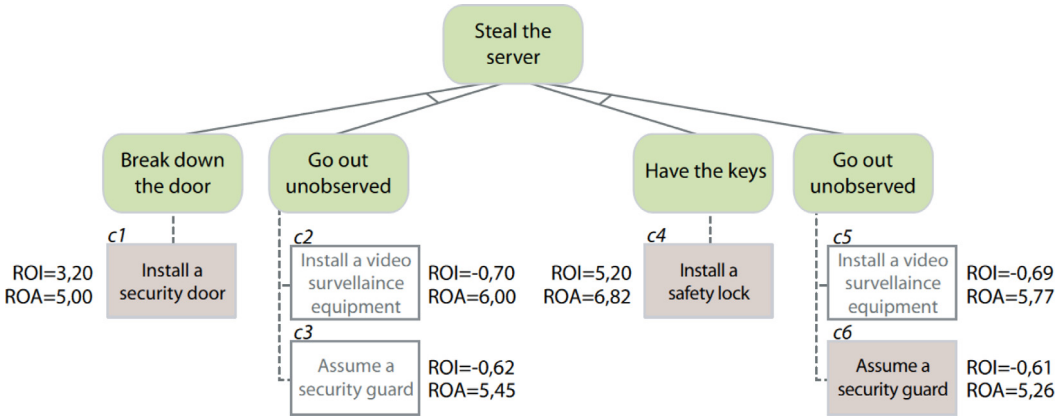

Fig. 5. A weighted attack tree in which weights are attached to vertices.
Source: Reproduced with permission from [214].

and follows the sequence of events downwards. However in Fig. 1 (left), the event flow is represented bottom-up. The data described in Tables 22 and 23, shows that 102 of the 118 attack graph configurations surveyed (86.4%) presented events as top-down, and 59 of the 61 attack trees surveyed (96.7%) presented these as bottom-up.

Fig. 6 introduces another key difference between the visual representation of attack graphs and attack trees. Although this attack graph represents a single attack goal (`user 2`) with two paths leading to it, attack graphs are frequently used to represent complex attacks which have multiple paths and goals. Aguessy notes that attack trees are limited in that they only represent a single attack, whereas an attack graph can represent multiple attacks [66].

Aguessy is referring to what are known as *full attack graphs* and *partial attack graphs*. A full attack graph outlines all potential vulnerabilities and all possible attack paths in a given network [28,40,45,215,216]. A partial attack graph — also referred to as a *minimal attack graph* [56], outlines the pattern of interactions between nodes for a given attack [41,48,141,168,216]. The ability to represent a full attack graph is important in scenarios such as the Stuxnet and Jeep Cherokee attacks and also in analysing network problems.

Generally, attack trees represent singular attacks, and the *attack forest* - a collection of attack trees [81,217], was an attempt to address this problem.

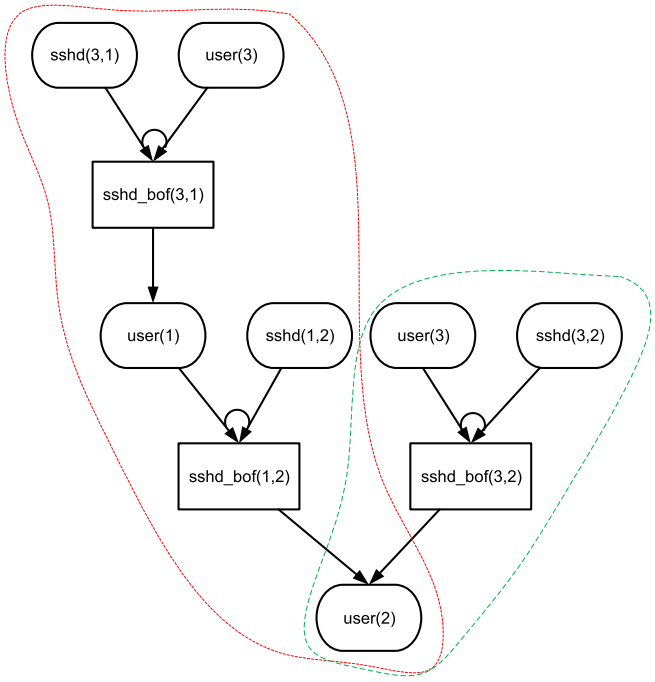

Fig. 6. An example attack graph.
Source: Adapted from [144].

4. Representing attack graphs and attack trees

Having outlined the broad principles relating to cyber-attacks, it is useful to highlight how cyber-attacks are represented in a visual form using AMTs. Numerous AMTs have been proposed in the academic literature. These can be broadly divided into three categories: *use case, temporal* and *graph* based methods. A number of these methods are highlighted in Fig. 2 and in Table 5. This section describes two graph based AMTs — attack graphs and attack trees.

Attack graphs, attack trees and their variants — which include: OCTAVE, event trees and decision trees, are graph based representations of a cyber-attack. Of these, attack graphs and attack trees are the most popular in the research literature.

The research shows a clear divide between papers that focus on attack graphs and those that focus on attack trees. In other words, authors either write about attack graphs or attack trees — with rarely an acknowledgement of the existence of the other. The present research argues that both attack graphs and attack trees are a graph based representation of a cyber-attack. However, it is important to differentiate between the graph theory based representation of graphs and trees, and the visual representation as found in the cyber-attack literature — and referred to in this paper.

In graph theory, a *tree* is an acyclic graph in which any two vertices are connected by exactly one path, and a *graph* is a cyclic graph [218]. Generally, the visual representation of decision trees, event trees, attack trees, fault trees and Petri nets is consistent with the definition of trees because they are visually represented as acyclic graphs.

Attack graphs have been visually represented as both cyclic [30,45,46,55,150,219–221] and acyclic graphs [34,157,166,222–226] - with the suffix *graph* being used by authors in both cases. In other words, the visual representation of attack graphs in the research considered in this paper, does not always conform to the graph theory definition of a graph. Two visual examples of cyclic and acyclic attack graphs are provided in Section 4.1.2 (Fig. 10). Although these attack graphs could more accurately be referred to as trees rather than graphs, this apparent mathematical anomaly

is not considered to be a problem in this paper because the present research focuses on the visual representation of attack graphs and uses the terms proposed by the authors in their papers — regardless of whether they are seen to strictly conform to the definitions outlined above.

In addition to these conceptual difficulties, a number of authors confuse the attack graph and attack tree — or at least 'blur' the difference. For instance, Khaitan and Raheja [227] refers to the attack tree proposal by Schneier [180] as an attack graph, and Chen and Cheng [228] refers to the paper by Ingols et al. [229] as a paper describing attack trees when in fact it describes an attack graph method. Some authors use the terms attack graph and attack tree interchangeably [225]. Perhaps recognising the conceptual similarities, a handful of authors have attempted to combine AMTs. For example, combine attack graphs and attack trees [193,226], attack trees and fault trees [230] and attack graphs with Petri nets [179].

4.0.1. Defining the graph

The graph based representation of cyber-attacks can be represented in the form: $G(V; E)$ which comprises of vertices $v \in V$ and edges: $e \in E$ which represent relationships between the nodes. The overall graph structure can be expressed as a tuple of the form $G = (S, \tau, S_0, S_s, L, EX)$ where:

- S is a finite set of states
- $\tau \subseteq S \times S$ is a transition relation
- $S_0 \subseteq S$ is a set of initial states
- $S_s \subseteq S$ is a set of success states — for example obtaining root or user privileges on a particular host
- $L : S \rightarrow 2^{AP}$ is a labelling of states with a set of atomic propositions (AP)
- EX is a finite set of exploits which connect the transition between two states

Vertices (V). The *vertices* - also referred to as *nodes*, can represent:

- An exploit that has been or could be applied to the given node [142,219,231].
- A precondition or postcondition.
- A combination of both the above.

Two examples of this are provided in Fig. 4. The example on the left is a graph where the vertices represent exploits, and the example on the right is a graph in which the vertices represent both preconditions/postconditions (`sshd(3,1)`) and exploits (`sshd_bof(3,1)`).

Edges (E). Edges can be directed — to represent specific transitions, or undirected — to represent a general connection between two nodes which indicates the perpetration of an exploit. Two examples of this are provided in Fig. 4. The example on the left demonstrates an undirected edge in which two exploits are connected together. The preconditions necessary for the execution of the `WUFTPDX(MAUDE,NED)` exploit are assumed to have been met in the transition. In other words, the authors have not considered it necessary to explicitly represent this. Such attack graphs are referred to in the literature as an *exploit-oriented graph* [231].

The example on the right demonstrates a directed edge in which a precondition (`sshd(3,1)`) is connected to an exploit (`sshd_bof(3,1)`).

This outlines one of the differences in the visual representation of an attack graph and an attack tree. Generally, attack trees use vertices to represent exploits and not preconditions, preconditions are assumed to have been met in the transition from one exploit to the next. Attack graphs represent both. 98 of the 118 attack graph configurations (83%) surveyed in Section 6 (Table 22), represented preconditions explicitly as a node. However, only 4 out of the 61 attack trees (6.5%, Table 23) specifically used different shapes to represent preconditions and exploits [232–235].

Weighted edges. Weights can be assigned to edges [141,145] or vertices [129] to represent costs, probability, risks or other metrics [141,236]. These metrics indicate the difficulty of perpetrating a particular type of attack, or probability/likelihood of a particular form of attack being successful [237]. The cost can be codified as continuous values, or as Boolean values such as: *easy* and *not easy*; *expensive* and *not expensive*; *intrusive* and *not intrusive* etc., and continuous values such as: *monetary cost to defend/attack*; *time to achieve/repel*; and *cost in resources to attack/defend* etc. [180].

An example of this is provided in the attack tree in Fig. 5 in which the authors calculate the *return on security investment (ROI)* and *return on attack (ROA)* of conducting particular exploits.

Sawilla and Ou apply the *AssetRank algorithm* (a modification of PageRank) to measure the importance of system privileges and vulnerabilities in an attack against a system. The resulting attack graph includes importance values which are added to each edge in the attack graph [173].

Attack prediction algorithms are typically based around Bayesian networks [85] or Markov decision processes [71] which rely on the metrics provided in the weighted edges. Examples of this include: the *behaviour based attack graph* which measures the risk of critical resources being compromised [85]; the *probabilistic attack graph* which calculates the annualised loss expectancy of a computer network to aid in prioritising vulnerability patching [71]; and the three probability measures (*exploit success probability, successful occurrence probability* and *condition obtained probability*) calculated by [224] to measure the likelihood: of a successful exploit; that the exploit has been applied; and that an attacker will achieve a given condition respectively. Further examples are provided in Table 4. This paper does not explore the representation of weighted edges.

Example of an AMT. It is now useful to apply the theory outlined above to an AMT example. The attack graph in Fig. 6 demonstrates how a perpetrator could gain `user` level privileges on a host which is denoted by the number 2. In this example, the goal is represented as an ellipse at the bottom of the attack graph (represented as `user(2)`).

The attack graph in Fig. 6 uses ellipses to represent the preconditions/postconditions, and rectangles to represent exploits. Edges in this attack graph connect preconditions to exploits indicating the preconditions required to achieve the exploit, and exploits to preconditions indicating that the successful achievement of the exploit produces a set of postconditions — which become the preconditions for a new exploit. There are four initial preconditions: `sshd(3,1)`, `user(3)` (presented twice) and `sshd(3,2)`. These are presented at the top of the attack graph.

The preconditions/postconditions and exploits are connected by a directed edge in the form of an arrow which represents event flow. So, the directed edge connecting `sshd_bof(3,1)` with `user(1)` (`sshd_bof(3,1)` → `user(1)`) indicates that when the exploit `sshd_bof(3,1)` is applied by host 3 on host 1, a new postcondition: `user(1)` is achieved which means that the perpetrator now has `user` privilege levels on host 1.

4.1. Attack graphs and attack trees

Having outlined the underlying theory relating to how AMTs are represented, the discussion now proceeds to describe attack trees and attack graphs.

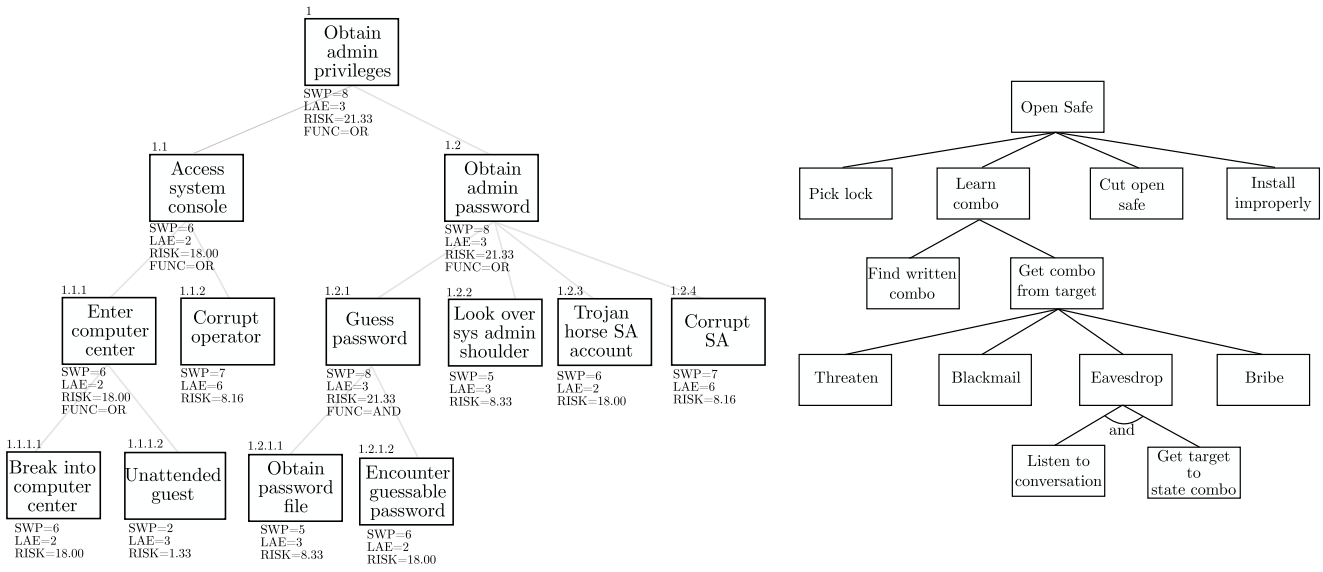

Fig. 7. A threat logic tree (left, adapted from [238]), and an attack tree (right, adapted from [180]).

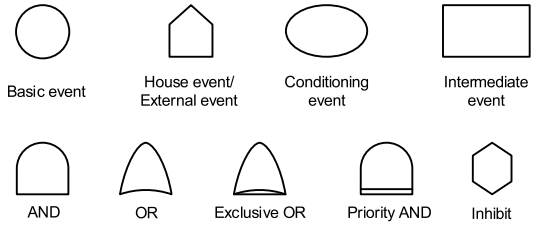

Fig. 8. Fault tree symbols [127].

4.1.1. Attack trees

Bruce Schneier is popularly attributed to have developed attack trees and to have introduced the concept of threat countermeasures and precondition logic [90,130,239–244]. The actual origins of attack trees can be found in the contributions by Weiss [238] which were developed further by Salter et al. [245] (co-authored by Bruce Schneier).

Weiss introduced *threat logic trees* in 1991 (Fig. 7, left), and provided an example which included risk measurements and precondition logic. Later, threat logic trees were recast in the form of attack trees by Salter et al. [245] - who also included the concept of threat countermeasures. However, the paper that receives all the credit for the inception of attack trees is the 1999 paper by Schneier [180]. As Fig. 7 (left) shows, if the textual labels are removed from the threat logic tree, there is no conceptual difference between the threat logic trees and the attack tree.

The visual structure of event trees and decision trees is similar to attack trees. Event trees [202] highlight success and failure in a system [246], whereas decision trees highlight system failure. Although both event trees and decision trees have been applied to a computer/cyber security context [234,238,247–250], neither of these methods have gained popularity.

Attack trees present cyber-attacks bottom-up. Shapes such as rectangles and ellipses, or plaintext with no shape are used to represent preconditions or exploits.

Two examples of the attack tree structure are provided in Fig. 7. Both the attack trees in these examples use rectangles to represent exploits. Preconditions are assumed to have been met in the transition between exploits. Fig. 7 (left) outlines the use of weights which in this case are added to nodes to identify risk levels.

Fig. 7 (right) outlines the steps that a perpetrator needs to complete to be able to `open a safe`. In this example, the conjunction of the perpetrator needing to `listen to conversation` AND `get target to state combo` leads to a successful `eavesdrop`. This is represented by the arc connecting the two edges accordingly. Although the word *and* is added to the arc to represent precondition logic. This is exceptional, and as the data in Table 23 shows, very few attack trees actually represent precondition logic.

In the same attack tree, the disjunctive (OR) relationship is represented by the absence of an arc. So, any one of: `threaten`, `blackmail`, `eavesdrop` or `bribe` will result in the perpetrator being able to: `get combo from target`.

Fault trees share the tree structure of attack trees The symbolic representation of fault trees was first proposed by the U.S. Nuclear Regulatory Commission [251]. Fault trees were later standardised by the IEC in 1990, [127], the European Cooperation for Space Standardization [252] and then by the British Standards Institute [253].

Fault trees are used in a number of industries such as in the aerospace industry [254–256], radioactive waste disposal [257], the automotive industry [258,259], and in the analysis of failure in computer systems [260–262]. Although the fault tree standard is a generic standard (not particularly focussing on cyber security as a target domain), more recently fault trees have become a popular means of representing cyber-attacks [234,263,264].

Contrary to the assertion by Mirembe and Muyeba [240] and Khand [264] - that fault trees lack suitable semantics to enable effective reasoning in regard to threat models, fault trees are in fact one of the most visually expressive AMTs because they utilise a wide range of standard symbols to express elements of an attack. The full set of defined symbols spans six pages and extends beyond the symbols presented in Fig. 8. The full set includes symbols for: *majority vote* gates, *priority and* (PAND) gates, *inhibit* gates, *NOT* gates and *sequential* gates — to name a few.

Fig. 9 outlines the steps that a perpetrator needs to complete in order to forge a package. This representation highlights some of the symbols used in the fault tree standard. For example, a precondition is represented by a circle, an exploit by a rectangle, and precondition logic by two distinct shapes (Fig. 8). The triangles represent a connection to another section of the fault tree — thus enabling fault trees to be 'compartmentalised'.

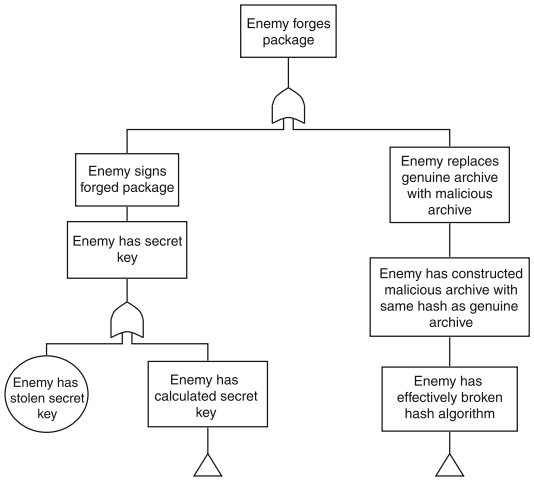

Fig. 9. A fault tree. *Source:* Reproduced with permission from [114].

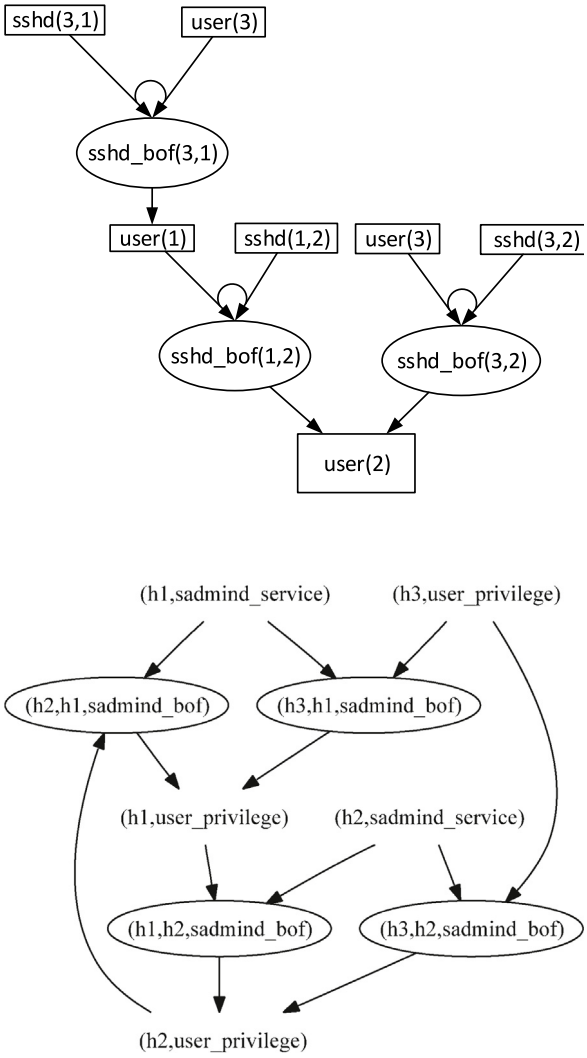

Fig. 10. Two sample attack graphs by Barik and Mazumdar [144, top] and Wang et al. [265, bottom] - both adapted.

4.1.2. Attack graphs

Attack graphs find their origins in the doctoral thesis and two early papers by Dacier [223,266,267]. These contributions emphasised the concept of a *privilege graph* which captures the inheritance of privileges. In this graph, a node represents a set of privileges owned by a user or a set of users and an edge represents a vulnerability. In such a graph, preconditions are presumed in the state transition.

Two attack graph examples are provided in Fig. 10. Both the attack graphs are examples of a partial attack graph wherein there is a single goal - `user` level access on host 2 - referred to as `user(2)` and `h2_user_privilege`.

The attack graph in Fig. 10 (top) represents exploits as rectangles and preconditions as ellipses. In this example, precondition logic is represented by the presence (AND) or absence (OR) of an arc. In Fig. 10 (bottom), exploits are represented as ellipses and preconditions as plaintext. In this example, precondition logic is not represented.

Fig. 10 (bottom) outlines the proliferation of the *sadmind malware* through a network of three hosts (`h1`, `h2` and `h3`) [265]. There are three important elements to this attack [268,269]:

- The perpetrator has local user privileges on host `h3` (`h3,user_privilege`) and aims to get user privilege status on hosts `h1` and `h2` i.e., to achieve postconditions: `h1,user_privilege` and `h2,user_privilege`.
- `h1` and `h2` are advertising the `sadmind` service. This is outlined in the graph as `h1,sadmind_service` and `h2,sadmind_service`. The `sadmind` service is a precondition to the attack.
- The perpetrator commits the `sadmind buffer overflow` exploit on both the hosts. This is outlined as `h3,h1,sadmind_bof` and `h3,h2,sadmind_bof` and this results in the user privilege status shown as `h2,user_privilege` in the graph.

As far as the AMT literature is concerned, attack graphs are the most popular form and a number of attack graph variants are highlighted in Table 6.

5. Visual syntax theory

This section briefly introduces a number of visual syntax design theories and principles which can be applied to design effective visual syntax design. These theories include Bertin's *visual variables* [316], Miller's 7 ± 2 Law [317], Petre's *primary and secondary notation theory* [318], *visual distance* (inspired by Petre [124,318]), the Gestalt theories [319], and Moody's *physics of notations* [124]. This discussion is followed by a description of further design considerations such as event flow, shapes and the use of colour.

5.1. Bertin's visual variables

Bertin proposes seven visual variables — position, size, shape, value, colour, orientation and texture [316]. These variables are generally used in most visual syntax based systems and are quite fundamental to visual syntax design. Moody [17] considers them to be the *"vocabulary"* and *"building blocks"* for visual syntax design and proposes that they are *"for graphic design what the periodic table is for chemistry"*.

Three of these variables — shapes, colour and texture are relevant to the present study. Shapes and colour are discussed later in Sections 5.7.2 and 5.7.3 respectively, and the application of all three in attack graphs and attack trees is discussed in further detail in Section 6.5.

Table 6
Types of attack graph.

Category	References
'Generic' attack graph	[30,34,35,37,41,45,46,50,52,55,56,60,61,65,67,79,85,107,143,144,146,150,152,155–157,159,161,166–169,173,177,178,193,216,219–222,224–226,231,265,270–294]
Alert correlation graph	[31,54,75]; *alert correlation graph*: [24,58,59,172][2]; *hyper-alert correlation graph*: [31,33,57,155]; *alert dependency graph*: [24]; *intrusion graph (i-graph)*: [149]
Vulnerability graph	*Exploitation graph*: [45,88,295]; *exploit oriented graph*: [231]; *state enumeration attack graph*: [168,296]; *dependency attack graph*: [296]; *coordinated attack graph*: [42]
Miscellaneous attack graph	*Personalised attack graph*: [297,298]; *host access graph*: [175]; *hybrid attack graph*: [171]; *knowledge graph*: [299]; *mission dependency graph*: [300]; *Bayesian attack graph*: [301]; *multiple prerequisite attack graph*: [34,302]; *evidence graph*: [303]; *logical attack graph*: [30,66,215,282,304–307]; *host-compromised attack graph*: [271]; *predictive attack graph*: [271,302,308–310]; *attack strategy graph*: [32]; *privilege graph*: [179,223,267,311,312], [141][1], [145][1]
Dependency graph	*Exploit dependency graph*[3]: [45,62,64,97,231,277,313,314]; *exploitation graph*: [88]; *hybrid dependency graph*: [47]; *general dependency graph*: [64]; *behaviour based attack graph*: [85]; *probabilistic attack graph*: [41,52,71]; *attack scenario graph*: [64]; *vulnerability cause graph*: [148,315]; *exploit oriented graph*: [231]

[1] Although the authors refer to the graph generally as an attack graph, it is in fact based on the *privilege graph* proposed by [223] and based around the acquisition of privileges.
[2] note that although Templeton and Levitt [172], Cuppens and Miege [58] and Alserhani et al. [165] did not specifically refer to their graph as an '*alert-correlation graph*' the description tallies with the alert correlation graph described by subsequent authors. Note also that Qin and Lee [75] and Sundaramurthy et al. [54] referred to their graph as a *correlation graph* and Sundaramurthy et al. refer to the graph in one instance as an *alert correlation graph*.
[3] Both terms - *exploit dependency graph* and *dependency graph* are used in the literature.

Table 7
AMTs that contradict Miller's 7 ± 2 Law.

AMT	Total	Colour	Edge	Shape
FACT Tree [230]	11	2	2	7
Bowtie by [207]	21	10	2	9
Misuse case map [94][1]	18	1	2	15
Misuse sequence diagrams [188]	15	5	2	11
CORAS [320,321]	20	4	3	13
Attack defense tree [322]	10	0	2	8
Incident tree [69]	10	6	1	3
Attack tree [323]	10	0	2	8
Attack graph [276]	13	9	1	3
Attack graph [281]	10	4	1	5[2]
Attack graph [55]	8	4[3]	2	2
Attack execution graph [77]	8	5	1	4
Attack graph [288]	9	5[4]	1	3

[1]Total count in this method was difficult to calculate as some - *exploit path without damage* for instance, could be considered to be a shape as well as a path. If treated in this way, the total count could be as much as 26. However, only those specifically referred to as a path by the authors have been considered thus.
[2]Plaintext element treated as a shape.
[3]All edges are blue, and this is not treated as a separate colour. In other words, if edges were black and blue, then the colour count would have increased by 1, in this case it has not.
[4]3 edges of different colours treated as 3 colours rather than 3 edge types.
Note: The data in the table is calculated as follows. The total number of colours (not including black and white), shapes and edge types were added together to give a total number of elements. The representation of precondition logic was counted as a shape (rather than an edge type). The upper section of the table reports AMTs that clearly violate Miller's law, i.e., the total number of elements is more than 9, whereas the lower part highlights those where the total number of elements is 8 or 9. Although Miller's law states 7 ± 2, the table ignores, 5, 6 and 7 total elements. In other words, it represents the more 'extreme' examples.

5.2. Miller's 7 ± 2 law

Miller [324] proposes that there is a relationship between the limits of one-dimensional absolute judgement and limits on short term memory. As the number of different stimuli increase, the ability to remember each stimuli decreases.

Miller's Law advocates that the average human can remember seven plus or minus two (7 ± 2) objects which implies that the larger the visual syntax of an AMT - i.e., the total number of shapes, colours and edge types, the less likely an observer is to effectively perceive the message being conveyed by the AMT.

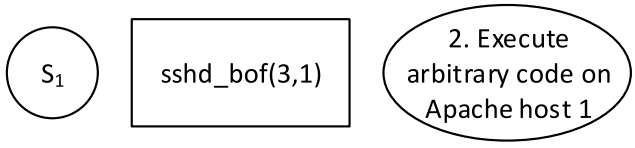

Fig. 11. Attack graph labelling. *Source:* Left - *character* [42], middle - *pseudonymous* [144], right - *textual* [149]. All diagrams adapted.

Examples of this as applied to AMTs are provided in Table 7 which shows that the Bowtie advocated by Levy et al. [207] conveys more than 10 colours, 6 icons, 2 shapes, and 2 connectors/edges, the misuse case maps proposed by Karpati et al. [94] comprises of 17 shapes, the misuse sequence diagrams proposed by Katta et al. [188] comprises of 13 shapes, and *CORAS* proposed by Rumbaugh et al. [325] comprises of 13 shapes and 3 edge types.

5.3. Petre's principle of primary and secondary notation

Petre makes the distinction between primary and secondary notation [318]. *Primary notation* is the set of visual variables that form the generic structure of diagrams — such as shapes and lines. *Secondary notation* - referred to by Moody [124] as dual coding, refers to objects that are not part of the formal definition of a framework, but which exhibit relationships and structures important in aiding the perception of the observer and providing further ancillary information. In addition, secondary notation creates a visual distance [124] (discussed in Section 5.4).

Examples of secondary notation include labels and text [326, 327] - which when combined with visual notation, are more effective than either one of them on their own [328] and more likely to improve perception [329].

Secondary notation should be used as a means of providing additional information regarding a concept or to differentiate sub-concepts, but not to represent the concept itself.

Labels are a form of secondary notation which are added to a node in an AMT to aid the understanding of events and statuses. AMTs utilise three types of label: *character, pseudonymous* and *textual*. These are demonstrated in Fig. 11.

A *character* label (Fig. 11, left) utilises a single character, number or other variable to represent the event. Character labels result in compact models and are useful in demonstrating concepts. However, they require recourse to a key or reference material to help understand the label.

A *pseudonymous* label (Fig. 11, middle) uses formal semantics such as an IDS alert ID or CVE ID. Pseudonymous labels bring an observer closer to the event as seen by an analyst, however, it requires recourse to a key or reference material for an observer not familiar with the domain.

A *textual* label (Fig. 11, right) is a textual description of the exploit/precondition. This requires little or no recourse to a separate key/description and is particularly useful if the graph is to be used to aid cyber-attack perception amongst non-experts. However, textual labels consume more graph space in comparison with pseudonymous and character labels. Tables 22 and 23 show that textual labels are particularly popular in attack trees ($n = 48$, 78.7%) in comparison with attack graphs ($n = 17$, 14.4%).

The use of textual labels impacts the shape that can be used in a model. Shapes such as ellipses and rectangles can accommodate textual labels, however, triangles, circles and diamonds cannot. This is discussed further in Section 6.5.3.

5.4. Visual distance

Section 5.3 outlined that Petre's secondary notation [318] can be used to create *visual distance* between objects [124]. Visual distance refers to the perceptible steps — in terms of variables such as shape, colour, value and texture, between objects [330–332]. This helps to distinguish between objects in a diagram.

One of the most effective ways of creating a perceptible distance is to use particular shape pairings. For example, there is a perceptible visual distance between a *circle*|*rectangle*, and *rectangle*|*hexagon* pairings. However, the visual distance is less perceptible in shape pairings such as a *right angled rectangle*|*rectangle with rounded corners* (referred to in this paper as a *rounded rectangle*), *circle*|*ellipse*, *square*|*rectangle* and *hexagon*|*octagon* pairings. Similarly, a shape filled with contrasting colours such as *black*|*white*, or *red*|*green* can create a perceptible visual distance. However, an alteration in edge colour or texture does not. In other words, the visual distance between two circles filled with red and green respectively is more perceptible compared to two circles with a red and green edge respectively.

Two examples demonstrate this. The shapes in the data flow diagram proposed by Gane and Sarson ([333], Fig. 12, top) have small visual distances because they are all rectangle variants. Two of the shapes are right angled rectangles (Fig. 12, top, left and right). One of these (right) has a vertical line. The third shape (middle) is a rounded rectangle with a horizontal line. On the other hand, the De Marco data flow diagrams [334, Fig. 12, bottom] maintains better visual distance by using two rectangles (distinguished by size) and a circle.

Another way of creating a visual distance is by using colour to fill objects. An example of this is provided by Man et al. [35, Fig. 13] where blue circles represent normal/user privilege levels on a host, and red circles represent root privilege [35]. In this example, colour provides further differentiation between the same construct — preconditions.

An example of inadequate visual distance is provided by Alhomidi et al. (Fig. 14). In this example, exploits and preconditions — the primary constructs, are represented using the same shape — an ellipse, and constructs are differentiated by using upper and lower case pseudonymous labels to represent preconditions (`IIS(0,2)` and `SCL(0,1)`) and exploits (for example `inj(1,2)`, `sshd_bof(2,3)`) respectively.

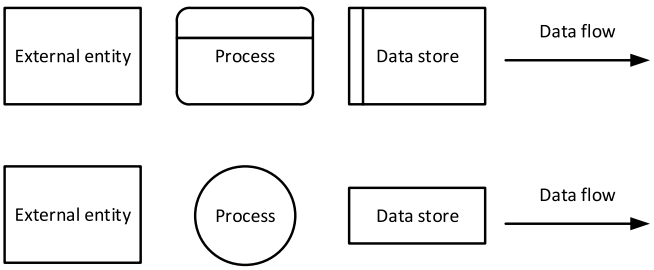

Fig. 12. Data flow diagram notation. Gane and Sarson [333, top] and De Marco [335, bottom].
Source: Both adapted from [124].

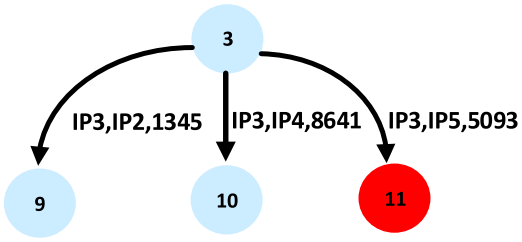

Fig. 13. Using secondary notation to create visual distance.
Source: Adapted from [35].

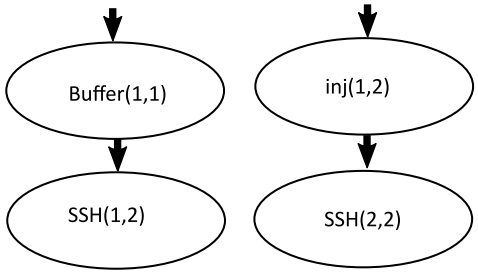

Fig. 14. An example of inadequate visual distance (adapted from [142]).

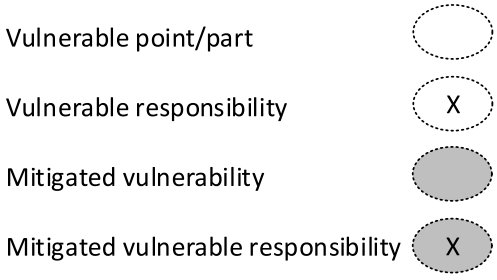

Fig. 15. Visual distance in misuse case maps.
Source: Adapted from [94].

This example contravenes Petre's *primary and secondary notation* theory because Alhomidi et al. have used secondary instead of primary notation to differentiate two constructs. Furthermore, the use of textual labels — albeit differentiated by upper case/lower case, conveys the unintended perception of similarity. This is referred to as Gestalt's *factor of similarity* (discussed later in Section 5.5). Consequently, a non-expert might not be able to easily differentiate between the preconditions (`inj(1,2)`) and the exploits (`SSH(2,2)`).

The visual distance between the *vulnerable point/part* and *vulnerable responsibility*; *vulnerable part* and *mitigated vulnerability*, and the *mitigated vulnerable responsibility* and *mitigated vulnerable*

Table 8
The Gestalt theories.

Factor	Description
Proximity	Objects placed closely together, can be perceived as being part of a whole. Examples of this include multiple shades of blue in the sky.
Similarity	When objects share similar properties — for example colour, they are perceived to be associated and/or similar.
Uniform destiny ('common fate')	When multiple factors (or multiple, different instances of a single factor) come into conflict, one factor is dominant, and overrides the other.
Prägnanzstufen' ('law of simplicity')	Simple arrangements of objects are likely to be perceived first.
Direction	Given a diagram with multiple branches, the branch that best follows the original line is likely to be perceived a continuation of that line.
Closure	If objects are close together, they are likely to be perceived as being part of a whole.
'Good curve'	Observers are likely to follow a continued line even if the angle changes.

responsibility in the *misuse case maps* [94, Fig. 15] is inadequate because these are differentiated by the presence or absence of a cross.

Visual distance in the context of attack graphs and attack trees is analysed in further detail in Section 6.4. As the discussion therein will show, numerous attack graph and attack tree configurations have been proposed in the research literature in which there is either no visual distance, or a small — barely perceptible, visual distance.

5.5. The Gestalt theories

The Gestalt theories of the perception and interpretation of grouped objects have become a popular mechanism for evaluating observer response to diagrams and objects [336]. These theories were framed by [319] into seven factors described in Table 8.

The Gestalt laws indicate that ineffectively designed diagrams can distract an observer from or conceal the intended meaning of the diagram.

5.6. Moody's physics of notations

One of the key contributions to research on effective visual syntax design are the *physics of notations* proposed by Moody [124]. The *physics of notations* are a set of nine visual syntax design principles drawn from various disciplines including: cognitive psychology, perceptual psychology, communication theory and cartography. These principles form the guidelines for effective diagrams and outline how the eight Bertin variables should be manipulated.

The nine principles (described more fully in Table 9) are:

1. Semiotic clarity
2. Perceptual discriminability
3. Semantic transparency
4. Complexity management
5. Cognitive integration
6. Visual expressiveness
7. Dual coding
8. Graphic economy
9. Cognitive fit

These methods are highlighted in Table 9.

5.7. Further visual syntax considerations

This section introduces a number of further visual syntax considerations such as the concept of event flow (Section 5.7.1), shape – one of Bertin's visual variables (Section 5.7.2) and colour (Section 5.7.3) – all of which are important elements of a visual syntax.

5.7.1. Event flow

Event flow refers to the direction that the sequence of events follow. Event flow is not represented in Bertin's model. This is surprising given that it is a critical variable in terms of how observers process information.

The direction of information flow in a diagram can be an important factor in aiding an observer's perception of the information contained therein. Studies into visual scanning have investigated scanning behaviour to identify location of fixation and direction of saccades — the movement of eyes in the same direction between two or more points of fixation. These studies have analysed, left–right and right–left saccades to understand eye movement habits [355].

Most research in this field has revealed that eye movement and scanning tendencies tend to be influenced by past experiences — many of which are culturally driven [355]. Aesthetic preferences in terms of imagery represented left–right or right–left is strongly influenced by reading habits — themselves driven by cultural habits [331,356,357].

There is less research into top-down/bottom-up preferences — possibly because most research focuses on reading preferences and/or aesthetic preferences of images rather than aesthetic preferences for process/event flow systems.

Events can be presented top-down — as in *flow charts* [358], TROPOS [359] and *SDL diagrams* [360], bottom-up — as in fault trees and attack trees, left–right — as in *CORAS* [192], Riskit [361], and the *event-driven architecture* [362], or right–left. There are very few if any event flow models where events are represented right–left.

Some frameworks allow diagrams to be presented according to a user's preference. For example *state diagrams* and the *DRAKON* model [363] can be presented either top-down and/or left–right.

The bottom-up approach adopted by fault trees and borrowed by attack trees aims to focus on the problem/goal — referred to as an *undesirable event* in an fault tree, and enable the non-expert to dissect the causes of the event by reading down to the causes. An observer might typically follow the tree top-down — despite it being presented bottom-up.

The attack graph begins the visual narrative at the top of the graph with an expression of the problems which ultimately lead to the goal/event — which is represented at the bottom. Again, the non-expert will follow this top-down.

5.7.2. Shapes

Shapes are one of the most important visual variables [340] and can capture important phenomena more powerfully and succinctly than words [364]. Shapes should be used as the primary means of communicating information as they are the primary visual variable for aiding object recognition [124].

The discussion in Section 5.4 has shown that the selection of shapes is important in aiding perception and that certain shape pairings such as *ellipse*|*circle*, or *rounded rectangle*|*rectangle* can be counter-intuitive.

A detailed analysis of the use of shapes in attack graphs is provided in Section 6. The discussion therein demonstrates that ellipses, rectangles, and circles are popular methods of presenting cyber-attack constructs. The discussion also highlights that some researchers prefer not to use a shape to represent a construct but prefer instead to represent the construct using plaintext.

Table 9
Moody's Physics of Notations [124].

Principle	References	Comments/examples
Semiotic clarity	Redundancy (a shape represents multiple concepts), and ambiguity (concept has no corresponding shape) should be eliminated	Of the 33 commonly used symbols in the UML class diagram, there are 5 synographs, 20 homographs and 2 symbol excesses [337]. 10 (84.7%) of the attack graphs reviewed and 37 (60.7%) of the attack trees reviewed had problems of ambiguity (Section 6.4).
Perceptual discriminability	Techniques such as: *visual distance* (Section 5.4), *shape primacy, redundant coding, perceptual popout* and *textual differentiation* should be used to make objects more distinguishable.	The misuse case maps [94] comprises of 17 shapes. A black triangle represents an *exploit path without damage, exploit path with possible damage, compromised/misused responsibility, order of the exploit paths* and *last one of the exploit paths*. Similarly, a dotted oval represents *vulnerable point/part, vulnerable responsibility, compromised/misused responsibility, mitigated vulnerability* and *mitigated vulnerable responsibility*. It is difficult to discriminate between these shapes without recourse to a key in working examples [94,117,338].
Semantic transparency	A concept should be directly derived from the object. This is *"the visual equivalent of onomatopoeia in spoken language"* [339]. A *semantically perverse* representation is one where an object infers a different or opposite meaning.	*Icon based shapes* are sometimes considered useful in aiding the perception of complex concepts [340]. A stick figure is to represent a user in UML. in *CORAS* icon of a man represents a threat or stakeholder, a bag of money represents an asset, a lock represents a vulnerability [320]. However, Masri et al. [341] found that whilst the use of icons was beneficial to most observers, they were less effective in aiding the perception of English-as-a-second-language (ESL) participants.
Complexity management	Complex diagrams can be managed by reducing the number of objects on the diagram.	This can be achieved by using methods such as *modularisation* — which divides a diagram into multiple smaller and more cognitively manageable components, and *hierarchy* - which abstracts elements of the diagram to different levels. The Bowtie proposed by Levy et al. [207] exemplifies the problem of complexity.
Cognitive integration	In diagrams comprising of multiple diagram forms, methods such as *conceptual integration* and *perceptual integration* should be used to reduce complexity.	*Conceptual integration* proposes the use of mechanisms that allow an observer to assemble information from multiple diagrams and *perceptual integration* proposes the use of mechanisms to enable easier navigation between diagrams. A good example of how complexity can be managed is provided in the fault tree models where a triangle is used to connect sections of a large model.
Visual expressiveness	More effective utilisation of design space can be achieved by maximising the use of Bertin's variables.	This principle appears to conflict with Miller's 7 ± 2 law [317,342,343] and to challenge Moody's own view that diagrams should be kept within *"perceptual and cognitive limits"* [17].
Dual coding	Text and visual notation are more cognitively effective when used together [328,329]	Similar to Petre's *principle of primary and secondary notation*, Moody proposes that secondary notation should be used to bolster cognitive perception.
Graphic economy	As time progresses, newer symbols are added to a framework and older ones rarely declared defunct — resulting in an over-sized syntax.	The decision not to explicitly represent secondary constructs such as initial preconditions, goals and perpetrator capability in some attack graphs, [175,243,271] can be seen to adhere with the principle of graphical economy. However, this creates a cognitive load for the observer who has to discern the secondary constructs.
Cognitive fit	Diagrams should e developed according to audiences. In theory this would suggest one visual syntax form for experts and another for non-experts.	Experts and non-experts process diagrams in different ways [331,344,345], however, few modelling systems are mature enough to make provision for both experts and non-experts. ORM (Object Role Modelling) [346] and Oracle Data Modelling [347] are exceptions.

5.7.3. Colour

Research into colour and perception has included attempts to understand the links between colour and perceived download speeds [365], the impact of colour in decision making [366], the link between colour depth and quality of perception [367], cultural and geographical differences in colour emotional responses [368,369], the effect of colour on investment decisions [370], and the link between colour and hazards [371].

Colour is a central variable in conveying information to an observer. Colour has suggestive power and the ability to capture and hold the attention of an observer [372]. If colour is added to a shape, it could have a greater impact on user interpretation than the shape on its own [373].

Notwithstanding the benefits of the use of colour, numerous considerations have to be made when designing colour based visual syntax. If not applied effectively, colour can create an imbalance between elements and make one element appear more important than another [372]. The number of colours used in a diagram is limited by one's working memory Hogganvik [320] and for each colour added to a diagram increases the cognitive load in terms of Miller's 7 ± 2 Law [324]. Table 7 outlines examples such as those by Levy et al. [207] and Sawilla and Ou [276] where more than eight colours have been used in the attack model.

Although colour is an effective variable, consideration should be given to a number of factors such as colour blindness - 8% of the global male and 0.5% of the global female population suffer from a form of colour blindness [374]; the likelihood that diagrams may be reproduced in black and white [375]; and consideration that colour is only suitable for conveying nominal data [376].

The discussion in Section 6.4.1 demonstrates that although colour is used frequently in attack graphs and attack trees, generally, there is no underlying rationale for the selection of colour to represent a particular concept.

5.8. Assessing the cognitive effectiveness of AMTs

Notwithstanding the benefits of AMTs in aiding cyber-attack perception — as outlined in Section 2, more research is needed to assess the cognitive effectiveness of AMTs.

Table 10 outlines previous research into the cognitive effectiveness of AMTs. The data describes the: AMTs compared in each study, format of the study, measures used to determine effectiveness, and study sample sizes.

Although the research outlined therein is a significant contribution to this domain, the data reveals a number of shortcomings relating to:

Table 10
Previous AMT comparison studies.

AMT	Description of study	Effectiveness measurement	n	*pref*[1]	Citation
Misuse cases	Effectiveness of AMT and practitioner perceptions	Case study with observations	10	i	[109]
The Common Criteria, misuse cases & attack trees	High level analysis of the *"learnability, usability, solution inclusiveness, clarity of output, and analyzability"* of AMTs	Self-observation/critical evaluation	2		[348]
DREAD, NIST SP800-30, OCTAVE-S & CORAS	Which AMT *"performs best"*	Observational. Completion of a risk reduction exercise using the four techniques	1		[349]
Misuse case & FMEA	Comparison of techniques for ability to identify user related failures	80 min task to analyse scenarios and identify failures	42	*TAM*	[350]
Attack trees & misuse cases	Comparison of techniques in aiding practitioner perception in threat identification	2 × 90 min controlled experiments to measure performance and perception	28/35[2]	*TAM*	[186]
Misuse case maps	Effectiveness in aiding non-expert stakeholders develop an understanding of multi-stage intrusions	Questionnaire response	12	*TAM*	[94]
Misuse case maps	Effectiveness in aiding observers find vulnerabilities and mitigations	Controlled experiment/test to solve series of tasks and self-reported TAM score	33	*TAM*	[338]
Attack trees	Suitability for modelling cyber-threat and in aiding experts understand threat	Qualitative interview	2		[351]
Misuse case maps & misuse sequence diagrams	Comparison of techniques for understanding, performance and perception	90 min task comprising of T/F questions (understanding), identifying/listing vulnerabilities (performance)	42	*TAM*	[188]
CORAS	The effect of visual syntax on understanding a risk scenario using the *CORAS language*	Questions relating to model navigation and understanding of concepts	25		[352]
CORAS	What is the preferred method of visualising vulnerabilities and visualising risk? comparison of the UML profile and the standard UML use case icons	Survey comparing alternative representations of risk scenarios	33		[353]
CORAS	An empirical investigation of risk modelling preferences among professionals and students to improve	Questionnaire emailed to participants to make selection between modelling alternatives	33		[25]
Attack graph & fault tree	An empirical investigation into the effectiveness of both techniques in aiding cyber-attack perception	computer based test	63		[19]
27 attack graph configurations	Evaluation of visual syntax preferences	Participants ranked preferred configurations	212	Conjoint analysis	[20]

[1]*pref*: Preference/acceptance testing method. *i*=interview; *TAM*=Technology Acceptance Model [354]
[2]2 separate experiments

1. *The diversity of AMT selection.* There are no known comparative studies into the effectiveness of attack graphs in aiding cyber-attack perception. Although there are conceptual similarities in the visual syntax of attack trees and attack graphs, only three of the studies under review considered attack trees [186,348,351].
2. *Fundamental conceptual differences in AMTs.* The visual syntax of some of the AMTs compared in these studies is so conceptually different that their selection could be considered to be cognitively biased. For example, the studies by Diallo et al. [348] and Opdahl and Sindre [186] compared common criteria method and misuse cases with attack trees, and misuse cases with attack trees respectively. The visual syntax of misuse cases and attack trees are fundamentally different and likely at the outset to render results in favour of the attack tree — which they did.
3. *Statistical significance.* In a number of studies, the number of participants have been too small to allow for statistically significant conclusions [94,109,348,349,351].
4. A number of studies outline the need to ground studies into the effectiveness of AMTs with firm pedagogic underpinning [109,348,349,351]. In these studies, the measures of effectiveness are unclear and seem to be based on personal judgement rather than in recognised methodologies and theories.

Collectively these shortcomings outline the need for further research and they also define some of the parameters that comparative evaluations should consider. The research presented in Lallie et al. [19,20] aims to address these shortcomings and propose an attack graph visual syntax which is cognitively effective and also preferred by practitioners — thereby increasing the likelihood of adoption.

This problem could be reduced or eliminated if appropriate visual syntax design theories – such as those described in this section – or others such as: the Cognitive Theory of Multimedia [377], Cognitive Dimensions of Notations [378], Guidelines of Modeling (GoM) [379], Semiotic Quality (SEQUAL) framework [380] were applied to the design and testing of the visual syntax.

A number of design theories have been applied in numerous contexts to test visual syntax design. However, most of these efforts have applied these principles post-design. For example, [115] applied the physics of notations to the KAOS goal modelling language and its supporting tool, *Objectiver* to determine how well the model complies with the principles. El Kouhen et al. [337] applied the physics of notations to a study in which non-expert users proposed design notations aimed at speeding up perception of UML.

Masri et al. [341] applied the cognitive theory of multimedia learning [377] and the cognitive load theory [381,382] to assess the impact of icons in entity relationship diagrams on user perception.

Further contributions have included the application of design testing methods to i* [*i-star*383], BPNM [327], UML [340,384], and misuse cases [320].

Not only have these efforts to evaluate methods been done — as Moody [17] puts it in a *"limited fashion"*, but most of these efforts have been applied post-design. The contributions by Hogganvik and Lallie et al. [19,20,320] are notable exceptions which apply these techniques at the outset when designing conceptual modelling systems.

6. An analysis of attack graph and attack tree visual syntax

This Section draws on the visual syntax design theories presented in Section 5 and presents a quantitative analysis of the visual syntax used in attack graphs and attack trees.

The analysis seeks to determine the popular methods of presenting constructs such as event flow, preconditions, exploits and precondition operators. This reveals *custom and practice* which outlines popular modes of visual expression.

In order to determine custom and practice, two questions are asked whenever a construct is analysed:

1. Is the construct commonly represented in attack graphs and/or attack trees?
2. If so, how — in quantitative terms, is the construct represented in the literature?

The structure of this section can be described as follows. The discussion begins with an explanation and demonstration of the *internal and external semiotic inconsistency* problem. This is a key problem which applies to AMTs described throughout this section. This is followed by an analysis of event flow representation in attack graphs and attack trees to determine the popular forms of event representation. The discussion proceeds to demonstrate why it is important to maintain an adequate visual distance between objects before analysing the use of colour and shape edge effects to increase the visual distance. Following this, the analysis focuses on the shapes used to represent preconditions (including initial preconditions, postconditions and goals), exploits and precondition operators. This discussion also considers why shapes such as diamonds and circles are wholly unsuited in AMTs.

Review methodology. The data outlined herein was collected through a systematic literature review which sourced peer-reviewed journal/conference papers and books/chapters which specifically use attack graphs or attack trees to demonstrate cyber-attack related concepts. 223 attack graph related papers and 147 attack tree papers were analysed giving 370 papers in total. The search terms applied in the review are described in Table 11. A process of forward and backward snowballing was applied to identify further sources. No date/time limitations were applied to the searches.

Table 11
Search terms applied in the literature search.

Category	Search terms
Use case based AMTs	Abuse cases, misuse cases, misuse case maps, misuse sequence diagrams, security use cases, CORAS
Temporal methods	Diamond model, cyber kill chain, Riskit
Miscellaneous methods	Event tree, OCTAVE, the Bowtie method, influence diagrams, extended influence diagram
Attack trees	Attack tree, defense tree, attack defense tree, penetration attack tree, cyber threat tree, security goals indicator tree, incident tree, FACT tree, attack countermeasures tree, dynamic attack tree, BDMP tree, threat tree, threat net, attack nets, protection tree, vulnerability tree, fault trees
Attack graph	Attack graph, alert correlation graph, hyper-alert correlation graph, alert dependency graph, intrusion graph, i-graph, vulnerability graph, exploit oriented graph, state enumeration attack graph, dependency attack graph, coordinated attack graph, personalised attack graph, host access graph, hybrid attack graph, knowledge graph, mission dependency graph, Bayesian attack graph, multiple prerequisite attack graph, evidence graph, logical attack graph, host-compromised attack graph, predictive attack graph, attack strategy graph, privilege graph, dependency graph, exploit dependency graph, exploitation graph, hybrid dependency graph, general dependency graph, behaviour based attack graph, probabilistic attack graph, attack scenario graph, vulnerability cause graph, exploit oriented graph

Sources were identified through Google Scholar, Web of Science and Scopus. From this analysis, 120 attack graph and 61 attack tree visual syntax configurations used in published papers were analysed.

Each attack graph/attack tree figure and the corresponding narrative provided by the author was analysed at source to understand the methods used to represent event flow, precondition operators, preconditions, exploits, attack goals, the use of colour, the use of labels and the use of shape edge texture.

The resulting data is presented in Tables 22 and 23. These two tables form the raw data from which the subsequent analysis is done.

The analysis counts attack graph/attack tree configurations and not authors. In other words, if an author has presented the same attack graph/attack tree configuration in more than one paper, then that is counted as a single attack graph/attack tree. Examples of this include the: *attack graphs* by Jha et al. [41,52] and Sheyner et al. [168],[1] *attack graphs* by Li and Vaughn [385], Li et al. [88], *attack graphs* by Liu et al. [272], Liu [37], *exploit dependency graphs* by [313], Wang et al. [62],[2] *hyper alert correlation graphs.* by Ning and Xu [32], Ning et al. [33], Ning et al. [57], *attack graphs* by Sheyner and Wing [152], Sheyner [270], *host attack graphs* by Xie et al. [237], Xie et al. [386], *protection trees* by Edge et al. [387], Edge [388] and *attack countermeasure trees* by Roy et al. [389], Roy et al. [390], Roy et al. [391].

The analysis relies heavily on quantitative data. Occasionally, this data is presented as $n = 21$, *38.9%*. In this example, n is the

[1] Sheyner et al. is co-authored by Jha et al. and treated as the 'same' attack graph.

[2] Wang et al. is co-authored by Noel et al. and treated as the 'same' exploit dependency graph.

Table 12
Review of representation of labels.

	Attack tree						Attack graph						All					
Shape	n	%	β	σ	z	p	n	%	β	σ	z	p	n	%	β	σ	z	p
No label	.	.	.	.	.	.	3	2.5	0.85	0.98	0.87	0.39	3	1.7	0.85	0.98	0.87	0.39
Pseudonymous	4	6.6	1.10	0.94	1.17	0.24	65	54.2	3.78	0.83	4.57	0.00	69	38.1	3.84	0.83	4.65	0.00
Label	10	16.4	1.95	0.87	2.23	0.03	29	24.2	2.98	0.84	3.56	0.00	39	21.5	3.27	0.83	3.93	0.00
Textual	47	77.0	3.46	0.83	4.17	0.00	23	19.2	2.75	0.84	3.27	0.00	70	38.7	3.85	0.83	4.67	0.00
Total							120						181					

σ = standard deviation

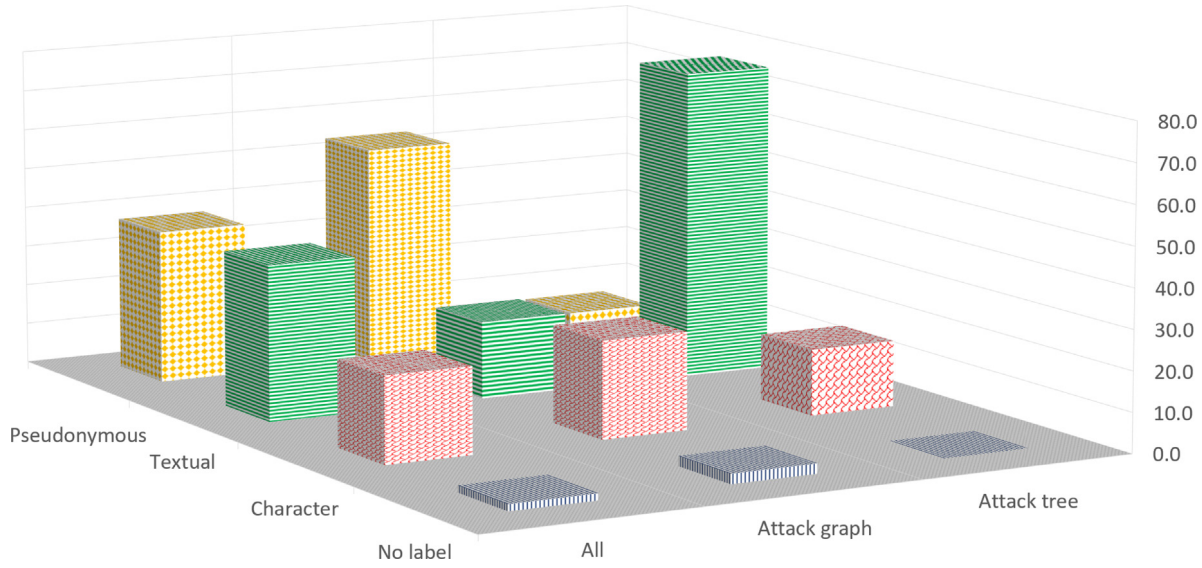

Fig. 16. Review of representation of labels.

total number and the percentage is calculated out of the total number of attack graphs or attack trees surveyed which is 120 and 61 respectively.

A Poisson log-linear model was used to calculate parameter estimates (β - also referred to as coefficients) and the effect size (z). Parameter estimates outline the size of the contribution of a predictor and describe the effect of a one-unit change in the predictor on the response if all other predictors are kept constant, and the effect size is a method of quantifying the difference between groups. Collectively, this helps to establish whether there is an association between categorical variables (i.e., whether the variables are independent or related.)

Section 6.2 will show that a number of authors represent the same construct using more than one shape. For example, [148] used a rectangle and a hexagon to represent a precondition in the same attack graph.

14 authors represented preconditions using multiple shapes in the same paper. 8 authors represented exploits using multiple shapes in the same paper. 6 authors represented both preconditions and exploits using multiple shapes in the same paper. This provides a total of 20 authors who represented preconditions using multiple shapes in the same paper and 14 authors who represented exploits using multiple shapes in the same paper. In these cases, it would be unfair to arbitrate in favour of one or the other. Consequently, configurations which represented the same construct using more than one shape were disqualified from the analysis.

This problem did not apply for any other constructs. Initial preconditions, precondition operators, goals etc., are analysed using the full 181 configurations.

6.1. Labels

Table 12 and Fig. 16 outline the use of labels in attack graphs and attack trees.

The data shows that for attack graphs and attack trees overall, the popular approach to the use of labels is to present them as: pseudonymous labels ($n = 69, 38.1\%, \beta = 3.84, z = 4.65, p = 0.00$), character labels ($n = 39, 21.5\%, \beta = 3.27, z = 3.93, p = 0.00$) and textual labels ($n = 70, 38.7\%, \beta = 3.85, z = 4.67, p = 0.00$).

When this data is analysed for attack graphs and attack trees separately, the results show that while pseudonymous labels are popular in attack graphs ($n = 65, 54.2\%, \beta = 3.78, z = 4.57p = 0.00$), they are not in attack trees ($n = 4, 6.6\%, \beta = 1.10, z = 1.17p = 0.24$).

Character labels are popular in both attack trees ($n = 10, 16.4\%, \beta = 1.95, z = 2.23p = 0.03$) and attack graphs ($n = 29, 24.2\%, \beta = 2.979, z = 3.56p = 0.00$). Similarly, textual labels are popular in both attack trees ($n = 47, 77\%, \beta = 3.46, z = 4.17, p = 0.00$) and attack graphs ($n = 23, 19.2\%, \beta = 2.752, z = 3.267p = 0.00$).

6.2. Internal and external semiotic inconsistency

The *principle of semiotic clarity* was described in Section 5 which explained that each concept should have one corresponding object in order to reduce redundancy and ambiguity. This section proposes two variations of the principle of semiotic clarity: *internal semiotic inconsistency* and *external semiotic inconsistency*.

Internal semiotic inconsistency refers to instances where authors represents the same construct using two or more different

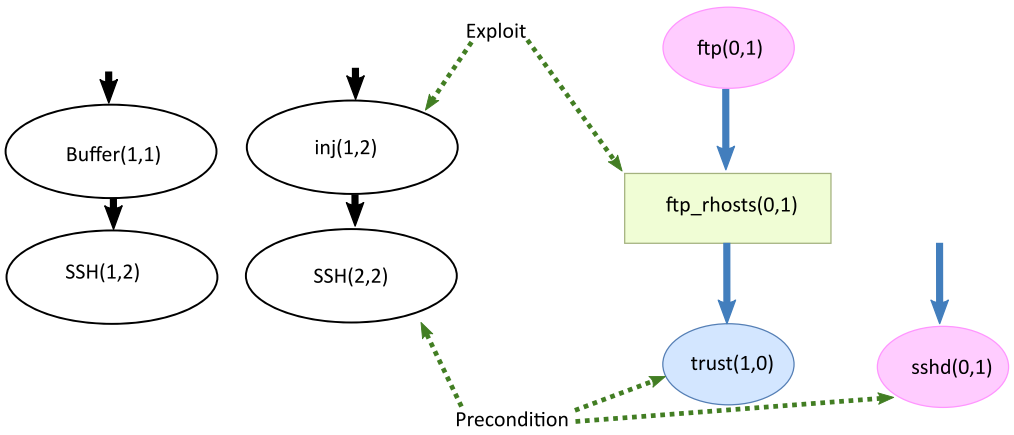

Fig. 17. An example of external semiotic inconsistency.
Source: Adapted from [142] (left) and [55] (right).

Table 13
Attack graph internal semiotic inconsistency.

	Semiotic inconsistency	Citation
Separate papers	g_o *vs* g_e	[169] *vs* [281]
	pr_c, ex_r *vs* unclear representations	[60] *vs* [284]
	pr_p, ex_e *vs* pr_e, ex_r	[150] *vs* [144]
Same paper	pr_r *vs* pr_p	[392]
	pr_r *vs* pr_h	[148]
	pr_{r_blue} *vs* pr_{h_purple}	[55]
	pr_r *vs* pr_h	[315]
	or_{td} *vs* or_{lr}*	[271]
	pr_e, ex_r *vs* pr_r, ex_{na}	[219]
	ef_{bu} *vs* ef_{td}	[224]
	ex_e, pr_d *vs* ex_d, pr_e	[282]
	pr_p, ex_e *vs* pr_p, ex_c	[56]

*This is for both the attack graph and predictive attack graph in the same paper.
Key: *g*=goal; *pr*=precondition, *ex*=exploit, *ef*=event flow; *o*=octagon; *e*=ellipse; *c*=circle; *r*=rectangle; *p*=plaintext; *h*=hexagon; *rr*=rounded rectangle; *td*=top-down; *lr*=left–right; *bu*=bottom-up

variables in the same paper or in subsequent papers, i.e., a construct is represented by not one but multiple shapes — thereby generating redundancy.

A visual example of the internal semiotic inconsistency problem is presented in the two papers by Barik and Mazumdar [144, 150] who presented preconditions using plaintext [150] and then in a subsequent paper [144], using ellipses. In the same two papers, Barik and Mazumdar represented exploits using ellipses [150] and then using rectangles [144].

Further examples of *internal semiotic inconsistency* in attack graphs are presented in Table 13 which highlights a number of inconsistencies in papers by the same author/authors, and most notably a number of examples of *internal semiotic inconsistency* within the same paper. In all these cases, the narrative presented by authors was explored to understand why the visual syntax was different. Generally, the reasoning for these inconsistencies was not clear.

External semiotic inconsistency refers to instances where multiple authors represent the same construct in different ways. An example of the *external semiotic inconsistency* problem is presented in Fig. 17 which demonstrates how the same construct is represented in different ways by different authors.

Throughout the rest of this section, further examples of internal and external semiotic inconsistency are provided in terms of the way preconditions, exploits and precondition operators are represented.

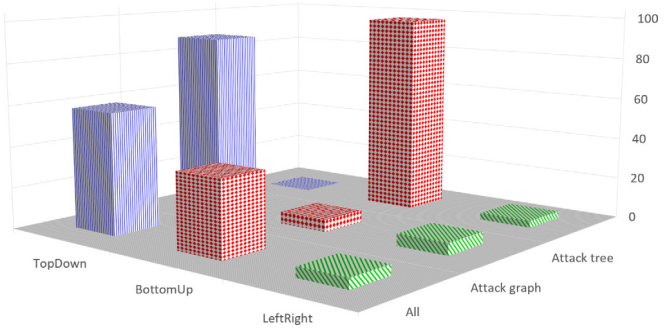

Fig. 18. Review of shapes used to represent event flow.

6.3. Event flow

The discussion in Section 3 highlighted that one of the key differentiators in the visual representations of attack graphs and attack trees is the method used to represent event flow.

Figs. 18 and 23 highlight the approaches towards representing event flow in attack graphs and attack trees. These approaches are: top-down, bottom-up and left–right. There are no examples of the right–left representation of event flow. 58.6% of all the AMTs represented event flow as top-down ($\beta = 4.26, p = 0.00$) and 36.5% represented event flow as bottom-up ($\beta = 3.79, p = 0.00$). There was a stronger effect size in favour of the top-down approach ($z = 5.19$) compared to the bottom-up approach ($z = 4.59$). The effect sizes could be subject to Simpson's Paradox and can be explained by there being more attack graphs in the sample than attack trees. The primary method of representing event flow in attack graphs is the top-down approach ($\beta = 4.26, p = 0.00$) and in attack trees is the bottom-up approach ($\beta = 3.68, p = 0.00$). The effect size in both these cases is large ($z = 5.19, z = 4.45$ respectively).

There were some anomalies to this observation. 7 (*5.8%*, $\beta = 1.61$) attack graphs were presented bottom-up and 7 (*5.8%*, $\beta = 1.61$) were presented left–right. The effect size was small in both cases ($z = 1.80$). The attack graphs presented bottom-up were by [30,33,146,147,178,193,193,271,286,292,297,298].

2 (*3.3%*, $\beta = 0.51$) attack trees were presented left–right, these were the *attack tree* by [203], and the *penetration attack tree* by [393].

The bottom-up attack graph presented in [147] is the combination of two modelling techniques — an attack graph and an *extended influence diagram*. The two attack graphs presented

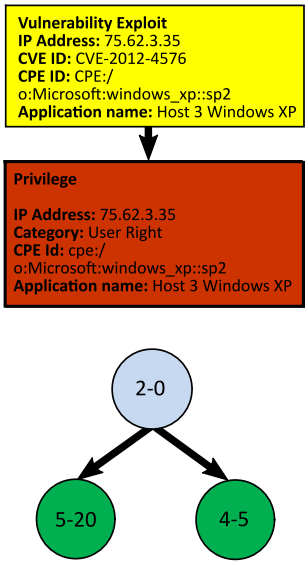

Fig. 19. Examples of a perceptible visual distance. *Source:* Top: [107], bottom: [285] - both adapted.

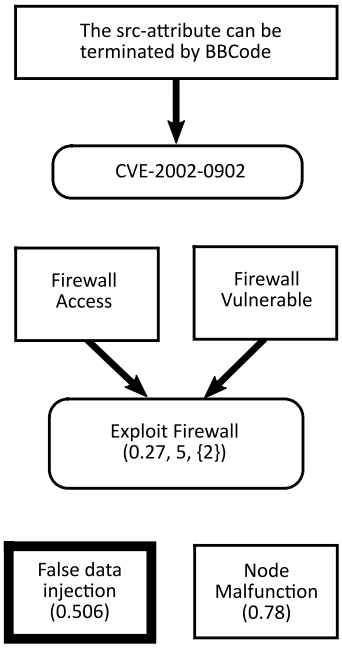

Fig. 20. Examples of a less perceptible visual distance. *Source:* Top: [315]; middle: [220]; bottom: [292] - both adapted.

by [224] were presented bottom-up for illustrative purposes. The final attack graph in the same paper was presented top-down.

The decision to present the *hyper alert correlation graph* left–right by [33] might have been influenced by the amount of text the authors needed to present in each ellipse. The *predictive graph* presented by [271] is presented both top-down and left–right. The same applies to the general attack graph. In this case, there appears to be no rationale for the event flow.

6.4. Visual distance in attack graphs and attack trees

Section 5.4 introduced the concept of visual distance. The discussion in Section 5.4 outlined that variables such as colour,

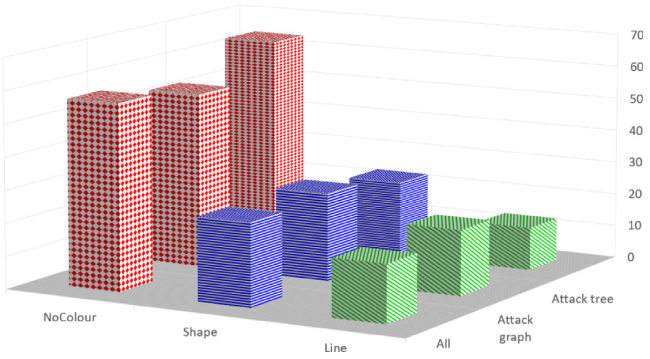

Fig. 21. Review of the use of colour in attack graphs and attack trees.

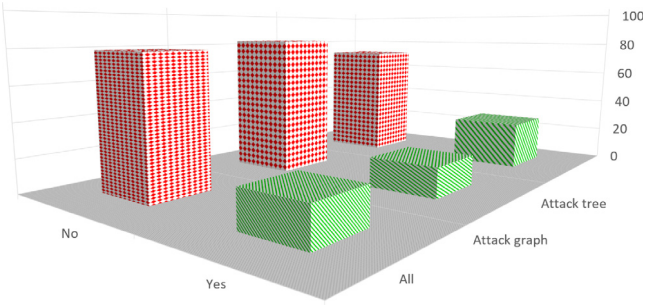

Fig. 22. Review of the use of shape edge texture.

Table 14
Visual distance in attack graphs and attack trees.

VD*		Description and supporting citations
Attack graph	No visual distance	[142,149,157,175,280,282,286,291,293,392]
	Small visual distance	**Colour** *red and yellow rectangle:* [107]; *colour differentiates ex and pr:* [285]; *thick edge differentiates exploits:* [292]; *rectangle and rounded rectangle:* [220,315]
Attack tree	No visual distance	[23,23,69,90,92,93,130,180,230,239,243,256,263, 264,322,394–415]
	Small visual distance	*Ellipse and rectangle appears to be used interchangeably:* [416,417]

*VD = Visual Distance

value and texture — three of Bertin's visual variables, can be used to increase the visual distance between objects. The discussion also outlined that careful consideration should be made to ensure that there is a perceptible visual distance between two objects that represent different constructs. A difference in shape or the colour that fills the shape creates a perceptible visual distance. However, an alteration in edge colour or texture does not.

Table 14 outlines 54 instances where there is either no visual distance between constructs i.e., they utilise the same shape for both preconditions and exploits, or a small — possibly imperceptible visual distance. This can be analysed further to show that 10 (8.5%) attack graphs and 37 (60.7%) attack trees had no visual distance between objects.

A number of examples demonstrate a perceptible visual distance. Kaynar and Sivrikaya [107] (Fig. 19 top) used the colour of the rectangle — red or yellow, to distinguish between a precondition or exploit respectively. Similarly, Nandi et al. [285] (Fig. 19 bottom) use colour to differentiate exploits and preconditions.

The example by Chaufette and Haag [315] (Fig. 20 top) and Durkota et al. [220] (Fig. 20 middle) utilise a rectangle and a

Table 15
Review of shapes used to represent preconditions.

		Attack tree						Attack graph						All					
	Shape	n	%	β	σ	z	p	n	%	β	σ	z	p	n	%	β	σ	z	p
Precondition	noshape	15	28.8	2.34	0.86	2.73	0.01	28	28.9	2.94	0.84	3.52	0.00	43	28.9	3.37	0.83	4.06	0.00
	ellipse	1	1.9	0.00	1.15	0.00	1.00	17	17.5	2.46	0.85	2.89	0.00	18	12.1	2.51	0.85	2.96	0.00
	rectangle	21	40.4	2.66	0.84	3.15	0.00	15	15.5	2.34	0.86	2.73	0.01	36	24.2	3.19	0.83	3.83	0.00
	circle	7	13.5	1.61	0.89	1.80	0.07	10	10.3	1.95	0.87	2.23	0.03	17	11.4	2.46	0.85	2.89	0.00
	plaintext	5	9.6	1.30	0.92	1.41	0.16	24	24.7	2.79	0.84	3.32	0.00	29	19.5	2.98	0.84	3.56	0.00
	diamond	.	.	.	.	.	.	3	3.1	0.85	0.98	0.87	0.39	3	2.0	0.85	0.98	0.87	0.39
	rrectangle†	3	5.8	0.85	0.98	0.87	0.39	na	.	.	.	.	.	3	2.0	0.85	0.98	0.87	0.39
	Total	52						97						149					
Initial preconditions	noshape	59	96.7	3.68	0.83	4.45	0.00	107	89.2	4.27	0.82	5.20	0.00	166	91.7	4.71	0.82	5.74	0.00
	ellipse	.	.	.	.	.	.	4	3.3	1.10	0.94	1.17	0.24	4	2.2	1.10	0.94	1.17	0.24
	rectangle	2	3.3	0.51	1.03	0.50	0.62	4	3.3	1.10	0.94	1.17	0.24	6	3.3	1.47	0.91	1.62	0.11
	circle	.	.	.	.	.	.	3	2.5	0.85	0.98	0.87	0.39	3	1.7	0.85	0.98	0.87	0.39
	triangle	.	.	.	.	.	.	2	1.7	0.51	1.03	0.50	0.62	2	1.1	0.51	1.03	0.50	0.62
	Total	61						120						181					
Attack goal	noshape	49	80.3	3.50	0.83	4.22	0.00	93	77.5	4.13	0.82	5.02	0.00	142	78.5	4.55	0.82	5.55	0.00
	ellipse	.	.	.	.	.	.	5	4.2	1.30	0.92	1.41	0.16	5	2.8	1.30	0.92	1.41	0.16
	rectangle	7	11.5	1.61	0.89	1.80	0.07	7	5.8	1.61	0.89	1.80	0.07	14	7.7	2.27	0.86	2.65	0.01
	circle	2	3.3	0.51	1.03	0.50	0.62	5	4.2	1.30	0.92	1.41	0.16	7	3.9	1.61	0.89	1.80	0.07
	plaintext	2	3.3	0.51	1.03	0.50	0.62	4	3.3	1.10	0.94	1.17	0.24	6	3.3	1.47	0.91	1.62	0.11
	diamond	.	.	.	.	.	.	1	0.8	0.00	1.16	0.00	1.00	1	0.6	0.00	1.16	0.00	1.00
	hexagon	.	.	.	.	.	.	1	0.8	0.00	1.16	0.00	1.00	1	0.6	0.00	1.16	0.00	1.00
	octagon	.	.	.	.	.	.	4	3.3	1.10	0.94	1.17	0.24	4	2.2	1.10	0.94	1.17	0.24
	rrectangle†	1	1.6	0.00	1.16	0.00	1.00	.	.	.	.	.	.	1	0.6	0.00	1.16	0.00	1.00
	Total	61						120						181					
Exploit	noshape	5	8.9	1.30	0.92	1.41	0.16	6	5.4	1.47	0.91	1.62	0.11	11	6.6	2.04	0.87	2.35	0.02
	ellipse	3	5.4	0.85	0.98	0.87	0.39	42	37.8	3.34	0.83	4.03	0.00	45	26.9	3.41	0.83	4.11	0.00
	rectangle	31	55.4	3.05	0.84	3.64	0.00	17	15.3	2.46	0.85	2.89	0.00	48	28.7	3.48	0.83	4.19	0.00
	circle	5	8.9	1.30	0.92	1.41	0.16	16	14.4	2.40	0.85	2.81	0.01	21	12.6	2.66	0.84	3.15	0.00
	plaintext	6	10.7	1.47	0.91	1.62	0.11	22	19.8	2.71	0.84	3.21	0.00	28	16.8	2.94	0.84	3.52	0.00
	diamond	.	.	.	.	.	.	2	1.8	0.51	1.03	0.50	0.62	2	1.2	0.51	1.03	0.50	0.62
	hexagon	1	1.8	0.00	1.16	0.00	1.00	1	0.9	0.00	1.15	0.00	1.00	2	1.2	0.51	1.03	0.50	0.62
	octagon	1	1.8	0.00	1.16	0.00	1.00	.	.	.	.	.	.	1	0.6	0.00	1.16	0.00	1.00
	triangle	.	.	.	.	.	.	1	0.9	0.00	1.15	0.00	1.00	1	0.6	0.00	1.16	0.00	1.00
	rrectangle†	4	7.1	1.10	0.94	1.17	0.24	4	3.6	1.10	0.94	1.17	0.24	8	4.8	1.74	0.89	1.96	0.05
	Total	56						111						167					
Event flow	topdown	0	0.0	.	.	.	.	106	88.3	4.26	0.82	5.19	0.00	106	58.6	4.26	0.82	5.19	0.00
	bottomUp	59	96.7	3.68	0.83	4.45	0.00	7	5.8	1.61	0.89	1.80	0.07	66	36.5	3.79	0.83	4.59	0.00
	leftright	2	3.3	0.51	1.03	0.50	0.62	7	5.8	1.61	0.89	1.80	0.07	9	5.0	1.85	0.88	2.10	0.04
	Total	61						120						181					
Colour	no colour	39	63.9	3.27	0.83	3.93	0.00	64	53.3	3.76	0.83	4.55	0.00	103	56.9	4.23	0.82	5.15	0.00
	shape	14	23.0	2.27	0.86	2.65	0.01	32	26.7	3.08	0.84	3.68	0.00	46	25.4	3.43	0.83	4.14	0.00
	line	8	13.1	1.74	0.89	1.96	0.05	24	20.0	2.79	0.84	3.32	0.00	32	17.7	3.08	0.84	3.68	0.00
	Total	61						120						181					
Line texture	No	44	72.1	3.39	0.83	4.09	0.00	99	82.5	4.20	0.82	5.10	0.00	143	79.0	4.56	0.82	5.56	0.00
	Yes	17	27.9	2.46	0.85	2.89	0.00	21	17.5	2.66	0.84	3.15	0.00	38	21.0	3.25	0.83	3.90	0.00
	Total	61						120						181					
Precondition operators	noshape	7	11.5	1.61	0.89	1.80	0.07	105	87.5	4.25	0.82	5.17	0.00	112	61.9	4.32	0.82	5.25	0.00
	ellipse	1	1.6	0.00	1.16	0.00	1.00	2	1.7	0.51	1.03	0.50	0.62	3	1.7	0.85	0.98	0.87	0.39
	rectangle	2	3.3	0.51	1.03	0.50	0.62	2	1.7	0.51	1.03	0.50	0.62	4	2.2	1.10	0.94	1.17	0.24
	circle	2	3.3	0.51	1.03	0.50	0.62	2	1.7	0.51	1.03	0.50	0.62	4	2.2	1.10	0.94	1.17	0.24
	plaintext	1	1.6	0.00	1.16	0.00	1.00	3	2.5	0.85	0.98	0.87	0.39	4	2.2	1.10	0.94	1.17	0.24
	diamond	.	.	.	.	.	.	3	2.5	0.85	0.98	0.87	0.39	3	1.7	0.85	0.98	0.87	0.39
	FaultTree	22	36.1	2.71	0.84	3.21	0.00	1	0.8	0.00	1.16	0.00	1.00	23	12.7	2.75	0.84	3.27	0.00
	Arc	21	34.4	2.66	0.84	3.15	0.00	1	0.8	0.00	1.16	0.00	1.00	22	12.2	2.71	0.84	3.21	0.00
	triangle	1	1.6	0.00	1.16	0.00	1.00	1	0.8	0.00	1.16	0.00	1.00	2	1.1	0.51	1.03	0.50	0.62
	Symbol	3	4.9	0.85	0.98	0.87	0.39	.	.	.	.	.	.	3	1.7	0.85	0.98	0.87	0.39
	line	1	1.6	0.00	1.16	0.00	1.00	.	.	.	.	.	.	1	0.6	0.00	1.16	0.00	1.00
	Total	61						120						181					
Labels	No label	.	.	.	.	.	.	3	2.5	0.85	0.98	0.87	0.39	3	1.7	0.85	0.98	0.87	0.39
	Pseudonymous	4	6.6	1.10	0.94	1.17	0.24	65	54.2	3.78	0.83	4.57	0.00	69	38.1	3.84	0.83	4.65	0.00
	Label	10	16.4	1.95	0.87	2.23	0.03	29	24.2	2.98	0.84	3.56	0.00	39	21.5	3.27	0.83	3.93	0.00
	Textual	47	77.0	3.46	0.83	4.17	0.00	23	19.2	2.75	0.84	3.27	0.00	70	38.7	3.85	0.83	4.67	0.00
	Total							120						181					

†Rounded rectangle
σ = standard deviation

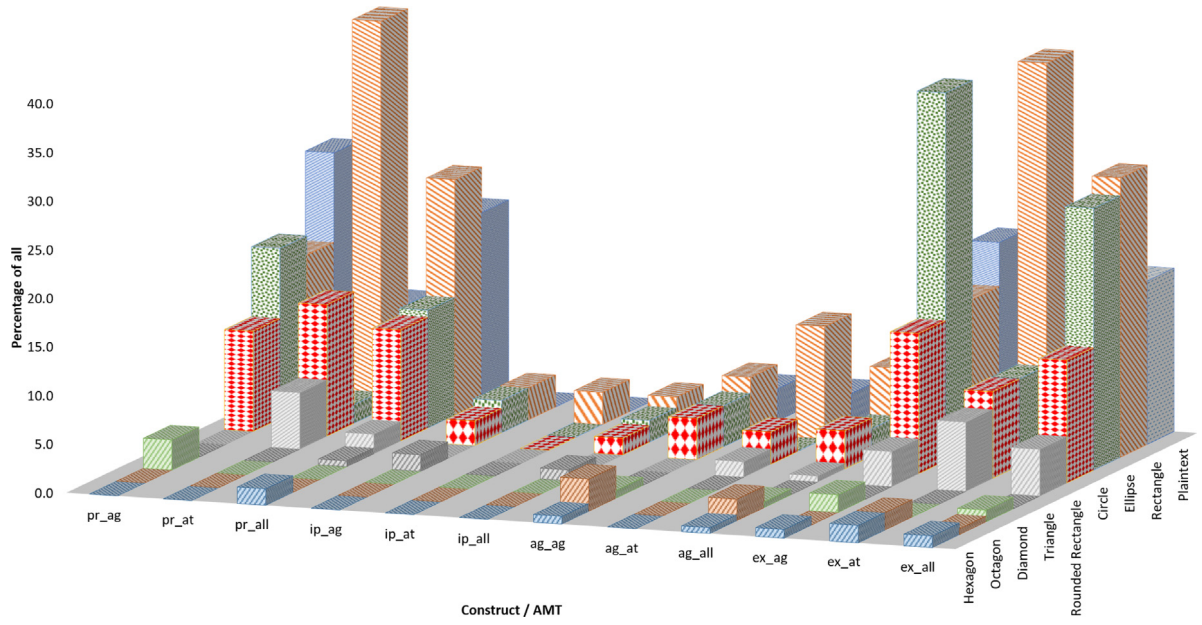

Fig. 23. AMT shapes count.

rounded rectangle to represent a precondition and exploit respectively. Sen and Madria [292] (Fig. 20, bottom) utilise a thick edge to differentiate exploits from preconditions. In all three of these examples, there is an imperceptible visual distance which makes it difficult to discern between the shapes.

6.4.1. Colour in AMTs

Section 5.7.3 introduced the use of colour in AMTs. The discussion therein outlined that although there are benefits to the use of colour, there are numerous considerations that also have to be borne when designing colour based visual syntax. One of these is that colour should be applied effectively, i.e., there should be a cognitive reason for the use of a particular colour. Relevant cognitive reasons could include to increase the visual distance and make constructs more discernible, or to add emphasis.

Table 16 presents a detailed overview of the use of colour in preconditions and exploits, and Tables 22 and 23 provide a comprehensive overview of the decorative colour applied to each attack graph and attack tree respectively with accompanying citations and references to the original attack graph/attack tree representations in the Appendix (see Fig. 21).

78 (43.1%) AMTs utilised colour to either shade the shape or the line. 46 (*25.4%*, $\beta = 3.43, z = 4.14, p = 0.00$) exclusively coloured the shape and 32 (*17.7%*, $\beta = 3.08, z = 3.68, p = 0.00$) exclusively coloured the line. 103 (56.9%) did not use colour in any way ($\beta = 4.23, z = 5.15, p = 0.00$).

The results show that although authors generally prefer not to use colour, the use of colour to alter shapes and lines is common and statistically significant. For example, although there is a general preference not to use colour in attack trees ($n = 39$, 63.9%, $\beta = 3.27, z = 3.93, p = 0.00$), the use of coloured shapes ($n = 14$, 23.0%, $\beta = 2.27, z = 2.65p = 0.01$) and coloured lines ($n = 8$, 13.1%, $\beta = 1.74, z = 1.96p = 0.05$) was common and statistically significant.

Similarly, the use of coloured shapes ($n = 32$, 26.7%, $\beta = 3.08, z = 3.68p = 0.00$) and coloured lines ($n = 24$, 20.0%, $\beta = 2.79, z = 3.32p = 0.00$), was common and statistically significant.

4 of the 56 attack graphs presented the whole graph in a particular colour. In these examples, colour was not used for any perceivable cognitive reason. It was not used to increase the visual distance or to add emphasis as all the shapes were grey [46] or turquoise [56,156,221].

Table 16
Use of colour in preconditions and exploits.

	Ellipse	e_{bu}	e_{gn}	e_{rd}	e_{gy}	e_{tq}	e_{ye}	e_{vi}	e_{wh}
Precondition	AG	1	0	0	1	0	1	0	0
	AT	0	1	0	0	0	1	0	0
Exploit	AG	3	1	1	2	2	0	0	1
	AT	0	1	0	0	0	0	0	0
	Rectangle	r_{bu}	r_{gn}	r_{gy}	r_{og}	r_{rd}	r_{bk}	r_{vi}	r_{ye}
Precondition	AG	0	0	2	0	2	0	1	2
	AT	0	1	2	0	0	0	0	0
Exploit	AG	2	1	3	0	0	0	0	3
	AT	2	1	4	1	0	0	0	0
	Circle	c_{bk}	c_{bu}	c_{gn}	c_{gy}	c_{og}	c_{rd}	c_{wh}	c_{pk}	c_{ye}
Precondition	AG	0	2	3	1	1	3	0	0	1
	AT	0	2	0	1	0	1	0	0	0
Exploit	AG	2	1	1	2	0	1	1	2	0
	AT	0	0	0	1	0	1	0	0	0

Colour codes used in the descriptions are according to [418] corresponding to: *vi*:violet, *bu*:blue, *gn*:green, *og*:orange, *bk*:black, *ye*:yellow, *pk*:pink, *gy*:grey, *tq*:turquoise, *rd*:red, *wh*:white, *bn*:brown.

Nevertheless, a handful of attack graphs and attack trees use colour to increase the visual distance between objects — for example to represent elements such as: initial preconditions [35, 271,274,279], and exploits [41,177].

Table 16 reveals considerable variance in the way coloured edges are used. Taking the representation of preconditions using a circle as a case in point, it can be observed that whilst 11 of the 22 representations apply a clear circle (with black edges), the remaining 11 are variations which — for example, apply a green circle [175], blue circle [279], red circle [60], grey circle [193], yellow circle [274], green circle [77] and an orange circle [77].

6.4.2. Shape edge texture

Alterations in line style, colour or density are examples of how shape edge texture can be adjusted to increase the visual distance between objects to make them more perceptible. 14 (11.9%) attack graphs and 17 (27.9%) attack trees utilised texture

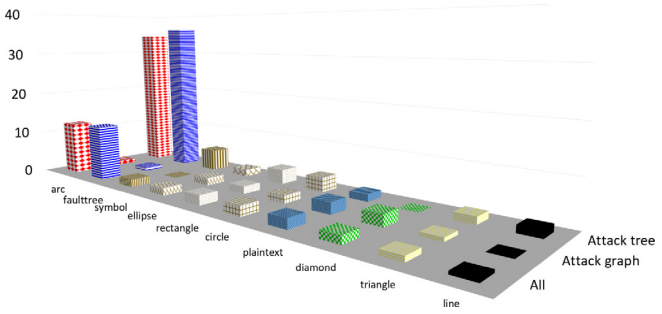

Fig. 24. Shapes used to represent precondition logic.

Table 17
Representation of shape edge texture.

Texture	Description and citation
Edge texture	Dotted rectangle represents precondition [148]; dotted edge represents an exploit [237,386]
Double/triple edges	Doubled edged rectangles represent an attack path [152,270]; double edged circle represents a goal: [159]; double thick edge represents a goal: [225]; double edged ellipse represents a goal [278]; double edged circle represents a goal [37,272]; triple edged hexagon represents a goal [62,313]; thick lined circle represents a goal [219]
Coloured edges	Red and green edges represent a precondition and exploit respectively [97]; red edges represent an attack path [305]
Mixed	Pink with dotted black edge represents an exploit [285]

Table 18
Representing initial preconditions in attack graphs and attack trees.

AMT	Shape	Citation
Attack graph	Rectangle	[88,289,385,392]; *yellow rectangle:* [167]
	Circle	*Black circle* [271]; *grey circle:* [274]; *pink circle:* [290]
	Ellipse	*grey ellipse:* [31,231]; *violet ellipse:* [55]; *orange ellipse:* [226]
	Triangle	[46]; *grey triangle:* [281]
Attack tree	Rectangle	[412]; *grey rectangle*: [232]

in some form or other, these are highlighted in Table 17, and some examples are provided herein.

Figs. 22 and 23 highlight the approaches towards using line texture in attack graphs and attack trees. Tables 22 and 23 provide a comprehensive overview of the shape edge texture applied to each attack graph and attack tree respectively with accompanying citations and references to the original attack graph/attack tree representations in the appendices.

The data shows that 143 of the 181 attack graphs and attack trees (*79%*, $\beta = 4.56$) analysed did not utilise shape edge texture ($z = 5.56, p = 0.00$), whereas 38 (*21.0%*, $\beta = 3.25$) did with a smaller effect size ($z = 3.90, p = 0.00$). This was analysed further to reveal that 99 of the 120 attack graphs (*82.5%*, $\beta = 4.20$) and 44 of the 61 attack trees (*72.1%*, $\beta = 3.39$) did not utilise line density. In both cases, the effect size was large indicating a preference not to utilise shape edge texture ($z = 4.20, p = 0.00$ compared to $z = 2.66, p = 0.00$, and $z = 4.09, p = 0.00$ compared to $z = 2.89, p = 0.00$ respectively). Examples that utilise shape edge texture are highlighted in Table 17, and some examples are provided herein.

Preconditions and *exploits* have been represented using a dotted edge to represent the conjunction of two preconditions [237], to represent exploits [285], to represent *vulnerability nodes*, or a red/green edge to represent no preconditions and exploits respectively [97].

Entire *attack paths* have been represented with double edged rectangles which mark the nodes in the attack path [152] or red edges [305].

The use of line density in attack trees is somewhat similar. Line density has been used to represent *attack paths* [203], *attack goals* [68,92,419] and *mitigations* [99,214,239,243,403].

Table 19
Analysis of exploit representation in attack trees.

Shape	Citation
Circle	[99,256,403,407]; *red circle:* [243]
Ellipse	[68,410] *green ellipse:* [398]
Hexagon	[419]
Plaintext	[23,23,130,203,401,411]
Rectangle	[90,92,103,180,235,239,263,263,264,351,393–396,399,400,404–406,408,412,413,415,417,420] *blue rectangle:* [233,234] *rounded rectangle:* [87,230,397] *green rectangle:* [214] *grey rectangle:* [421] [93] [402] [422] *orange rectangle:* [187]
Octagon	*Turqoise octagon:* [232]
Mixed	*Circle and rectangle:* [69] [416] *green hexagon and red hexagon:* [409] *grey circle and grey triangle:* [322] *rectangle and ellipse:* [423] *rectangle, rounded rectangle, and green rectangle:* [414]

Table 20
Representation of precondition logic in attack graphs and attack trees.

Shape	Citation
Attack graphs	**Arc** [149]; Circle [297], *Black circle* - [215]; **Diamond** [65,66,173,276]; **Ellipse** *Ellipse with the word 'and' inside* [292,298]. *Ellipse represents 'or', diamond represents 'and'* [65,66,148]; **Fault tree symbols** *Fault tree symbols turned upside down and coloured black:* [291]; **Plaintext** [146–148,286,301]; **Triangle The word 'and' added to a triangle** [107]
Attack Tree	**Arc:** [87,99,103,130,214,243,322,387,388,396,398,401–403,420]; *Arc with the word 'or'. Absence of arc represents 'and'*: [417]; *Double arc is an 'and'*: [393]; *Arc with the word 'and' and a double arc with the word 'or'*: [412]; *Arc – word 'and' added to the arc – the absence of an arc represents an 'or'*: [90,180,239]
	Fault tree symbols: [23,69,93,234,235,256,263,399,407,408,416,419,422–424]; *The words 'and' and 'or' added the fault tree shapes*: [23,230,389–391,400]; *The words 'and/or' added to the side of the fault tree symbol*: [263]; *The word 'or' added to a rectangle which sit to the side of the fault tree symbol*: [404]; *Incorporate both an 'and' and a 'sequential and' with the latter being represented with a left–right arrow in the 'and' symbol*: [394,395]
	Miscellaneous methods. Circle. With the word 'and/or': [409]; *The word 'and' inside, absence of circle represents 'or'*: [351]; **Ellipse and rectangle** *Green ellipse and blue rectangle to represent 'and/or' respectively*: [421]; **Edge/line**. *use the edge connecting events to represent precondition operators. A solid edge represents 'and' and a dotted edge represents 'or':* [323]; **Plaintext**: [413]; **Rectangle.** *Thick edged rectangle and black rectangle represent 'and/or' respectively*: [264]; *Rectangle: 'or', rounded rectangle: 'and'*: [414]; $\wedge$ *and* $\vee$ *to represent 'and/or' respectively:* [397,406,411]; **Triangle.** *Triangle with a 'plus' symbol inside: 'and'* - [405]

6.5. The use of shapes in attack graph/attack tree visual syntax

Shapes are a central and important visual variable. Shapes are used in modelling systems to represent important concepts.

Table 21
Custom and practice in the representation of constructs in attack graphs and attack trees.

Concept	Custom and Practice
Event flow	There is a preference towards representing event flow as both top-down ($n = 106, 58.6\%, \beta = 4.26, z = 5.19p = 0.00$) and bottom-up ($n = 66, 36.5\%, \beta = 3.79, z = 4.59p = 0.00$)
Colour	There is a strong preference not to use colour to in attack graphs and attack trees ($n = 103, 56.9\%, \beta = 4.23, z = 5.15p = 0.00$). Where colour is used, it is used to colour the entire shape ($n = 46, 25.4\%, \beta = 3.43, z = 4.14p = 0.00$), or the edge of the shape ($n = 32, 17.7\%, \beta = 3.08, z = 3.68p = 0.00$). Grey, green, blue, red, yellow are the most popular colours. Colour is generally used to represent particular concepts and occasionally used to increase visual distance.
Line texture	There is a preference not to use line texture in attack graphs and attack trees ($n = 143, 79.0\%, \beta = 4.56, z = 5.56p = 0.00$). Although a number of authors have used line texture ($n = 38, 21.0\%, \beta = 3.25, z = 3.90p = 0.00$), there is no consistent approach to how colour is used in attack graphs or attack trees
Preconditions	There is a preference across attack graphs and attack trees to represent preconditions using rectangles, ($n = 36, 24.2\%, \beta = 3.19, z = 3.83p = 0.00$), plaintext ($n = 29, 19.5\%, \beta = 2.98, z = 3.56p = 0.00$), circles ($n = 17, 11.4\%, \beta = 2.46, z = 2.89p = 0.00$), and ellipses ($n = 18, 12.1\%, \beta = 2.51, z = 2.96p = 0.00$) in that order. There is also a preference not to represent preconditions ($n = 43, 28.9\%, \beta = 3.37, z = 4.06p = 0.00$).
Initial preconditions	The general preference amongst authors is not to represent initial preconditions in attack graphs or attack trees ($n = 166, 91.7\%, \beta = 4.71, z = 5.74p = 0.00$).
Perpetrator capability	Perpetrator capability is not specifically represented in any of the attack graphs or attack trees surveyed.
Attack goals	The general preference amongst authors is not to represent attack goals ($n = 142, 78.5\%, \beta = 4.55, z = 5.55p = 0.00$). However, there is a preference to represent attack goals using rectangles ($n = 14, 7.7\%, \beta = 2.27, z = 2.65p = 0.01$).
Exploits	There is a preference across attack graphs and attack trees to represent exploits using a rectangle ($n = 48, 28.7\%, \beta = 3.48, z = 4.19p = 0.00$), ellipse ($n = 45, 26.9\%, \beta = 3.41, z = 4.11p = 0.00$), plaintext ($n = 28, 16.8\%, \beta = 2.94, z = 3.52p = 0.00$), circle ($n = 21, 12.6\%, \beta = 2.66, z = 3.15p = 0.00$), and rounded rectangle ($n = 8, 4.8\%, \beta = 1.74, z = 1.96p = 0.05$) in that order. There is also a preference not to represent exploits ($n = 11, 6.6\%, \beta = 2.04, z = 2.35p = 0.02$).
Precondition logic	The representation of precondition logic is common in attack trees where the common methods of representing precondition logic is to use the fault tree method ($n = 23, 12.7\%, \beta = 2.75, z = 3.27p = 0.00$) or the arc method ($n = 22, 12.2\%, \beta = 2.71, z = 3.21p = 0.00$). Most attack graphs and attack trees do not represent precondition logic ($n = 112, 61.9\%, \beta = 4.32, z = 5.25p = 0.00$).
Labels	The common methods of representing labels in both attack graphs and attack trees is as textual ($n = 70, 38.7\%, \beta = 3.85, z = 4.67p = 0.00$), pseudonymous ($n = 69, 38.1\%, \beta = 3.84, z = 4.65p = 0.00$) and character labels ($n = 39, 21.5\%, \beta = 3.27, z = 3.93p = 0.00$).

This section analyses the shapes used to represent preconditions, exploits and precondition operators in attack graphs and attack trees. The discussion begins by providing a high level analysis of the shapes used to represent preconditions and exploits (Section 6.5.1). This is followed by an analysis of the visual syntax used to represent preconditions (Section 6.5.1 - including initial preconditions, perpetrator capability and goals), exploits () and precondition operators (Section 6.5.2). The discussion then proceeds to demonstrate the problem of using diamonds and circles to represent constructs (Section 6.5.3).

An overview of the shapes used is outlined in Fig. 23.

6.5.1. The visual representation of preconditions and exploits

14 of the authors investigated in this study represented preconditions using multiple shapes in the same paper. 6 authors represented both preconditions and exploits using multiple shapes in the same paper. These 20 papers were disqualified from the analysis leaving 141 valid configurations.

An overview of the shapes used in attack graphs and attack trees is presented in Fig. 23 and Table 15. Tables 22 and 23 provide a comprehensive overview of the shapes applied to each attack graph and attack tree respectively with accompanying citations and references to the original attack graph/attack tree representations.

The data shows that the ellipse, rectangle, circle, plaintext, diamond, hexagon, octagon and triangle have been used to represent preconditions and exploits. Of these shapes, ellipses, rectangles, circles and plaintext are the dominant forms of representing preconditions and exploits.

Preconditions. Fig. 23 and Table 15 outline the shapes used to represent preconditions.

The data shows that overall — that is including both attack graphs and attack trees in the 149 papers reviewed, the popular shapes used to represent preconditions were the rectangle ($n = 36, 24.2\%, \beta = 3.19, z = 3.83, p = 0.00$), plaintext ($n = 29, 19.5\%, \beta = 2.98, z = 3.56, p = 0.00$), ellipse ($n = 18, 12.1\%, \beta = 2.51, z = 2.96, p = 0.00$) and circle ($n = 17, 11.4\%, \beta = 2.46, z = 2.89, p = 0.00$).

This data can be broken down to analyse the use of shapes across attack graphs and attack trees separately. The data shows that attack graphs generally use plaintext ($n = 24, 24.7\%, \beta = 2.79, z = 3.32, p = 0.00$), ellipses ($n = 17, 17.5\%, \beta = 2.46, z = 2.89, p = 0.00$), rectangles ($n = 15, 15.5\%, \beta = 2.34, z = 2.73, p = 0.01$), and circles ($n = 10, 10.3\%\beta = 1.95, z = 2.23, p = 0.03$) to represent preconditions. 28 ($28.5\%, \beta = 2.94, z = 3.52, p = 0.00$) attack graphs did not represent preconditions. Attack trees favour rectangles ($= 21, 40.4\%, \beta = 2.66, z = 3.15, p = 0.00$) and circles ($n = 7, 13.5\%, \beta = 1.61, z = 1.80$). 15 ($28.8\%, \beta = 2.34, z = 2.73, p = 0.01$) attack trees did not represent preconditions.

Initial Preconditions

The data provided in Fig. 23 and Table 15 outlines the representation of initial preconditions in attack graphs and attack trees. Table 18 provides further insights into specific representations.

The data shows that overall, it is not common to represent initial preconditions (*n=166, 91.7%,* $\beta = 4.71, z = 5.74$). When analysed across attack graphs and attack trees, the data shows that 13 (10.8%) of the attack graph configurations and 2 (3.3%) attack tree configurations represented initial preconditions. The method of representing initial preconditions was diverse and there was not enough data to determine custom and practice. Given that the data shows that it is uncommon to represent

initial preconditions, the corresponding narratives were explored to understand the reasoning behind why:

1. The author considered it important or unimportant that initial preconditions be expressed specifically.
2. The reason for representing an initial precondition in a manner different to the precondition.

In either case, there was no explanation.

Perpetrator Capability

Although perpetrator capability can be an important consideration in an attack model [141], none of the attack models considered in this survey distinctly presented this within the configuration. This may be because perpetrator capability is a secondary construct, and many researchers choose to represent it as a primary construct in the form of a precondition. Examples of this representation include Wang et al. [425] who represent the availability of the `ftp` service and the possession of `trust` privileges using plaintext, and Wang and Jajodia [426] (Fig. 10) who represent the availability of the `sadmind` service (a precondition) and `user` privilege (a perpetrator capability) using ellipses.

Attack Goals

The final postcondition in an attack is the goal of the attack. The data provided in Table 15 outlines the representation of attack goals in attack graphs and attack trees.

39 (21.5%) of the attack graphs and attack trees specifically represented attack goals. Of these, 27 were attack graphs and 10 were attack trees. Notably, 142 (*78.5%*, $\beta = 4.55$) attack graphs/attack trees did not represent attack goals. The corresponding effect size ($z = 5.55, p = 0.00$) indicated that representation of attack goals was not common. Notwithstanding, the representation of attack goals using rectangles ($n = 14$, 7.7%, $\beta = 2.27, z = 2.65p = 0.01$) is statistically significant although the effect size is small.

Here again, the corresponding narratives were explored to understand the reasoning behind why:

1. The author considered it important or unimportant that attack goals be expressed specifically.
2. The reason for representing an attack goals in a manner different to the precondition.

In either case, there was no explanation.

Exploits. The data provided in Fig. 23 and Table 15 outlines the representation of exploits in attack graphs and attack trees. Table 19 provide a comprehensive overview of the precise exploit configurations of each attack graph and attack tree respectively.

The data shows that overall — that is including both attack graphs and attack trees, the popular shapes used to represent exploits were rectangle ($n = 48$, 28.7%, $\beta = 3.48, z = 4.19, p = 0.00$), ellipse ($n = 45$, 26.9%, $\beta = 3.41, z = 4.11, p = 0.00$), plaintext ($n = 28$, 16.8%, $\beta = 2.94, z = 3.52, p = 0.00$), circle ($n = 21$, 12.6%, $\beta = 2.66, z = 3.15, p = 0.00$) and rounded rectangle ($n = 8$, 4.8%, $\beta = 1.74, z = 1.96, p = 0.05$).

When broken down to analyse the use of shapes across attack graphs and attack trees separately, the data shows that the most popular forms of representing exploits in attack graphs were ellipse ($n = 42$, 37.8%, $\beta = 3.34, z = 4.03, p = 0.00$), plaintext ($n = 22$, 19.8%, $\beta = 2.71, z = 3.21, p = 0.00$), rectangle ($n = 17$, 15.3%, $\beta = 2.46, z = 2.89, p = 0.00$) and circle ($n = 16$, 14.4%, $\beta = 2.40, z = 2.81, p = 0.01$). In the attack tree representations, the only shape to render a statistically significant result was the rectangle ($n = 31$, 55.4%, $\beta = 3.05, z = 3.64, p = 0.00$).

The results show that although the overall results suggest that the ellipse, rectangle, circle, plaintext and rounded rectangle are popular forms of representing exploits, the ellipse, circle and plaintext are popular for the attack graph but not the attack tree. The rounded rectangle — whilst statistically significant overall, is not so for either the attack graph or attack tree.

6.5.2. Representing precondition logic

The discussion in Section 3.2 outlined the importance of presenting precondition logic. Data relating to how precondition logic is represented in attack graphs and attack trees is presented in Table 15, Table 20 and Fig. 24. Raw data relating to the use of visual syntax to represent precondition logic in attack graphs and attack trees is provided in Table 20.

112 (61.9%, $\beta = 4.32, z = 5.25, p = 0.00$) attack graphs/attack trees did not represent precondition logic. The fault tree ($n = 23$, 12.7%, $\beta = 2.75, z = 3.27, p = 0.00$) and arc ($n = 22$, 12.2%, $\beta = 2.71, z = 3.21, p = 0.00$) were the most popular form of representing precondition logic.

When the attack graph and attack tree are analysed separately, one can see that this is subject to Simpson's paradox. 15 (12.5%) attack graphs presented precondition logic, 105 (*87.5%*, $\beta = 4.25$) did not. The effect size ($z = 5.17, p = 0.00$) indicates that it is not common to represent precondition logic in attack graphs. Furthermore, although the ellipse, rectangle, circle, plaintext, fault tree symbol, arc and triangle have been used to represent precondition logic, these results are not statistically significant and the effect size is very small (ranging from $z = 0$ to $z = 0.87$).

The representation of precondition logic is much more common in the attack tree literature. 54 (89.0%) attack trees presented precondition logic. The fault tree ($n = 22$, 36.1%, $\beta = 2.71, z = 3.21, p = 0.00$) and the arc ($n = 21$, 34.4%, $\beta = 2.66, z = 3.15, p = 0.00$) were the most popular form of representing precondition logic.

6.5.3. The problem with diamonds and circles

The data presented in Fig. 23 and Table 15 demonstrates that circles are a popular method of representing preconditions and exploits, but diamonds, hexagons, octagons and triangles are not. Diamonds, hexagons, octagons and triangles are probably unpopular because the use of these shapes compromises the size of the textual label that can be added to the shape. However, despite their popularity, circles are not suitable for representing textual labels.

6.5.4. Circles

Gadyatskaya et al. [398] uses a combination of circles and ellipses to represent preconditions and exploits. In this example, it appears that the width is being adjusted to accommodate the label being presented.

In an example by Ralston et al. [234],[3] the text is hanging over the edges of some of the circles. Another example of this is provided by Patel et al. [233] who present textual labels within circles — here again the text is hanging over the edges of some of the circles.

6.5.5. Diamonds

5 of the AMTs that used a diamond combined it with a pseudonymous label and 'stretched' the diamond to enable the label to fit [66,157,276,300,305].

Generally, the combination of circles/diamonds with character or pseudonymous labels is not a problem [30,65,219,282,307],

[3] Tanu and Arreymbi presents the same attack graph in their paper — with the same results [235].

Table 22
A review of attack graph visual syntax. *This table is presented generally in alphabetic order of author surname. However, some entries are not in alphabetic order and have been swapped to ensure the page space is used efficiently.*

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	notes
Attack graph	[156]	td	X	X	p	e	X	✓*	ps	X	* all nodes are turqoise
Logical attack graph	[66]	td	X	e|d*	r	d |e	X	✓	ps	X	*Ellipses represent AND, diamonds represent OR, however the same node represents preconditions as represents exploits.
Attack graph	[67]	td	X	X	✓	$r r_{bu}$ $\|e_{rd}$	r_{og}	✓	tx	✓	
Attack graph*	[55]	td	e_{vi}	X	e_{bu}	r_{gn}	X	✓	ps	X	*potential mitigations represented as orange rectangles
State enumeration attack graph	[142]	td	X	X	e	e*	X	X	ch	X	*ellipses are used to represent both exploits and preconditions, the distinction is made with exploits represented as lower case text and preconditions as uppercase
Attack graph*	[65]	td	X	d|e†	u	u	X	X	ch	X	*The terms attack graph and attack tree are used interchangeably, in particular when referring to the figure †(∧), d(∨)
Attack graph	[178]	lr	X	X	e*	p_{ie}	X	✓	ps	X	*grey represents states visible to an IDS
Attack graph	[150]	td	X	X	p	e	X	X	ps	X	
Attack graph	[144]	td	X	X	e	r	X	X	ps	X	Note the difference in visual syntax of the two attack graphs proposed in [150]and [144]
Attack graph	[288]	bu	X	X	e_{ye}	r_{bu}	X	✓	ps	X	
Attack graph	[193]	lr	X	X	c |c_{gy}	p_{ie}	X	✓	ps	X	
Attack graph	[219]	td	X	X	p	e	c*	X	tx	✓	*thick lined circle
Attack graph	[219]	td	X	X	r	X	X	X	ps	✓	
Attack graph	[289]	td	r	X	r	e	p	X	ps	X	
Coordinated attack graph	[42]	td	X	X	c*	p_{ie}	X	X	tx|ch†	X	*represented as statuses †text:exploits, character:preconditions
Vulnerability cause graph (second generation)*	[148]	td	X	r|h†	r |h§	h	X	X	tx	✓	*unclear use of hexagons and rectangles; †dotted rectangle encompasses precondition nodes. Does not appear to distinguish between conjunction/disjunction; §precondition is represented by both the rectangle ('simple node') and the hexagon ('compound node')
Vulnerability cause graph	[315]	td	X	X	r|rr |h*	rr	X	X	tx	X	*rectangles, rounded rectangles and hexagons are used interchangeably, however these are not clearly explained
Attack graph	[166,216]	td	X	X	p	e	X	X	ps	X	
Attack graph	[222]	td	X	X	p	e	X	X	ps	X	
Exploit dependency graph	[97]	td	X	X	e*	e †	X	✓	ps	✓	*red edge, †green edge
Privilege graph	[223]	td	X	X	p_{ie}	X	c	X	ch	X	
Dependency attack graph	[392]	td	X	X	p|r*	e	r_{gn}	y†	ps	X	*Privileges given as rectangles and other preconditions as plaintext †goal given as a grey rectangle
Risk flow attack graph	[392]	td	r	X	p	p	r_{gy}	r_{gy}	ps	X	
Attack graph	[85]	td	X	X	X	r	X	X	tx	X	
Attack graph	[220]	td	X	X	r	rr	d	X	tx	X	
Attack graph	[224]	td	X	X	p	e	X	X	ps	X	
Attack graph	[224]	bu	X	X	p	e	X	✓	ps	X	Note: two attack graph configurations are proposed in the same paper
Dependency Graph	[28]	td	X	X	e	p_{ie}	X	X	ps	X	
i-graph	[149]	td	X	arc	e	e	X	X	tx	X	*introduces 'quorum operators as a double arc as well as the normal conjunction/disjunction "Minimum Required Quorum (MRQ) on it, which represents the minimum number of child nodes whose goals need to be achieved in order for the node with incoming Quorum edges to be achieved"
Attack graph	[46]	td	t	X	r	e	o	✓†	ps	X	*given as a downward facing triangle, this image is also used in [281] †all shapes are grey
Attack graph	[161]	td	X	X	r*	e	X	✓†	tx	X	*A distinction is made between a threat and a vulnerability by the use of an 'angry face' and ladybird icon respectively † attacks presented with a red edge

(*continued on next page*)

Table 22 (*continued*).

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	notes
Attack graph	[157]	td	X	X	X	e|d	e|d	X	X	ps	
Attack graph	[221]	td	X	X	p	e*	X	y†	ps	X	*In a separate paper [56] present exploits as circles and as ellipses in two separate attack graphs; †all shapes presented in turqoise
Attack graph	[56]	td	X	X	p	e*	X	✓†	ps	X	*In the same paper, Ghosh and Ghosh present exploits as circles †all shapes presented in turqoise
Attack graph	[225]	td	X	X	e	p_{ie}	y*	X	ps	✓	*goal represented as an ellipse with a double thick edge
Attack graph	[30]	lr	X	X	d	e	X	X	ch	X	
Attack graph	[226]	td	e_{og}	X	e	r	h_{gn}	✓	ps	X	
Multiple prerequisite graph	[34]	td	X	X	c|r*	t†	X	X	ps	X	*circles represent states, rectangles represent preconditions. †downward pointing triangle
Attack graph	[34]	td	X	X	c	p_{ie}	X	X	ps	X	
Attack graph	[290]	td	c_{pk}	X	p	r	c_{pk}*	✓	ps	✓	* double edged circle
Attack graph	[41,52,168]*	td	X	X	X	r	X	X	ps	X	Three different attack graphs are presented in the same paper and one is repeated in the paper by Sheyner et al.. These are counted as three separate attack graphs *Sheyner et al. is co-authored by Jha et al. †in-edge labels
		td	X	X	X	c_{bk} †	X	✓	ps	X	
		td	X	X	X	c*	X	X	ps	X	
Attack graph	[45]	td	X	X	p	e_{gn}	X	✓	ps	X	
Attack graph*	[167]	td	r_{ye}	X	r_{ye}	e_{bu}	o_{pk} †	✓	ps	X	*On page 36, Jajodia and Noel present the attack graph previously presented by [46] in a colorised version. The original is not cited. †divided into overall postconditions represented as a pink hexagon or intermediate postconditions represented in plaintext
Attack graph	[143]	td	X	X	p	e	X	X	ps	X	
Probabilistic attack graph	[71]	td	X	X	r_{tq}	e_{tq}	X	✓	ps	X	
Attack graph	[107]	td	X	t_{gy}*	r_{rd}	r_{ye}	X	✓	ps|tx	X	*$AND_{greytriangle}$, OR represented (one presumes) by absence thereof
Compact attack graph	[427]	td	X	X	p	rr	X	X	ps	X	[427] is similar to [150]. Note: The *attack graph* and *compact attack graph* proposed in [427] and [283] are essentially the same
Attack graph	[283]	td	X	X	p	rr	X	X	ps	X	
Attack graph	[60]	td	X	X	c_{rd}	r_{gy}|r_{wh}*	X	✓	ps	X	*white and grey rectangles are used to represent exploits. White rectangles represent perpetrator action, grey rectangles represent final perpetrator action. A clear circle is used to represent hosts. NB. The editors and reviewers of 'Communications and Multimedia Security' appear to have missed the six profanities in the attack graph....
Attack graph	[284]	td	X	X	u*	u*	u*	✓	ps	X	*four colours and two shapes (rr, r) are used but there is no explanation outlining the meaning of the colours/shapes
Logical attack graph*	[306]	td	t	X	d	e	r†	X	ps	X	*Near identical graph to [307]
Access execution graph	[77]	td	X	X	c_{gn}|r_{vi}|c_{og}	r_{ye}	X	✓	tx*	X	*test used for certain prerequisites but not for anything represented using a circle
Predictive Attack graph	[271]	lr*	X	X	X	c	X	X	ch	X	*The predictive attack graph is presented both left–right and top-down in the same paper
		lr	X	X	X	c_{bk}	X	X	x	X	
		td	X	X	X	c	X	X	ch	X	
E-graph	[88,385]	td	r	X	X	r	r	X	ps	X	
Attack graph	[50]	td	X	X	e	e	X	✓*	tx	X	*The example highlights two nodes in grey
Attack graph	[50]	td	X	X	c_{gn}	p_{ie}	X	✓	ps	X	

(*continued on next page*)

Table 22 (*continued*).

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	notes
Attack graph	[271]	td	c_{bk}	X	X	c_{gy}	X	✓	ps	X	An attack graph is presented in each of the papers by [61,271], both of them are different. The attack graph presented in [271] is presented both top-down and left–right
Attack graph	[61]	td	X	X	X	c	X	X	ps	X	
Attack graph	[37,272]	td	X	X	X	e	✓*	X	ps	✓*	*Double edged circle represents goal
Compressed attack graph	[163]	td	X	X	u	e	X	X	ps	X	
Attack graph	[291]	td	X	ft	c	c	X	✓*	tx	X	* all edges are blue
Attack graph	[280]	td	X	X	p	p	X	X	ps	X	
Attack response graph	[179]	td	X	X	X	c	X	X	tx*	X	*textual explanation added at the side of each node
Attack graph	[35]	td	X	X	$c_{bu}\|c_{rd}$*	p_{ie}	X	✓	tx†	X	*blue is normally privilege level, red is root privilege. †exploit described as *ps*
Attack graph	[177]	td	X	X	X	$c_{wh}\|c_{pk}\|c_{rd}$*	X	✓*	ch	X	*colour used to denote severity of exploit
Personalised attack graph	[298]	bu	X	e*	r_{gy}	r	r†	✓	tx	✓	*with the word 'and' inside †bold blue edge
Attack graph	[159]	td	X	X	X	c	c*	X	ch	✓	*double edged circle
Attack graph	[285]	td	X	X	c*	$c_{bu}\|c_{pk}$ †	c_{gn}	✓	ch	✓	*pink with dotted black edge. †utilises blue nodes referred to as transition nodes
Hybrid attack graph	[428]	td	X	X	e	p_{ie}	X	X	ch	X	
Attack strategy graph	[32]	td	X	X	X	e*	X	X	tx	X	*source and destination ip addresses presented as in edge labels
Alert correlation graph	[33]	td	X	X	X	X	X	✓	ps	X	
Hyper alert correlation graph	[31]	lr	e_{gy}	X	X	e	X	✓	ch	X	
	[32,33,57]	lr	X	X	X	e	X	X	ch	X	
Integrated correlation graph	[33]	lr	X	X	X	$e_{wh}\|e_{gy}$	X	✓	ps	X	
Exploit dependency graph*	[62,313]	td	X	X	p	e	o†	X	ps§	✓	
Attack graph*	[231,281]	td	X	X	X	e_{tq}	X	✓	ps	X	*The *exploit dependency graph* presented in [231] is a compact version of the *state enumeration graph*
Condition oriented attack graph*	[231]	td	X	X	r_{gy}	p_{ie}	X	✓	ps	X	
Exploit oriented attack graph*	[231]	td	e_{gy}	X	e_{gy}	p_{ie}	x	✓	ps	X	
Attack graph (low Level)	[273]	td	X	u	e	r	X	✓	ps	X	
Attack graph	[169]	td	X	X	r_{pk}	e_{bu}	o_{pk}	✓	ps	X	*one of the preconditions: `execute(web)` appears to have been misrepresented without a shape or colour
Attack graph	[281]	td	t_g*	X	p|rye	e_{bu}	e_{ye}	✓	ps	X	*green downwards facing triangle
Privilege graph	[311]	td	X	X	c|e|p_{ie}*	X*	X	X	ch	X	*in edge labels represent the preconditions which are explained in the caption, circles and ellipses represent the state of the host after an exploit. Although exploits are not represented, it is assumed that they have been executed.
Attack state graph	[311]	td	X	X	p_{ie}*	X	X	✓	ch	X	*The explanation of the graph is insufficient and the observer is left to work out that the in-edge labels represented in the *attack state graph*, correspond to the in-edge labels presented in the *privilege graph* (Presented in the same paper). Clear and black ellipses are used to represent the states after an exploit has been applied. Here again, there is no explanation

(*continued on next page*)

Table 22 (*continued*).

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	notes
Logical attack graph	[215]	td	X	c_{bk}	c*	r†	X	X	ch	X	*referred to as a 'fact node'. There are two types of fact node: a 'primitive fact node' (small black circle), and a derived 'fact node' (a circle with a number in it). *represented as a 'derivation node' which is explained as "how the fact becomes true", i.e., how a possible vulnerability is realised".
Attack graph	[282]	td	X	X	e|d*	e|d*	X	X	ch	X	*Each preceding number in the attack graph corresponds to a key also provided in the paper. The numbers 1, 2, 6, 7, 11 and 12 correspond to exploits, the rest correspond to preconditions. The figure shows that diamonds and ellipses are used to represent exploits and preconditions interchangeably
Host access graph	[175]	td	X	X	c_{gn}	c_{gn}	X	✓	ch	✓	
Logical attack graph*	[307]	td	X	X	d	e	r†	X	ps	X	*Near identical graph to [306] †The graph includes 'configuration nodes' represented as rectangles. These are presumably (although not explained) the postconditions when node configurations that are changes as a consequence of an exploit
Attack graph	[173,276]	td	X	d|e	r*	d*	X	✓	ps	X	*coloured red to blue according to AssetRank value †d(∨) e(∧)
Attack graph	[292]	bu	X	e*	r	r	X	X	tx	✓†	*Ellipse with text † specific exploits outlined with double density line
Bayesian attack graph	[301]	td	X	r_{vi}	X	e	X	X	tx	X	
Attack graph	[79]	td	X	X	e	p_{ie}	x	X	tx	X	
Attack graph	[152,270]	td	X	X	X	c_{gy}*	X	✓	ch	✓	*shaded grey circles used to outline IDS alarms Red rectangles are used in examples to highlight attack paths, these are not part of the definition
Defense graph	[147]	bu	X	p|d	e	e	✓	✓	tx	X	
Attack graph	[293]	td	X	X	c	c	X	X	ch	X	
Mission dependency graph	[300]	td	X	X	r|d	e	X	X	ps	X	
Alert correlation graph	[54]	td	X	X	r	e	X	X	tx	X	
Attack graph	[146]	bu	X	p(∨)	p	p	X	X	tx	X	
Personalised attack graph	[297]	bu	X	c*	r†	r_{gy}	X	X	tx	✓	* conjunction represented with a circle — not labelled †solid and dashed line rectangles
Attack graph*	[275]	td	X	X	r	e	X	✓†	tx	X	*Although referred to as an attack graph, other authors have referred to a graph representing a similar concept as an alert correlation graph; †grey shapes used to highlight missing alerts
Attack graph	[277,425]	td	X	X	p	e	e_{gy}	✓	ps	X	
Attack graph	[265]	td	X	X	p	e	X	X	ps	X	
Multiple prerequisite graph	[429]	td	X	X	u	u	X	✓	ps	X	Multiple colours are used in the graph, the narrative does not reveal the meanings of the colours and it is not possible to discern preconditions and exploits
Attack graph	[286]	lr	X	p	c	c	X	X	ch	X	
Host attack graph	[237,386]	td	X	X	X	p_{ie}*	X	X	ps	✓	*exploits are represented with a dotted edge ellipses are used to identify hosts — hence the name
Sub attack graph	[386]	td	X	X	e*	p_{ie}†	X	X	ch	X	*represented as states, the specific state is not identified †an in edge label is provided, the label has to be correlated with a table (also provided)

(*continued on next page*)

Table 22 (*continued*).

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	notes
Attack graph	[278]	td	X	X	X	p_{ie}*	e†	✓	ch	✓	*exploit is presented as a label. A separate table is provided to aid the observer †doubled edged ellipse
Attack graph	[274]	td	c_{gy}*	X	c_{ye}|c_{rd} †	p_{ie}	X	✓	ch	X	*grey†red nodes represent root privilege, yellow nodes represent user privilege
Logical attack graph	[305]	td	X	X	e	d	X	✓*	ch	✓	*red edges as well as black edges
Attack graph	[287]	td	X	X	p	e	X	X	ps	✓	
Bayesian attack graph	[430]	td*	X	X	p	e |egy	X	✓	ch	X	* The same paper also presents the attack graph as left to right
Protocol vulnerability correlation graph	[431]	td	X	X	c*	X	X	✓	ch	✓	*light green edge
Attack graph	[294]	td	X	X	p	c	X	X	ch	X	
Attack graph	[279]	td	X	X	c_{bu}*	r_{bu} †	X	✓	ch	X	*represented as a target host against which an attack has been successful †exploits are represented as free standing rectangular boxes in which the exploit description is inserted
Attack graph	[155]	td	X	X	X	e	X	X	tx	X	

Key: **Column descriptors**. *ef*:event flow, *ipc*:initial precondition, *po*:precondition operator, *pr*:precondition, *ex*:exploit, *go*:goal, *col*:colour, *lb*:label type, *edg*:edge type
Colour codes used in accordance with the definitions provided by [418]. *bu*:blue, *bk*:black, *gn*:green, *gy*:grey, *og*:orange, *pk*:pink, *tq*:turquoise, *rd*:red, *vi*:violet, *wh*:white, *ye*:yellow
Shapes. *c*:circle, *r*:rectangle, *p*:plaintext, *e*:ellipse, *d*:diamond, *rr*:rounded rectangle, *h*:hexagon, *o*:octagon, *t*:triangle, *u*:unclear
Precondition operator. *ftr*:fault tree method, *sym*:symbol, *dl*:dotted line, ∧: AND; ∨:OR
Event flow. *td*:top-down, *bu*:bottom-up, *lr*:left–right, *rl*:right–left
Labels. *ps*:pseudonymous, *ch*:character, *tx*:textual, *ie*:in edge (label presented on the edge)

and the present review found only one instance where a textual label fits a circle without the need to modify the circle size [147].

This demonstrates that diamonds and circles — whilst suited to accommodating character labels, are not suited to accommodating pseudonymous or textual labels. The shape has to be unacceptably modified to accommodate the label. This is not a problem unique to attack graphs. In a Riskit diagrams presented by Kontio and Basili, circles are used to represent processes [200]. Here again there are numerous examples where the textual label does not fit into the circle. In most cases, the text is allowed to bleed over the edge.[4]

7. Discussion

This paper provided an overview of cyber-attack theory and emphasised the primary constructs (exploit, precondition and precondition logic) and secondary constructs (initial precondition, perpetrator capability, postcondition and goal). These primary constructs represent the basic elements of a cyber-attack and are considered to be the concepts that should be represented in an AMT.

A key argument in this paper has been that although the research shows a clear divide between papers that focus on attack graphs and those that focus on attack trees, both attack graphs and attack trees are essentially a graph based structure with the main differences being the manner in which event flow is represented, the representation of preconditions, and representation of full and partial attacks. This is an important argument in the context of the present study and has rarely been expressed before. This premise enables similar studies to generalise the problem domain to include both attack graphs and attack trees.

One of the central arguments of this paper is that the visual syntax of attack models such as attack graphs and attack trees is not scientifically designed, and that researchers use self-nominated visual syntax to model the attack. This is referred to by Alexander [131] as the *unselfconscious design approach* and manifests itself in some graph based AMTs in three ways:

1. Failure to standardise.
2. Ineffective design.
3. An assumption of cognitive effectiveness.

7.1. A failure to standardise

The visual syntax of modelling systems such as fault trees and Petri nets is standardised [127,128] - resulting in a common understanding of the visual syntax used to represent elements of the respective framework.

Attack graphs and attack trees which suffer from a lack of standardisation and a distinct lack of prescriptive methodologies and common approaches in terms of their visual syntax [129] which leads to the problem of *"ambiguous semantics"* [130]. This paper (Section 6) showed that there are more than seventy five attack graph visual syntax configurations and more than twenty attack tree configurations.

The visual syntax in these examples differ in terms of the shapes used to represent constructs such as preconditions and exploits. The availability of numerous opposing proposals can give rise to confusion for researchers and practitioners in deciding which to use and is evidence of an *"immature research field"* resulting in a *"fragmentation of research efforts"* [132].

[4] Consider for example Figure 6 in [200].

7.2. Ineffective design

Many AMTs appear not to have undergone an effective design process. Too often, the cognitive value of graphical models is overlooked in the design of the model.

As in many other visual modelling systems, the design of AMT visual syntax is *"ad hoc"* [132], *"unscientific"*, and based on *"intuition... [and]... personal taste rather than scientific evidence"* [17]. Hogganvik and Stølen adds that the visual syntax is used as *"decoration"* or aesthetic attribute to make the model *"look nicer"* [352]. The effect of this is that the visual modelling system conveys unintended and distorted meanings [17].

7.3. An assumption of cognitive effectiveness

Moody argues that cognitive effectiveness *"is not an intrinsic property"*, it has to be designed into the framework, but too often, the cognitive value of diagrams is assumed without evidence. Bad layout decisions can distort information and give way to unintended judgements on diagram aesthetics [17].

Ineffective design can lead to systems that are cognitively ineffective — resulting in visual syntax that is visually and cognitively unintuitive.

Quite often, these designs have not been tested for their efficacy in enabling observers to understand complex attack sequences. Where they have been tested, this has often been done post-design [320].

There are very few formal methods of evaluating conceptual models and visual syntax [432] and where attempts have been made to evaluate such methods, these have been done in a *"limited fashion"* [433].

Section 5.8 showed that there are limitations in terms of the *diversity of AMT selection, fundamental conceptual differences in AMTs, statistical significance*, and a failure in the effective pedagogic grounding of these studies.

The key contribution of this paper has been a quantitative analysis of the attack graph and attack tree visual syntax proposed by researchers. This analysis has highlighted the range of visual variables used in the published literature. This has been referred to in this paper as *custom and practice* and is summarised in Table 21.

The discussion has also demonstrated four related problems:

- The problem of internal and external semiotic inconsistency.
- That there is wide ranging practice deployed in the construction of attack graphs and attack trees.
- That there are no standard methods for presenting either attack graphs or attack trees.
- The visual syntax of both attack graphs and attack trees is self-selected.

This discussion emphasises the importance of designing an attack graph visual syntax using evidence based design principles. This discussion alongside the theories and methodologies relating to visual syntax design outlined in Section 5 provide the basis for the attack graph design described in [19] and [20].

The work described in Lallie et al. [19] and Lallie et al. [20] is one of the first attempts to develop a standardised attack graph visual syntax.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Table 23
A review of attack tree visual syntax *This table is presented generally in alphabetic order of author surname. However, some entries are swapped to ensure the page space is used efficiently and may not appear in alphabetic order.*

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	Notes
Attack tree	[203]	lr	X	X	X	p	X	X	tx	✓*	*Attack path
Attack tree	[203]	lr	X	X	X	p	X	X	tx	✓*	*Attack path
Attack tree	[394]	bu	X	ft*	r	r	X	X	tx	X	* ft AND/OR and SAND
Dynamic Attack Tree*	[395]	bu	X	ft†	r	r	X	X	tx	X	*Dynamic because it captures probabilities, otherwise the same as the attack tree [394] †ft AND/OR and SAND and SOR
Attack defense tree	[243]	bu	X	arc	c_{rd}	c_{rd}	X	✓	tx	✓*	*dotted dash to highlight mitigations. Mitigations (defenses) are highlighted as green rectangles
Defense tree	[214]	bu	X	arc	X	rr_{gn}	X	✓	tx	✓*	*dotted dash to highlight mitigations. Mitigations (defenses) are highlighted as purple rectangles
Defense tree	[87]	bu	X	arc	X	rr	X	X	tx	✓*	*dotted dash to highlight mitigations. Mitigations (defenses) are highlighted as rectangles
Attack tree	[396]	bu	X	arc*	r	r	X	X	tx	X	* Double line and, absence or, coupled with text
Attack tree	[397]	bu	X	sym	rr	rr	X	X	tx	X	
Attack tree	[92]	bu	X	X	r	r	r*	X	tx	✓*	*thick lined rectangle represents the goal
Fault tree	[419]	bu	X	ftr	X	h*	r†	X	ps	✓†	*double lined rectangle represents goal
Attack tree	[68]	bu	X	X	X	e	X	X	tx	✓*	*used to represent 'implicit' and 'explicit' subgoals
Attack tree	[323]	bu	X	dl*	✓†	fl§	X	X	ps	✓*	*solid edge = and, dotted edge = or †user/computer icon used to represent an achieved postcondition which becomes a precondition to the next stage § flag symbol
Attack tree	[23]	bu	X	ftr	p	p	✓*	X	tx	✓†	*Represented as a 'bomb' icon †The use of dashed and solid lines is not explained in the paper
BDMP tree	[23]	bu	X	ftr	p	p	✓†	✓	tx	✓§	*red dotted arrow used to express SAND †Represented as a 'bomb' icon §The use of dashed and solid lines is not explained in the paper
Protection tree	[387,388]	bu	X	arc	X	X*	X	X	tx	X	*The point of the protection tree is that it highlights protections and not exploits
Attack tree	[90]	bu	X	arc*	r	r	X	X	tx	X	*with a textual label for 'and'
Vulnerability tree	[416]	bu	X	ftr	c|r	X	X	X	tx	X	All the symbols in the vulnerability tree are directly based on the fault tree
Attack tree	[351]	bu	X	c*	X	r	r_{bk}	X	tx	X	*circle with the word 'and'
Defense tree	[417]	bu	X	arc*	r	r	X	X	tx	X	*arc means OR, absence of an arc 'should' represent AND, however this is not clear † Ellipse is used to represent 'security variables', a diamond is used to represent a 'utility node' and a hexagon is used to represent a 'security metric'
Attack tree	[421]	bu	X	$e_{gn}\|e_{bu}$	X	r_{gn}	X	✓*	tx	X	*Green ellipse and blue rectangle used to represent AND/OR respectively
Attack tree	[398]	bu	X	arc*	$e_{gn}\|e_{ye}$	e_{gn} †	c_{bn}	✓§	tx	X	*SAND represented by directed arc, AND/OR not demonstrated †Darker shade of green to the precondition §All ellipses have a red edge
Fault tree	[399]	bu	X	ftr	r	r	X	X	ps	X	
Attack tree	[232]	bu	r_{gy}	X	rr_{gn}	o_{tq}	X	✓	tx	X	
Attack tree	[400]	bu	X	ftr	r	r	r	X	ch	X	
Attack tree	[93]	bu	X	ftr	r_{gy}	r_{gy}	X	✓*	tx	X	
Attack tree	[401]	bu	X	arc*	p	p	X	X	tx	X	*Also utilises a directed arc to represent SAND
Attack tree	[402]	bu	X	arc	r_{gy}	r_{gy}	X	X	tx	X	
Attack tree	[264]	bu*	X	r†	r	r	X	X	ch	X	*Although ef is bu, all the edges point downwards towards the cause of the problem †thick edged rectangle and black rectangle represent AND/OR respectively

(*continued on next page*)

Table 23 (*continued*).

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	Notes
Attack Defense tree	[403]	bu	X	arc	c	c	X	✓*	tx	✓†	*all circles have a red edge. Defense nodes represented as r_{gn}, †dotted edges connect defense nodes to exploits
Fault tree	[256]	bu	X	ftr	c	c	rr*	✓*	ch	X	*all nodes are green
Threat logic trees	[420]	bu	X	arc	X	r	X	X	tx	X	
Threat tree	[103]	bu	X	arc	X	r	X	X	tx	X	
Attack tree	[263]	bu	X	ftr	r*	r	X	X	tx	X	
Attack tree	[130]	bu	X	arc	p	p	X	X	tx	X	
Attack tree	[404]	bu	X	ftr	r	r	X*	✓§	ch	X	*The goal node is supported by a textual label with the words 'goal' †All nodes represented in blue
Vulnerability tree	[405]	bu	X	t*	r†	r†	X	X	tx	X	*Triangle with a plus sign represents AND †double rectangle
Attack tree	[406]	bu	X	sym*	r	r	X	X	ch	X	*∧/∨ symbols in a circle
Penetration attack tree	[393]	lr	X	arc*	X	r	X	X	ch	X	*double arc represents AND
Cyber threat tree	[424]	bu	X	ftr	X	X*	X	X	ch	X	*physical exploit targets — such as 'Hydro plant' are represented, however, the exploit itself is not outlined
Attack tree	[239]	bu*	X	arc†	r§	r§	X	X	tx	✓§	*Although ef is bu, all the edges point downwards towards the cause of the problem †Appended with the word 'AND' §Mitigations highlighted as dashed lines
Threat tree	[407]	bu	X	ftr	c	c	X	X	ch	X	
Attack tree	[408]	bu	X	ftr	r*	r*	X	X	tx	X	*rectangle with a ling 2/3 of the way across. ALSO note, improbable and probable events distinguished by faded ink and normal ink respectively
Vulnerability tree	[233]	bu	X	X	c_{bu}*	r_{bu}	X	✓†	tx	X	*Notable that the circle forces the authors to produce very large circles to get all the text in †All nodes are light blue
Security goal indicator trees	[409]	bu	X	c*	h_{gn}|h_{rd}†	h_{gn}|h_{rd}†	r_{bu}	✓†	tx	X	*Circle with the words 'AND/OR' † h_{rd} is used to represent a 'negative indicator'
Attack tree	[410]	bu	X	arc	e	e	X	✓*	tx	X	*All nodes have a red edge
Attack tree	[411]	bu*	X	sym†	p	p	X	X	ch	X	*Although ef is bu, all the edges point downwards towards the cause of the problem †∧ and ∨ symbols in a circle
Vulnerability tree	[234]	bu	X	ftr	c_{bu}	r_{bu}	X	✓*	tx	X	*All nodes are blue
Incident tree	[69]	bu	X	ft*	r|c	r|c	r_{rd}	✓†	tx	X	*multiple colours depending on attack path †multiple colours
Attack tree	[412]	bu	r	arc*	r	r	X	X	ch	✓†	*Double arc used to represent OR † double edged rectangle
Attack tree	[413]	bu*	X	p	r	r	X	X	tx	X	*Although ef is bu, all the edges point downwards towards the cause of the problem
Attack counter-measure tree*	[389–391]	bu	X	ftr	r	X	X	X	tx	X	*Mitigations given as ellipses. In the 2012 paper, a detection event (rectangle with triangular right edge) is also added
FACT tree	[230]	bu	X	ftr	rr*	rr	r	✓†	tx	✓§	*a 'basic event' is represented as an ellipse. The basic event appears to represent preconditions alongside rounded rectangles †blue trapezoid represents safety countermeasures and a pink pentagon represents security countermeasures § red dotted arrows represent 'triggers'
Attack tree	[422]	bu	X	ft*	✓†	r_{gy}	X	✓*	tx	X	*blue and green or †incorporated within the and/or

(*continued on next page*)

Acknowledgements

The authors would like to thank the following for use of figures in this paper: Bistarelli et al. for permission to reproduce Fig. 5 and Brooke and Paige for permission to reproduce Fig. 9.

Appendix. A review of attack graph and attack tree visual syntax

See Tables 22 and 23.

Table 23 (*continued*).

Type	Citation	ef	ipc	po	pr	ex	go	col	lb	edg	Notes
Attack tree	[180]	bu	X	arc*	r	r	X	X	tx	X	*labelled with the word 'and'
Attack defense tree	[322]	bu	X	arc	c_{gy} |t_{gy}	c_{gy}|t_{gy}	c_{gy}	✓*	ps	✓†	*three shades of grey, darker (circle) to represent goal, lighter (circle and triangle) to represent sub-goal node, lighter (star) to represent a sub-graph †dotted line represents 'non-function dependency'
Attack tree	[235]	bu	X	ftr	c	r	X	✓*	tx	X	*All nodes are presented in grey
Attack tree	[414]	bu	X	r*	rr|r|r_{gn}	rr|r|r_{gn}	X	✓	tx	X	*AND represented as a rounded rectangle, OR represented as a rectangle
Attack tree	[415]	bu	X	X	r	r	X	X	tx	X	
Attack tree	[187]	bu	X	X	X	r_{og}	X	✓	tx	X	
Vulnerability tree	[423]	bu	X	ftr	X	r|e	X	X	tx	X	
Attack defense tree	[99]	bu	X	arc	X	c†	X	X	tx	✓*	*dashed lines used to represent defensive actions †rectangles used to represent defense nodes

Key: **Column descriptors**. *ef*:event flow, *ipc*:initial precondition, *po*:precondition operator, *pr*:precondition, *ex*:exploit, *go*:goal, *col*:colour, *lb*:label type, *edg*:edge type
Colour codes used in accordance with the definitions provided by [418]. *bu*:blue, *bk*:black, *gn*:green, *gy*:grey, *og*:orange, *pk*:pink, *tq*:turquoise, *rd*:red, *vi*:violet, *wh*:white, *ye*:yellow
Shapes. *c*:circle, *r*:rectangle, *p*:plaintext, *e*:ellipse, *d*:diamond, *rr*:rounded rectangle, *h*:hexagon, *o*:octagon, *t*:triangle, *u*:unclear
Precondition operator. *ftr*:fault tree method, *sym*:symbol, *dl*:dotted line, ∧: AND; ∨:OR
Event flow. *td*:top-down, *bu*:bottom-up, *lr*:left–right, *rl*:right–left
Labels. *ps*:pseudonymous, *ch*:character, *tx*:textual, *ie*:in edge (label presented on the edge)

References

[1] IBM Security, Ibm study: hidden costs of data breaches increase expenses for businesses, Technical report, IBM, 2018.
[2] C. Valasek, C. Miller, Remote exploitation of an unaltered passenger vehicle, Report, Ioactive, 2015.
[3] J.M. Ehrenfeld, Wannacry, cybersecurity and health information technology: a time to act, J. Med. Syst. 41 (7) (2017) 104.
[4] N. Falliere, L.O. Murchu, E. Chien, W32. stuxnet dossier, Technical Report, Symantec Corp., Security Response, 2011.
[5] J.W. Coffey, Ameliorating sources of human error in cybersecurity: technological and human-centered approaches, in: 8th International Multi-Conference on Complexity, Informatics and Cybernetics, Pensacola, IMCIC, 2017, pp. 85–88.
[6] N. Baftiu, Cyber security in kosovo, Eur. J. Econ. Law Soc. Sci. 1 (1) (2017).
[7] DCMS, Cyber security breaches survey 2017, Technical Report, Department for Digital, Culture, Media & Sport, UK Government, 2017.
[8] M. Sasse, I. Flechais, Usable security: why do we need it? how do we get it? in: Security and Usability: Designing Secure Systems That People Can Use, O'Reilly, Farnham, UK, 2005.
[9] B.D. Payne, W. Edwards, A brief introduction to usable security, IEEE Internet Comput. 12 (3) (2008).
[10] J.R. Nurse, S. Creese, M. Goldsmith, K. Lamberts, Trustworthy and effective communication of cybersecurity risks: a review, in: 1st Workshop on Socio-technical Aspects in Security and Trust (STAST2011), IEEE, 2011, pp. 60–68.
[11] CSRC, Usability of security, Technical Report, NIST, 2016.
[12] Odgers Berndtson, Cyber security - what boards need to know, 2013, http://www.odgersberndtson.com/media/2253/cyber_security_-_what_boards_need_to_know_01.pdf. (Accessed 20 September 2018).
[13] KPMG, Cyber security: a failure of imagination by ceos, 2015, https://home.kpmg.com/xx/en/home/insights/2015/12/cyber-security-a-failure-of-imagination-by-ceos.html. (Accessed 28 December 2016).
[14] Tanium, The accountability gap, Technical Report, Tanium, 2017.
[15] R.R. Kasemsri, A survey, taxonomy, and analysis of network security visualization techniques (Ph.D. thesis), 2006.
[16] D. Staheli, T. Yu, R.J. Crouser, S. Damodaran, K. Nam, D. O'Gwynn, S. McKenna, L. Harrison, Visualization evaluation for cyber security: trends and future directions, in: Proceedings of the Eleventh Workshop on Visualization for Cyber Security, ACM, 2014, pp. 49–56.
[17] D. Moody, What makes a good diagram? improving the cognitive effectiveness of diagrams in is development, in: Advances in Information Systems Development, Springer, New York, USA, 2007, pp. 481–492.
[18] R. Kang, L. Dabbish, N. Fruchter, S. Kiesler, My data just goes everywhere: user mental models of the internet and implications for privacy and security, in: 2015 Symposium on Usable Privacy and Security, SOUPS2015, 2015.
[19] H.S. Lallie, K. Debattista, J. Bal, An empirical evaluation of the effectiveness of attack graphs and fault trees in cyber-attack perception, IEEE Trans. Inf. Forensics Secur. 13 (5) (2018) 1110–1122.
[20] H.S. Lallie, K. Debattista, J. Bal, Evaluating practitioner cyber-security attack graph configuration preferences, Comput. Secur. 79 (2018) 117–131.
[21] T. Keller, S.-O. Tergan, Visualizing knowledge and information: an introduction, in: Knowledge and Information Visualization, Springer, New York, USA, 2005, pp. 1–23.
[22] J. Homer, A. Varikuti, X. Ou, M.A. McQueen, Improving attack graph visualization through data reduction and attack grouping, in: 5th International Workshop on Visualization for Computer Security, VizSec 2008, Springer, 2008, pp. 68–79.
[23] G. Dondossola, L. Pietre-Cambacedes, J. McDonald, M. Ekstedt, A. Torkilseng, É.d.F. RSE, Modelling of cyber attacks for assessing smart grid security, in: Proceedings Cigré D2 2011 Colloquium, 2011.
[24] S. Roschke, F. Cheng, C. Meinel, A new alert correlation algorithm based on attack graph, in: 4th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2011), Springer, 2011, pp. 58–67.
[25] I. Hogganvik, K. Stølen, Investigating preferences in graphical risk modeling, Technical Report, SINTEF, 2007.
[26] D. Schweitzer, W. Brown, Using visualization to teach security, J. Comput. Sci. Coll. 24 (5) (2009) 143–150.
[27] G.A. Fink, C.L. North, A. Endert, S. Rose, Visualizing cyber security: usable workspaces, in: 6th International Workshop on Visualization for Cyber Security, Vizsec 2009, IEEE, 2009, pp. 45–56.
[28] W.L. Fithen, S.V. Hernan, P.F. O'Rourke, D.A. Shinberg, Formal modeling of vulnerability, Bell Labs Tech. J. 8 (4) (2004) 173–186.
[29] T. Heberlein, M. Bishop, E. Ceesay, M. Danforth, C. Senthilkumar, T. Stallard, A taxonomy for comparing attack-graph approaches, Technical Report, Netsq.com, 2012.
[30] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S.R. Rajagopalan, A. Singhal, Aggregating vulnerability metrics in enterprise networks using attack graphs, J. Comput. Secur. 21 (4) (2013) 561–597.
[31] P. Ning, Y. Cui, D.S. Reeves, Analyzing intensive intrusion alerts via correlation, in: Recent Advances in Intrusion Detection, RAID2002, Springer, 2002, pp. 74–94, 200.
[32] P. Ning, D. Xu, Learning attack strategies from intrusion alerts, in: Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM, 2003, pp. 200–209.
[33] P. Ning, et al., Building attack scenarios through integration of complementary alert correlation method, in: Network and Distributed System Security Symposium (NDSS) Symposium (2004), vol. 4, 2004, pp. 97–111.
[34] K. Ingols, R. Lippmann, K. Piwowarski, Practical attack graph generation for network defense, in: 2015 Symposium on Usable Privacy and Security Conference on Computer Security Applications, ACSAC'06, IEEE, 2006, pp. 121–130.

[35] D. Man, B. Zhang, W. Yang, W. Jin, Y. Yang, A method for global attack graph generation, in: IEEE International Conference on Networking, Sensing and Control, 2008. ICNSC 2008, IEEE, 2008, pp. 236–241.
[36] P. Williams, Executive and board roles in information security, Netw. Secur. 2007 (8) (2007) 11–14.
[37] B. Liu, Sentiment analysis and subjectivity, in: Handbook of Natural Language Processing, vol. 2, 2010, pp. 627–666.
[38] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle, Grids-a graph based intrusion detection system for large networks, in: Proceedings of the 19th National Information Systems Security Conference, vol. 1, Baltimore, 1996, pp. 361–370.
[39] F. Cuppens, Managing alerts in a multi-intrusion detection environment, in: 17th Annual Computer Security Applications Conference, ACSAC2001, vol. 1, IEEE, 2001, p. 22.
[40] P. Ammann, D. Wijesekera, S. Kaushik, Scalable, graph-based network vulnerability analysis, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS'02, ACM, 2002, pp. 217–224.
[41] S. Jha, O. Sheyner, J. Wing, Two formal analyses of attack graphs, in: Proceedings of the 15th IEEE Computer Security Foundations Workshop, CSFW-15, IEEE, 2002, pp. 49–63.
[42] S. Braynov, M. Jadliwala, Representation and analysis of coordinated attacks, in: Proceedings of the 2003 ACM Workshop on Formal Methods in Security Engineering, ACM, 2003, pp. 43–51.
[43] S. Cheung, U. Lindqvist, M.W. Fong, Modeling multistep cyber attacks for scenario recognition, in: Proceedings of the DARPA Information Survivability Conference and Exposition, DISCEX2003, vol. 1, IEEE, 2003, pp. 284–292.
[44] M. Howard, J. Pincus, J.M. Wing, Measuring Relative Attack Surfaces, Springer, New York, USA, 2005.
[45] S. Jajodia, S. Noel, B. O'Berry, Topological analysis of network attack vulnerability, in: Managing Cyber Threats, Springer, New York, USA, 2005, pp. 247–266.
[46] M. Frigault, L. Wang, Measuring network security using bayesian network-based attack graphs, in: 32nd Annual IEEE Conference on International Computer Software and Applications, IEEE, 2008.
[47] N. Idika, B. Bhargava, Extending attack graph-based security metrics and aggregating their application, IEEE Trans. Dependable Secure Comput. 9 (1) (2012) 75–85.
[48] R.W. Ritchey, P. Ammann, Using model checking to analyze network vulnerabilities, in: Proceedings of the IEEE Symposium on Security and Privacy, S&P'2000., IEEE, 2000, pp. 156–165.
[49] G. Sindre, A.L. Opdahl, Templates for misuse case description, in: Proceedings of the 7th International Workshop on Requirements Engineering, Foundation for Software Quality, REFSQ'2001, Citeseer, 2001.
[50] Y. Li, Probabilistic toponym resolution and geographic indexing and querying (Ph.D. thesis), 2007.
[51] L. Wang, C. Yao, A. Singhal, S. Jajodia, Implementing interactive analysis of attack graphs using relational databases, J. Comput. Secur. 16 (4) (2008) 419–437.
[52] S. Jha, O. Sheyner, J.M. Wing, Minimization and reliability analyses of attack graphs, Report, DTIC Document, 2002.
[53] X. Ou, S. Govindavajhala, A.W. Appel, Mulval: a logic-based network security analyzer, in: Usenix Security, 2005.
[54] S.C. Sundaramurthy, L. Zomlot, X. Ou, Practical IDS alert correlation in the face of dynamic threats, in: Proceedings of the International Conference on Security and Management, 2011.
[55] M. Albanese, S. Jajodia, S. Noel, Time-efficient and cost-effective network hardening using attack graphs, in: 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN2012, IEEE, 2012, pp. 1–12.
[56] N. Ghosh, S.K. Ghosh, A planner-based approach to generate and analyze minimal attack graph, Appl. Intell. 36 (2) (2012) 369–390.
[57] P. Ning, Y. Cui, D.S. Reeves, D. Xu, Techniques and tools for analyzing intrusion alerts, ACM Trans. Inf. Syst. Secur. 7 (2) (2004) 274–318.
[58] F. Cuppens, A. Miege, Alert correlation in a cooperative intrusion detection framework, in: Proceedings of the IEEE Symposium on Security and Privacy, 2002, IEEE, 2002, pp. 202–215.
[59] F.M. Alserhani, Knowledge-based model to represent security information and reason about multi-stage attacks, in: Advanced Information Systems Engineering Workshop (CAISE'2015), Springer, 2015, pp. 482–494.
[60] I. Kotenko, M. Stepashkin, Attack graph based evaluation of network security, in: Communications and Multimedia Security, Springer, 2006, pp. 216–227.
[61] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, Validating and restoring defense in depth using attack graphs, in: Military Communications Conference, 2006. MILCOM2006, IEEE, 2006, pp. 1–10.
[62] L. Wang, S. Noel, S. Jajodia, Minimum-cost network hardening using attack graphs, Comput. Commun. 29 (18) (2006) 3812–3824.
[63] S. More, M. Matthews, A. Joshi, T. Finin, A knowledge-based approach to intrusion detection modeling, in: IEEE Symposium on Security and Privacy Workshops (SPW), 2012, IEEE, 2012, pp. 75–81.
[64] M. Albanese, S. Jajodia, A. Pugliese, V. Subrahmanian, Scalable analysis of attack scenarios, in: European Symposium on Research in Computer Security, ESORICS2011, Springer, 2011, pp. 416–433.
[65] M. Alhomidi, M. Reed, Attack graph-based risk assessment and optimisation approach, Int. J. Netw. Secur. Appl. 6 (3) (2014) 31.
[66] F.-X. Aguessy, Évaluation dynamique de risque et calcul de réponses basés sur des modéles d'attaques bayésiens (Ph.D. thesis), 2016.
[67] I.K. Ahmed, A.O. Fapojuwo, Security threat assessment of simultaneous multiple denial-of-service attacks in IEEE 802.22 cognitive radio networks, in: 17th International Symposium on World of Wireless, Mobile and Multimedia Networks, WoWMOM2016, IEEE, 2016, pp. 1–9.
[68] K. Daley, R. Larson, J. Dawkins, A structural framework for modeling multi-stage network attacks, in: Proceedings of the International Conference on Parallel Processing Workshops, ICPP-02, IEEE, 2002, pp. 5–10.
[69] A. Rashid, R. Ramdhany, M. Edwards, S.M. Kibirige, A. Babar, D. Hutchison, Detecting and preventing data exfiltration, Report, University of Lancaster, 2014.
[70] A.E. Tucci, Cyber risks in the marine transportation system, in: Cyber-physical Security, Springer, New York, USA, 2017, pp. 113–131.
[71] G. Kap, D. Ali, Statistical analysis of computer network security, Technical Report, Kth Royal Institute of Technology, Sweden (October 2013), 2013.
[72] J.L. Obes, C. Sarraute, G. Richarte, Attack planning in the real world, Technical Report, Cornell University Library, 2013.
[73] M. Barrere, E.C. Lupu, Naggen: a network attack graph generation tool, in: Proceedings of the IEEE Conference on Communications and Network Security, CNS17, Las Vegas, NV USA.
[74] G. Gonzalez-Granadillo, E. Doynikova, I. Kotenko, J. Garcia-Alfaro, Attack graph-based countermeasure selection using a stateful return on investment metric, in: 10th International Symposium on Foundations and Practice of Security, Springer, 2017, pp. 293–302.
[75] X. Qin, W. Lee, Attack plan recognition and prediction using causal networks, in: 20th Annual Conference on Computer Security Applications, IEEE, 2004, pp. 370–379.
[76] X. Liu, C. Fang, D. Xiao, H. Xu, A goal-oriented approach for modeling and analyzing attack graph, in: International Conference on Information Science and Applications, ICISA2010, IEEE, 2010, pp. 1–8.
[77] E. LeMay, M.D. Ford, K. Keefe, W.H. Sanders, C. Muehrcke, Model-based security metrics using adversary view security evaluation (advise), in: 8th International Conference on Quantitative Evaluation of Systems, QEST2011, IEEE, 2011, pp. 191–200.
[78] V. Gorodetski, I. Kotenko, Attacks against computer network: formal grammar-based framework and simulation tool, in: 12th International Symposium on Recent Advances in Intrusion Detection, RAID2009, Springer, 2002, pp. 219–238, 84.
[79] V. Shandilya, C.B. Simmons, S. Shiva, Use of attack graphs in security systems, J. Comput. Netw. Commun. 2014 (2014).
[80] F. Cuppens, R. Ortalo, LAMBDA: a language to model a database for detection of attacks, in: 3rd International Workshop, RAID 2000, Springer, Toulouse, France, 2000, pp. 197–216.
[81] B. Kordy, L. Pièvre-Cambacédès, P. Schweitzer, DAG-Based attack and defense modeling: don't miss the forest for the attack trees, Comp. Sci. Rev. 13 (2014) 1–38.
[82] G. Sindre, A.L. Opdahl, Eliciting security requirements with misuse cases, Requir. Eng. 10 (1) (2005) 34–44.
[83] C.W. Geib, R.P. Goldman, Plan recognition in intrusion detection systems, in: Proceedings of the DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01, vol. 1, IEEE, 2001, pp. 46–55, 169.
[84] E.J. Byres, M. Franz, D. Miller, The use of attack trees in assessing vulnerabilities in scada systems, in: Proceedings of the International Infrastructure Survivability Workshop, Lisbon, Portugal.
[85] R. Dantu, K. Loper, P. Kolan, Risk management using behavior based attack graphs, in: Proceedings of the International Conference on Information Technology: Coding and Computing, ITCC 2004, Las Vegas, NV, USA.
[86] M. Sudit, A. Stotz, M. Holender, Situational awareness of a coordinated cyber attack, in: Defense and Security, International Society for Optics and Photonics, 2005, pp. 114–129.
[87] S. Bistarelli, M. Dall'Aglio, P. Peretti, Strategic games on defense trees, in: International Workshop on Formal Aspects in Security and Trust, Springer, 2006, pp. 1–15.
[88] W. Li, R.B. Vaughn, Y.S. Dandass, An approach to model network exploitations using exploitation graphs, Simulation 82 (8) (2006) 523–541.
[89] F. den Braber, T. Dimitrakos, B.A. Gran, M.S. Lund, K. Stølen, J.Ø. Aagedal, The coras methodology: model-based risk assessment using UML and UP, in: UML and the Unified Process, 2003, pp. 332–357.

[90] J.H. Espedalen, Attack trees describing security in distributed internet-enabled metrology (Ph.D. thesis), Department of Computer Science and Media Technology, Gjovik University College, 2007.
[91] J. Holsopple, S. Yang, B. Argauer, Virtual terrain: a security-based representation of a computer network, in: Proceedings of Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008, vol. 6973, International Society for Optics and Photonics, 2008.
[92] A. Buoni, M. Fedrizzi, J. Mezei, A delphi-based approach to fraud detection using attack trees and fuzzy numbers, in: Proceeding of the IASK International Conferences, Seville, Spain.
[93] T.R. Ingoldsby, Attack tree-based threat risk analysis, Technical Report, Amenaza Technologies Limited, 2010.
[94] P. Karpati, G. Sindre, A.L. Opdahl, Visualizing cyber attacks with misuse case maps, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2010, pp. 262–275.
[95] A. Matrosov, E. Rodionov, D. Harley, J. Malcho, Stuxnet under the microscope, Technical Report, ESET LLC (September 2010), 2010.
[96] J.P. Landry, J.H. Pardue, T. Johnsten, M. Campbell, P. Patidar, A threat tree for health information security and privacy, in: 17th Americas Conference on Information Systems, AMCIS 2011, 2011.
[97] I. Chokshi, N. Ghosh, S.K. Ghosh, Efficient generation of exploit dependency graph by customized attack modeling technique, in: 18th Annual International Conference on Advanced Computing and Communications, ADCOM2012, IEEE, 2012, pp. 39–45.
[98] B. Han, Q. Wang, F. Yu, X. Zhang, A vulnerability attack graph generation method based on scripts, in: 3rd International Conference on Information Computing and Applications, ICICA2012, 2012, pp. 45–50.
[99] T. Wang, H. Wang, B. Liu, P. Shi, Which strategy is better to restrain C&C activities of unstructured p2p botnets? J. Converg. Inf. Technol. 7 (20) (2012).
[100] L.H. Aslanyan, D. Alipour, M. Heidari, Comparative analysis of attack graphs, Math. Probl. Comput. Sci. 40 (2013) 85–95.
[101] A. Buldas, A. Lenin, New efficient utility upper bounds for the fully adaptive model of attack trees, in: International Conference on Decision and Game Theory for Security, GameSec2013, Springer, 2013, pp. 192–205.
[102] I. Kotenko, A. Chechulin, A cyber attack modeling and impact assessment framework, in: 5th International Conference on Cyber Conflict, CYCON2013, IEEE, 2013, pp. 1–24.
[103] A. Marback, H. Do, K. He, S. Kondamarri, D. Xu, A threat model-based approach to security testing, Softw. - Pract. Exp. 43 (2) (2013) 241–258.
[104] A. Borges, How to perform a heartbleed attack, 2014, https://alexandreborgesbrazil.files.wordpress.com/2014/04/hearbleed_attack_version_a_1.pdf. (Accessed 20 September 2018).
[105] T. Herr, Prep: a framework for malware and cyber weapons, in: 9th International Conference on Cyber Warfare & Security: ICCWS2014, Academic Conferences Limited, 2014, p. 84.
[106] P. Elkind, Inside the hack of the century, Technical Report, Fortune.com, 2015.
[107] K. Kaynar, F. Sivrikaya, Distributed attack graph generation, IEEE Trans. Dependable Secure Comput. 13 (5) (2016) 519–532.
[108] R.M. Lee, M.J. Assante, T. Conway, Analysis of the cyber attack on the ukrainian power grid, 2016, https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf. (Accessed 20 September 2018).
[109] M. Mæhre, Industrial experiences with misuse cases (Ph.D. thesis), Institutt for datateknikk og informasjonsvitenskap, 2005.
[110] P. Baybutt, Cyber security vulnerability analysis: an asset-based approach, Process Saf. Prog. 22 (4) (2003) 220–228.
[111] D. Fisk, Cyber security, building automation, and the intelligent building, Intell. Build. Int. 4 (3) (2012) 169–181.
[112] N.A. Sales, Regulating cyber-security, Northwest. Univ. Law Rev. 107 (2012) 1503.
[113] S.L. Harrington, Cyber security active defense: playing with fire or sound risk management, Richmond J. Law Technol. 20 (4) (2014) 12.
[114] P.J. Brooke, R.F. Paige, Fault trees for security system design and analysis, Comput. Secur. 22 (3) (2003) 256–264.
[115] R. Matulevičius, P. Heymans, Visually effective goal models using kaos, in: International Conference on Conceptual Modeling, Springer, 2007, pp. 265–275.
[116] K.K. Fletcher, X. Liu, Security requirements analysis, specification, prioritization and policy development in cyber-physical systems, in: 5th International Conference on Secure Software Integration & Reliability Improvement Companion, SSIRI-C-2011, pp. 106–113.
[117] P. Karpati, Y. Redda, A.L. Opdahl, G. Sindre, Comparing attack trees and misuse cases in an industrial setting, Inf. Softw. Technol. 56 (3) (2014) 294–308.
[118] M.K. Daly, Advanced persistent threat, Usenix, Nov. 4 (4) (2009) 2013–2016.
[119] O.S. Kerr, Virtual crime, virtual deterrence: a skeptical view of self-help, architecture, and civil liability, J. Law Econ. Policy 1 (2005) 197.
[120] D. Maughan, The need for a national cybersecurity research and development agenda, Commun. ACM 53 (2) (2010) 29–31.
[121] D.S. Wall, Enemies within: redefining the insider threat in organizational security policy, Secur. J. 26 (2) (2013) 107–124.
[122] A.D. Kent, Cyber security data sources for dynamic network research, in: Dynamic Networks and Cyber-Security, World Scientific, 2016, pp. 37–65.
[123] J. Wu, C. Ye, S. Jin, Adversarial organization modeling for network attack/defense, in: Proceedings of the second International Conference on Information Security Practice and Experience, IPSEC2006, Springer, 2006, pp. 90–99.
[124] D.L. Moody, The "physics" of notations: a scientific approach to designing visual notations in software engineering, in: 32nd International Conference on Software Engineering, vol. 2, IEEE, 2010, pp. 485–486.
[125] L.M. Scott, Images in advertising: the need for a theory of visual rhetoric, J. Consum. Res. 21 (2) (1994) 252–273.
[126] G.R. Kress, T. Van Leeuwen, Reading Images: The Grammar of Visual Design, Psychology Press, London, UK, 1996.
[127] IEC, IEC 61025 fault tree analysis, Technical Report, International Electrotechnical Commission, 1990.
[128] J.L. Peterson, Petri nets, ACM Comput. Surv. 9 (3) (1977) 223–252, 1755.
[129] G.C. Dalton, R.F. Mills, J.M. Colombi, R.A. Raines, Analyzing attack trees using generalized stochastic petri nets, in: Information Assurance Workshop, IEEE, 2006, pp. 116–123.
[130] S. Mauw, M. Oostdijk, Foundations of attack trees, in: 8th International Conference on Information Security and Cryptology, ICISC2005, Springer, 2006, pp. 186–198.
[131] C. Alexander, Notes on the Synthesis of Form, vol. 5, Harvard University Press, London, UK, 1964.
[132] D.L. Moody, Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data Knowl. Eng. 55 (3) (2005) 243–276.
[133] E. Byres, A. Ginter, J. Langill, How stuxnet spreads–a study of infection paths in best practice systems, Technical Report, Tofino Security, White Paper, 2011.
[134] B. Chandra, A technical view of the openssl 'heartbleed' vulnerability, Technical report, IBM, 2014.
[135] W. Du, Heartbleed attack lab, Technical Report, CIS, 2016.
[136] Boston University, Sony hack: a nauseating whodunit, Technical Report, Boston University, 2015.
[137] G. Sanchez, Critical controls that sony should have implemented, 2015, https://www.sans.org/reading-room/whitepapers/casestudies/case-study-critical-controls-sony-implemented-36022, (Accessed 14 September 2015).
[138] N. Vlajic, Sony hack:the most puzzling security story of 2014, 2015, http://www.eecs.yorku.ca/course_archive/2014-15/W/3482/SonyHack_Presentation.pdf. (Accessed 20 September 2018).
[139] Radiflow, Ukraine cyber attack analysis, 2016, http://radiflow.com/wp-content/uploads/2015/12/Ukraine_cyber_attack_report.pdf. (Accessed: 20 September 2018).
[140] R. Shirey, Internet security glossary, RFC 2828, RFC Editor.
[141] C. Phillips, L.P. Swiler, A graph-based system for network-vulnerability analysis, in: Proceedings of the 1998 Workshop on New Security Paradigms, ACM, 1998, pp. 71–79.
[142] M. Alhomidi, M.J. Reed, et al., Attack graphs representations, in: 4th Conference on Computer Science and Electronic Engineering, CEEC2012, IEEE, 2012, pp. 83–88.
[143] M. Jun-chun, W. Yong-jun, S. Ji-yin, C. Shan, A minimum cost of network hardening model based on attack graphs, Procedia Eng. 15 (2011) 3227–3233.
[144] M.S. Barik, C. Mazumdar, A graph data model for attack graph generation and analysis, in: Recent Trends in Computer Networks and Distributed Systems Security, Springer, New York, USA, 2014, pp. 239–250.
[145] L.P. Swiler, C. Phillips, D. Ellis, S. Chakerian, Computer-attack graph generation tool, in: DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings, vol. 2, IEEE, 2001, pp. 307–321.
[146] C.R. Taylor, K. Venkatasubramanian, C.A. Shue, Understanding the security of interoperable medical devices using attack graphs, in: Proceedings of the 3rd International Conference on High Confidence Networked Systems, HiCoNS2014, ACM, 2014, pp. 31–40, http://dx.doi.org/10.1145/2566468.2566482.
[147] T. Sommestad, M. Ekstedt, P. Johnson, Combining defense graphs and enterprise architecture models for security analysis, in: 12th International IEEE Conference on Enterprise Distributed Object Computing Conference, 2008, 2008, pp. 349–355.
[148] D. Byers, S. Ardi, N. Shahmehri, C. Duma, Modeling software vulnerabilities with vulnerability cause graphs, in: Proceedings of the International Conference on Software Maintenance, 2006.
[149] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, E. Spafford, ADEPTS: adaptive intrusion response using attack graphs in an e-commerce environment, in: Proceedings of the International Conference on Dependable Systems and Networks, 2005. DSN 2005, IEEE, 2005, pp. 508–517.

[150] M.S. Barik, C. Mazumdar, A novel approach to collaborative security using attack graph, in: 5th International Conference on Internet Multimedia Systems Architecture and Application, IMSAA, 2011, IEEE, 2011, pp. 1–6.
[151] J. Dawkins, J. Hale, A systematic approach to multi-stage network attack analysis, in: Proceedings of the 2nd IEEE International Information Assurance Workshop, 2004, IEEE, 2004, pp. 48–56.
[152] O. Sheyner, J. Wing, Tools for generating and analyzing attack graphs, in: 3rd International symposium for Formal Methods for Components and Objects (FMCO2004), Springer, 2004, pp. 344–371.
[153] J. Lee, H. Lee, H.P. In, Scalable attack graph for risk assessment, in: International Conference on Information Networking, ICOIN 2009, IEEE, 2009, pp. 1–5.
[154] S. Noel, S. Jajodia, Understanding complex network attack graphs through clustered adjacency matrices, in: 21st Annual Computer Security Applications Conference, ACSAC2005, IEEE, 2005, pp. 10–pp.
[155] B. Zhu, A.A. Ghorbani, Alert correlation for extracting attack strategies, Int. J. Netw. Secur. 3 (3) (2006) 244–258.
[156] S. Abraham, S. Nair, A predictive framework for cyber security analytics using attack graphs, Int. J. Comput. Netw. Commun. 7 (1) (2015) 1–17.
[157] M. GhasemiGol, A. Ghaemi-Bafghi, H. Takabi, A comprehensive approach for network attack forecasting, Comput. Secur. 58 (2016) 83–105.
[158] P. Johnson, A. Vernotte, M. Ekstedt, R. Lagerström, Pwnpr3d: an attack-graph-driven probabilistic threat-modeling approach, in: 11th International Conference on Availability, Reliability and Security (ARES), 2016, IEEE, 2016, pp. 278–283.
[159] S. Nanda, N. Deo, A highly scalable model for network attack identification and path prediction, in: Proceedings of the IEEE Southeastcon, 2007, IEEE, 2007, pp. 663–668.
[160] K. Bi, D. Han, J. Wang, K maximum probability attack paths dynamic generation algorithm, Comput. Sci. Inf. Syst. 13 (2) (2016) 677–689.
[161] N. Gao, Y. He, B. Ling, Exploring attack graphs for security risk assessment: a probabilistic approach, Wuhan Univ. J. Nat. Sci. 23 (2) (2018) 171–177.
[162] C.T. Murphy, S.J. Yang, Clustering of multistage cyber attacks using significant services, in: 13th Conference on Information Fusion (FUSION), 2010, IEEE, 2010, pp. 1–7.
[163] T. Long, Attack graph compression (Ph.D. thesis), 2009.
[164] R. Schuppenies, C. Meinel, F. Cheng, Automatic extraction of vulnerability information for attack graphs (Ph.D. thesis), Hasso-plattner-institute for II Systems Engineering, University of Potsdam, 2009.
[165] F. Alserhani, M. Akhlaq, I.U. Awan, A.J. Cullen, P. Mirchandani, MARS: multi-stage attack recognition system, in: 24th International Conference on Advanced Information Networking and Applications, AINA2010, IEEE, 2010, pp. 753–759.
[166] F. Chen, D. Liu, Y. Zhang, J. Su, A scalable approach to analyzing network security using compact attack graph, J. Netw. 5 (5) (2010) 543.
[167] S. Jajodia, S. Noel, Advanced cyber attack modeling analysis and visualization, Technical Report AFRL-RI-RS-TR-2010-078, George Mason University, Fairfax, 2010.
[168] O. Sheyner, J. Haines, S. Jha, R. Lippmann, J.M. Wing, Automated generation and analysis of attack graphs, in: Proceedings of the 2002 IEEE Symposium on Security and Privacy, IEEE, 2002, pp. 273–284.
[169] S. Noel, S. Jajodia, Optimal ids sensor placement and alert prioritization using attack graphs, J. Netw. Syst. Manage. 16 (3) (2008) 259–275, 46.
[170] M. Li, W. Huang, Y. Wang, W. Fan, The optimized attribute attack graph based on apt attack stage model, in: 2nd IEEE International Conference on Computer and Communications (ICCC), 2016, IEEE, 2016, pp. 2781–2785.
[171] T.H. Nguyen, M. Wright, M.P. Wellman, S. Baveja, Multi-stage attack graph security games: heuristic strategies, with empirical game-theoretic analysis, in: Proceedings of the 2017 Workshop on Moving Target Defense, ACM, 2017, pp. 87–97.
[172] S.J. Templeton, K. Levitt, A requires/provides model for computer attacks, in: Proceedings of the 2000 Workshop on New Security Paradigms, ACM, 2001, pp. 31–38.
[173] R.E. Sawilla, X. Ou, Identifying critical attack assets in dependency attack graphs, in: Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security, in: ESORICS '08, Springer-Verlag, Malaga, Spain, 2008, pp. 18–34.
[174] S. Hariri, G. Qu, T. Dharmagadda, M. Ramkishore, C.S. Raghavendra, Impact analysis of faults and attacks in large-scale networks, IEEE Secur. Priv. 99 (5) (2003) 49–54.
[175] N.R. Pokhrel, C.P. Tsokos, Cybersecurity: a stochastic predictive model to determine overall network security risk using markovian process, J. Inf. Secur. 8 (02) (2017) 91–105.
[176] H.M. Almohri, L.T. Watson, D. Yao, X. Ou, Security optimization of dynamic networks with probabilistic graph modeling and linear programming, IEEE Trans. Dependable Secure Comput. 13 (4) (2016) 474–487.
[177] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, J. Wing, Ranking attack graphs, in: Recent Advances in Intrusion Detection, Springer, 2006, pp. 127–144.
[178] M.L. Artz, Netspa: a network security planning architecture (Ph.D. thesis), 2002.
[179] B.B. Madan, K.S. Trivedi, Security modeling and quantification of intrusion tolerant systems using attack-response graph, J. High Speed Netw. 13 (4) (2004) 297–308.
[180] B. Schneier, Attack trees, Dr. Dobb's J. 24 (12) (1999) 21–29, http://dx.doi.org/10.1002/9781119183631.ch21.
[181] S.E. Baker, R. Edwards, How many qualitative interviews is enough, Technical Report, National Centre for Research Methods (NCRM), 2012.
[182] T. Scully, The cyber security threat stops in the boardroom, J. Bus. Contin. Emergency Plan. 7 (2) (2014) 138–148.
[183] I. Alexander, Misuse cases: use cases with hostile intent, IEEE Softw. 20 (1) (2003) 58–66.
[184] R. Matulevicius, N. Mayer, P. Heymans, Alignment of misuse cases with security risk management, in: Third International Conference on Availability, Reliability and Security, 2008. ARES 08, IEEE, 2008, pp. 1397–1404.
[185] J. Whittle, D. Wijesekera, M. Hartong, Executable misuse cases for modeling security concerns, in: 30th International ACM/IEEE Conference on Software Engineering, ICSE'08., IEEE, 2008, pp. 121–130.
[186] A.L. Opdahl, G. Sindre, Experimental comparison of attack trees and misuse cases for security threat identification, Inf. Softw. Technol. 51 (5) (2009) 916–932.
[187] I.A. Tøndel, J. Jensen, L. Røstad, Combining misuse cases with attack trees and security activity models, in: International Conference on Availability, Reliability, and Security, 2010. ARES'10, IEEE, 2010, pp. 438–445.
[188] V. Katta, P. Karpati, A.L. Opdahl, C. Raspotnig, G. Sindre, Comparing two techniques for intrusion visualization, in: IFIP Working Conference on the Practice of Enterprise Modeling, Springer, 2010, pp. 1–15.
[189] D.G. Firesmith, Security use cases, J. Object Technol. 2 (3) (2003).
[190] D. Raptis, T. Dimitrakos, B.A. Gran, K. Stølen, The coras approach for model-based risk management applied to e-commerce domain, in: Advanced Communications and Multimedia Security, Springer, New York, USA, 2002, pp. 169–181.
[191] Y. Stamatiou, E. Skipenes, E. Henriksen, N. Stathiakis, A. Sikianakis, E. Charalambous, N. Antonakis, K. Stølen, F. den Braber, M.S. Lund, et al., The CORAS approach for model-based risk management applied to a telemedicine service, in: Proceedings of Medical Informatics Europe, MIE2003.
[192] H.E. Dahl, I. Hogganvik, K. Stølen, Structured semantics for the coras security risk modelling language, Report STF07 A970, Technical Report, SINTEF Information and Communication Technology, 2007.
[193] K. Beckers, M. Heisel, L. Krautsevich, F. Martinelli, R. Meis, A. Yautsiukhin, Determining the probability of smart grid attacks by combining attack tree and attack graph analysis, in: International Workshop on Smart Grid Security, SmartGridSec2014, Springer, 2014, pp. 30–47.
[194] S. Caltagirone, A. Pendergast, C. Betz, The diamond model of intrusion analysis, Techreport, Cyber Squared Ltd, Southmoor, England, 2013, http://www.activeresponse.org/wp-content/uploads/2013/07/diamond.pdf.
[195] J. Kotheimer, K. O'Meara, D. Shick, Using honeynets and the diamond model for ICS threat analysis, Technical Report, Carnegie Mellon University (Software Engineering Institute), 2016.
[196] E.M. Hutchins, M.J. Cloppert, R.M. Amin, Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains, Leading Issues in Information Warfare & Security Research 1 (1) (2011) 80.
[197] I.-C. Mihai, S. Pruna, I.-D. Barbu, Cyber kill chain analysis, Int. J. Inf. Secur. Cybercrime 3 (2014) 37.
[198] A. Hahn, R.K. Thomas, I. Lozano, A. Cardenas, A multi-layered and kill-chain based security analysis framework for cyber-physical systems, Int. J. Crit. Infrastruct. Prot. 11 (2015) 39–50.
[199] J. Kontio, The riskit method for software risk management, version 1.00, Technical Report, University of Maryland, College Park, MD, USA, 1997.
[200] J. Kontio, V.R. Basili, Riskit: increasing confidence in risk management, Technical Report, np, 1998.
[201] B. Freimut, S. Hartkopf, P. Kaiser, J. Kontio, W. Kobitzsch, An industrial case study of implementing software risk management, in: ACM Sigsoft Software Engineering Notes, vol. 26, ACM, 2001, pp. 277–287.
[202] S. Kaplan, Matrix theory formalism for event tree analysis: application to nuclear-risk analysis, Risk Anal. 2 (1) (1982) 9–18.
[203] C. Alberts, A. Dorofee, OCTAVESM threat profiles, Technical Report, Software Engineering Institute, Carnegie Mellon University, 2001.
[204] K.-K.R. Choo, A cloud security risk-management strategy, IEEE Cloud Comput. 1 (2) (2014) 52–56.
[205] J. Van den Berg, J. van Zoggel, M. Snels, M. van Leeuwen, S. Boeke, L. van de Koppen, J. van der Lubbe, B. van den Berg, T. de Bos, On (the emergence of) cyber security science and its challenges for cyber security education, in: Proceedings of the NATO IST-122 Cyber Security Science and Engineering Symposium, 2014, pp. 13–14.

[206] A. Bialas, Critical infrastructures risk manager–the basic requirements elaboration, in: Theory and Engineering of Complex Systems and Dependability, Springer, New York, USA, 2015, pp. 11–24.
[207] J. Levy, P. Yu, R. Prizzia, Economic disruptions, business continuity planning and disaster forensic analysis: the hawaii business recovery center (hibrc) project, in: Disaster Forensics, Springer, New York, USA, 2016, pp. 315–334.
[208] R.D. Shachter, Evaluating influence diagrams, Oper. Res. 34 (6) (1986) 871–882.
[209] A.M. Agogino, A. Rege, IDES: influence diagram based expert system, Math. Model. 8 (1987) 227–233.
[210] S. Sanner, Relational dynamic influence diagram language (RDDL): language description (Ph.D. thesis), Australian National University, 2010.
[211] M. Ekstedt, T. Sommestad, Enterprise architecture models for cyber security analysis, in: Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES, IEEE, 2009, pp. 1–6.
[212] R. Lagerström, P. Johnson, P. Närman, Extended influence diagram generation, in: Enterprise Interoperability II, Springer, New York, USA, 2007, pp. 599–602.
[213] T. Sommestad, M. Ekstedt, P. Johnson, Cyber security risks assessment with bayesian defense graphs and architectural models, in: 42nd Hawaii International Conference on System Sciences, 2009. HICSS'09, IEEE, 2009, pp. 1–10.
[214] S. Bistarelli, F. Fioravanti, P. Peretti, Defense trees for economic evaluation of security investments, in: First International Conference on Availability, Reliability and Security, ARES'06, IEEE, 2006, pp. 8–pp.
[215] X. Ou, W.F. Boyer, M.A. McQueen, A scalable approach to attack graph generation, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, ACM, 2006, pp. 336–345.
[216] F. Chen, J. Su, Y. Zhang, A scalable approach to full attack graphs generation, in: 7th International Symposium, ESSoS 2015, Springer, 2009, pp. 150–163.
[217] B. Schneier, Secrets and Lies: Digital Security in a Networked World, Wiley, Indianapolis, Indiana, USA, 2000.
[218] R.J. Trudeau, Introduction to Graph Theory, Courier Corporation, Mineola, NY, USA, 2013.
[219] S. Bhattacharya, S. Malhotra, S. Ghsoh, A scalable representation towards attack graph generation, in: 1st International Conference on Information Technology, IT 2008, IEEE, 2008, pp. 1–4.
[220] K. Durkota, V. Lisỳ, B. Bošanskỳ, C. Kiekintveld, Optimal network security hardening using attack graph games, in: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI, 2015, pp. 7–14.
[221] N. Ghosh, S. Ghosh, An intelligent technique for generating minimal attack graph, in: First Workshop on Intelligent Security (Security and Artificial Intelligence), SECART09, Citeseer, 2009.
[222] P. Cheng, L. Wang, T. Long, Compressing attack graphs through reference encoding, in: 10th International Conference on Computer and Information Technology, CIT, 2010, IEEE, 2010, pp. 1026–1031.
[223] M. Dacier, Y. Deswarte, M. Kaâniche, Models and tools for quantitative assessment of operational security, in: Information Systems Security, IBM TJ Watson Research Center, Yorktown Heights, New York, USA, 1996, pp. 177–186.
[224] C. Feng, S. Jin-Shu, A flexible approach to measuring network security using attack graphs, in: International Symposium on Electronic Commerce and Security, ISECS2008, IEEE, 2008, pp. 426–431.
[225] R. Hewett, P. Kijsanayothin, Host-centric model checking for network vulnerability analysis, in: Annual Computer Security Applications Conference, 2008. ACSAC 2008, IEEE, 2008, pp. 225–234.
[226] J. Hong, D.-S. Kim, HARMS: hierarchical attack representation models for network security analysis, in: 10th Australian Information Security Management Conference, SRI Security Research Institute, Edith Cowan University, Perth, Western Australia, 2012.
[227] S. Khaitan, S. Raheja, Finding optimal attack path using attack graphs: a survey, Int. J. Soft Comput. Eng. 1 (3) (2011) 2231–2307.
[228] H.H. Chen, C.-S. Cheng, Fractional factorial designs, in: Design and Analysis of Experiments, Special Designs and Applications, vol. 3, John Wiley & Sons, Hoboken, New Jersey, USA, 2011, p. 299.
[229] K. Ingols, M. Chu, R. Lippmann, S. Webster, S. Boyer, Modeling modern network attacks and countermeasures using attack graphs, in: Annual Computer Security Applications Conference, 2009. ACSAC'09, IEEE, 2009, pp. 117–126.
[230] G. Sabaliauskaite, A.P. Mathur, Aligning cyber-physical system safety and security, in: Proceedings of the 2nd Asia-Pacific Conference on Complex Systems Design & Management (CSD&M Asia 2016), Springer, 2015, pp. 41–53.
[231] S. Noel, S. Jajodia, Managing attack graph complexity through visual hierarchical aggregation, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM, 2004, pp. 109–118.
[232] M. Higuero, J. Unzilla, E. Jacob, P. Saiz, M. Aguado, D. Luengo, Application of 'attack trees' in security analysis of digital contents e-commerce protocols with copyright protection, in: 39th Annual 2005 International Carnahan Conference on Security Technology, 2005. CCST'05, IEEE, 2005, pp. 57–60.
[233] S.C. Patel, J.H. Graham, P.A. Ralston, Quantitatively assessing the vulnerability of critical information systems: a new method for evaluating security enhancements, Int. J. Inf. Manage. 28 (6) (2008) 483–491.
[234] P.A. Ralston, J.H. Graham, J.L. Hieb, Cyber security risk assessment for SCADA and DCS networks, ISA Trans. 46 (4) (2007) 583–594.
[235] E. Tanu, J. Arreymbi, An examination of the security implications of the supervisory control and data acquisition (scada) system in a mobile networked environment: an augmented vulnerability tree approach, in: Proceedings of the 5th Annual Conference on Advances in Computing and Technology, AC&T, 2010, pp. 228–242.
[236] J. Steffan, M. Schumacher, Collaborative attack modeling, in: Proceedings of the 2002 ACM Symposium on Applied Computing, ACM, 2002, pp. 253–259.
[237] A. Xie, Z. Cai, C. Tang, J. Hu, Z. Chen, Evaluating network security with two-layer attack graphs, in: Annual Computer Security Applications Conference (ACSAC'09), IEEE, 2009, pp. 127–136.
[238] J.D. Weiss, A system security engineering process, in: Proceedings of the 14th National Computer Security Conference, vol. 249, 1991, pp. 572–581.
[239] A. Opel, Design and implementation of a support tool for attack trees (Ph.D. thesis), Otto-von-guericke University Magdeburg, 2005.
[240] D.P. Mirembe, M. Muyeba, Threat modeling revisited: improving expressiveness of attack, in: Second UKSIM European Symposium on Computer Modeling and Simulation, EMS'08, IEEE, 2008, pp. 93–98.
[241] S. Bortot, M. Fedrizzi, S. Giove, et al., Modelling fraud detection by attack trees and choquet integral, Technical Report, Department of Computer and Management Sciences, University of Trento, Italy, 2011.
[242] J. Wang, H. Mo, F. Wang, F. Jin, Exploring the network structure and nodal centrality of China's air transport network: a complex network approach, J. Transp. Geogr. 19 (4) (2011) 712–721.
[243] A. Bagnato, B. Kordy, P.H. Meland, P. Schweitzer, Attribute decoration of attack–defense trees, Int. J. Secure Softw. Eng. 3 (2) (2012) 1–35.
[244] R. Vigo, F. Nielson, H.R. Nielson, Automated generation of attack trees, in: 27th IEEE Computer Security Foundations Symposium, CSF, 2014, IEEE, 2014, pp. 337–350.
[245] C. Salter, O.S. Saydjari, B. Schneier, J. Wallner, Toward a secure system engineering methodolgy, in: Proceedings of the 1998 Workshop on New Security Paradigms, ACM, 1998, pp. 2–10.
[246] US Nuclear Safety Commission, Reactor safety study, Technical Report, US Nuclear Safety Commission, 1975.
[247] E.G. Amoroso, Fundamentals of Computer Security Technology, Prentice-Hall, Upper Saddle River, New Jersey, 1994.
[248] N.B. Amor, S. Benferhat, Z. Elouedi, Naive bayes vs decision trees in intrusion detection systems, in: Proceedings of the 19th ACM Symposium on Applied Computing, SAC'04, ACM, 2004, pp. 420–424.
[249] C. Livadas, R. Walsh, D. Lapsley, W.T. Strayer, Using machine learning technliques to identify botnet traffic, in: Proceedings of the 31st IEEE Conference on Local Computer Networks, IEEE, 2006, pp. 967–974.
[250] I. Fette, N. Sadeh, A. Tomasic, Learning to detect phishing emails, in: Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 649–656.
[251] N. Roberts, W. Vesely, D. Haasl, F. Goldberg, Fault tree handbook, nureg-0492, Technical report, US Nuclear Regulatory Commission, 1981.
[252] ECSS, Fault Tree Analysis - Adoption Notice ECSS/IEC 61025, 1997.
[253] BSI, BS EN 61025:2007 - fault tree analysis, 2007.
[254] W. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick III, J. Railsback, Fault tree handbook with aerospace applications version 1.1, Technical Report, NASA Office of Safety and Mission Assurance, NASA HQ, 2002.
[255] C.-Y. Cheng, S.-F. Li, S.-J. Chu, C.-Y. Yeh, R.J. Simmons, Application of fault tree analysis to assess inventory risk: a practical case from aerospace manufacturing, Int. J. Prod. Res. 51 (21) (2013) 6499–6514.
[256] A.J. Kornecki, M. Liu, Fault tree analysis for safety/security verification in aviation software, Electronics 2 (1) (2013) 41–56.
[257] Y.E. Senol, Y.V. Aydogdu, B. Sahin, I. Kilic, Fault tree analysis of chemical cargo contamination by using fuzzy approach, Expert Syst. Appl. 42 (12) (2015) 5232–5244.
[258] H. Lambert, Use of fault tree analysis for automotive reliability and safety analysis, Report UCRL-JC-154905, Lawrence Livermore National Lab., CA (US), 2003.
[259] F. Campean, E. Henshall, A function failure approach to fault tree analysis for automotive systems, Technical Report 0148-7191, SAE Technical Paper, 2008.
[260] J.B. Dugan, S.J. Bavuso, M.A. Boyd, Dynamic fault-tree models for fault-tolerant computer systems, IEEE Trans. Reliab. 41 (3) (1992) 363–377.

[261] R. Manian, J.B. Dugan, D. Coppit, K.J. Sullivan, Combining various solution techniques for dynamic fault tree analysis of computer systems, in: Proceedings of the 3rd IEEE International Conference on High-assurance Systems Engineering Symposium, 1998., IEEE, 1998, pp. 21–28.
[262] R.A. Sahner, K. Trivedi, A. Puliafito, Performance and reliability analysis of computer systems: an example-based approach using the sharpe software package, Springer Science & Business Media, New York, USA, 2012.
[263] M. Masera, I.N. Fovino, A. De Cian, Integrating cyber attacks within fault trees, Reliab. Eng. Syst. Saf. 94 (9) (2009) 1394–1402.
[264] P.A. Khand, System level security modeling using attack trees, in: 2nd International Conference on Computer, Control and Communication, IC42009, IEEE, 2009, pp. 1–6.
[265] H. Wang, J. Hernandez, P. Van Mieghem, Betweenness centrality in a weighted network, Phys. Rev. E 77 (4) (2008) 046105.
[266] M. Dacier, Towards quantitative evaluation of computer security (Ph.D. thesis), 1994.
[267] M. Dacier, Y. Deswarte, Privilege graph: an extension to the typed access matrix model, in: 3rd European Symposium on Research in Computer Security, ESORICS94, Springer, 1994, pp. 319–334.
[268] IBM DeveloperWorks, Ibm spss conjoint 24, Technical report, IBM, 2016.
[269] SANS, Malware FAQ: Sadmind/IIS Worm, Web page, SANS, 2016. http://uk.sans.org/security-resources/malwarefaq/sadmind_iis.php.
[270] O.M. Sheyner, Scenario graphs and attack graphs (Ph.D. thesis), 2004.
[271] R.P. Lippmann, K.W. Ingols, C. Scott, K. Piwowarski, K.J. Kratkiewicz, M. Artz, R.K. Cunningham, Evaluating and strengthening enterprise network security using attack graphs, Technical Report, Michigan Institute of Technology, 2005.
[272] Y. Liu, L. Xiao, X. Liu, L. Ni, X. Zhang, Location awareness in unstructured peer-to-peer systems, IEEE Trans. Parallel Distrib. Syst. (2005) 163–174.
[273] S. Noel, M. Jacobs, P. Kalapa, S. Jajodia, Multiple coordinated views for network attack graphs, in: IEEE Workshop on Visualization for Computer Security, VISZEC 05, IEEE, 2005, pp. 99–106.
[274] T. Zhang, M.-Z. Hu, D. Li, L. Sun, An effective method to generate attack graph, in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, ICMLC 2005, vol. 7, IEEE, 2005, pp. 3926–3931.
[275] L. Wang, A. Liu, S. Jajodia, Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts, Comput. Commun. 29 (15) (2006) 2917–2933.
[276] R. Sawilla, X. Ou, Googling attack graphs, Technical report, Defence R & D Canada, 2007.
[277] L. Wang, A. Singhal, S. Jajodia, Measuring the overall security of network configurations using attack graphs, in: 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Springer, 2007, pp. 98–112.
[278] A. Xie, L. Zhang, J. Hu, Z. Chen, A probability-based approach to attack graphs generation, in: 2nd International Symposium on Electronic Commerce and Security, ISECS'09, vol. 2, IEEE, 2009, pp. 343–347.
[279] S. Zhong, D. Yan, C. Liu, Automatic generation of host-based network attack graph, in: WRI World Congress on Computer Science and Information Engineering, CSIE2009, vol. 1, IEEE, 2009, pp. 93–98.
[280] J. Ma, Y. Wang, J. Sun, X. Hu, A scalable, bidirectional-based search strategy to generate attack graphs, in: IEEE 10th International Conference on Computer and Information Technology (CIT), 2010, IEEE, 2010, pp. 2976–2981.
[281] S. Noel, S. Jajodia, L. Wang, A. Singhal, Measuring security risk of networks using attack graphs, Int. J. Next-generation Comput. 1 (1) (2010) 135–147, 59.
[282] X. Ou, A. Singhal, Attack graph techniques, in: Quantitative Security Risk Assessment of Enterprise Networks, Springer, 2011, pp. 5–8.
[283] M. Keramati, A. Akbari, M. Keramati, CVSS-Based security metrics for quantitative analysis of attack graphs, in: 3th International EConference on Computer and Knowledge Engineering, ICCKE2013, IEEE, 2013, pp. 178–183.
[284] I. Kotenko, A. Chechulin, Fast network attack modeling and security evaluation based on attack graphs, J. Cyber Secur. Mobility 3 (1) (2014) 27–46.
[285] A.K. Nandi, H.R. Medal, S. Vadlamani, Interdicting attack graphs to protect organizations from cyber attacks: a bi-level defender–attacker model, Comput. Oper. Res. 75 (2016) 118–131.
[286] J. Wu, K. Ota, M. Dong, C. Li, A hierarchical security framework for defending against sophisticated attacks on wireless sensor networks in smart cities, IEEE Access 4 (2016) 416–424.
[287] B. Zhang, Q. Li, Y. Zhang, X. Liu, Z. Ni, Generation of cyber-security reinforcement strategies for smart grid based on the attribute-based attack graph, J. Power Technol. 96 (3) (2016) 170.
[288] A. Bates, W.U. Hassan, K. Butler, A. Dobra, B. Reaves, P. Cable, T. Moyer, N. Schear, Transparent web service auditing via network provenance functions, in: Proceedings of the 26th International Conference on World Wide Web, WWW2017, International World Wide Web Conferences Steering Committee, 2017, pp. 887–895.
[289] G.S. Bopche, B.M. Mehtre, Graph similarity metrics for assessing temporal changes in attack surface of dynamic networks, Comput. Secur. 64 (2017) 16–43.
[290] M. Jabbar, G.S. Bopche, B. Deekshatulu, B. Mehtre, Diversity-aware, cost-effective network security hardening using attack graph, in: 5th International Symposium on Security in Computing and Communication SSCC2017, Springer, 2017, pp. 1–15.
[291] P. Luckett, J. McDonald, W.B. Glisson, Attack-graph threat modeling assessment of ambulatory medical devices, in: Hawaii International Conference on System Sciences, HICSS-50, 2017.
[292] A. Sen, S. Madria, Risk assessment in a sensor cloud framework using attack graphs, IEEE Trans. Serv. Comput. 10 (6) (2017) 942–955.
[293] F. Sun, J. Pi, J. Lv, T. Cao, Network security risk assessment system based on attack graph and markov chain, 910 (1) (2017) 012005.
[294] Y. Zheng, K. Lv, C. Hu, A quantitative method for evaluating network security based on attack graph, in: 11th International Conference on Network and System Security, Springer, 2017, pp. 349–358.
[295] H.T. Cheng, H. Shan, W. Zhuang, Infotainment and road safety service support in vehicular networking: from a communication perspective, Mech. Syst. Signal Process. 25 (6) (2011) 2020–2038.
[296] S. Noel, E. Robertson, S. Jajodia, Correlating intrusion events and building attack scenarios through attack graph distances, in: 20th Annual Computer Security Applications Conference, ACSAC2004, IEEE, 2004, pp. 350–359.
[297] M. Urbanska, M. Roberts, I. Ray, A. Howe, Z. Byrne, Accepting the inevitable: factoring the user into home computer security, in: Proceedings of the Third ACM Conference on Data and Application Security and Privacy, CODASPY2013, ACM, 2013, pp. 325–332.
[298] S. Mukherjee, A heuristic-based approach to automatically extract personalized attack graph related concepts from vulnerability descriptions (Ph.D. thesis), 2017.
[299] J. Qian, X.-Y. Li, C. Zhang, L. Chen, T. Jung, J. Han, Social network deanonymization and privacy inference with knowledge graph model, IEEE Trans. Dependable Secure Comput. (2017).
[300] X. Sun, A. Singhal, P. Liu, Towards actionable mission impact assessment in the context of cloud computing, in: IFIP Annual Conference on Data and Applications Security and Privacy XXXI (DBSec2017), Springer, 2017, pp. 259–274.
[301] D. Sgandurra, A. Paudice, E.C. Lupu, et al., Efficient attack graph analysis through approximate inference, ACM Trans. Priv. Secur. 20 (3) (2017) 10, http://dx.doi.org/10.1145/3105760.
[302] R.P. Lippmann, K.W. Ingols, K.J. Piwowarski, Generating a Multiple-prerequisite Attack Graph, US Patent 7,971,252, 2011.
[303] W. Wang, T.E. Daniels, A graph based approach toward network forensics analysis, ACM Trans. Inf. Syst. Secur. 12 (1) (2008) 4.
[304] D. Saha, Extending logical attack graphs for efficient vulnerability analysis, in: Proceedings of the 15th ACM Conference on Computer and Communications Security, ACM, 2008, pp. 63–74.
[305] Z. Zhang, S. Wang, Boosting logical attack graph for efficient security control, in: 7th International Conference on Availability, Reliability and Security, ARES2012, IEEE, 2012, pp. 218–223.
[306] S. Kumar, A. Negi, K. Prasad, A. Mahanti, Evaluation of network risk using attack graph based security metrics, in: IEEE 14th International Conference on Dependable, Autonomic and Secure Computing, 14th Pervasive Intelligence and Computing, 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PICOM/DATACOM/CYBERSCITECH), 2016, IEEE, 2016, pp. 91–93.
[307] K. Prasad, S. Kumar, A. Negi, A. Mahanti, Generation and risk analysis of network attack graph, in: Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015, Springer, 2016, pp. 507–516.
[308] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, et al., Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation, in: DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings, vol. 2, IEEE, 2000, pp. 12–26.
[309] S. Noel, S. Jajodia, Attack graphs for sensor placement, alert prioritization, and attack response, in: Cyberspace Research Workshop, 2007, pp. 1–8.
[310] S. Noel, S. Jajodia, Advanced vulnerability analysis and intrusion detection through predictive attack graphs, Technical Report, Armed Forces Communications and Electronics Association (AFCEA), 2009.
[311] R. Ortalo, Y. Deswarte, M. Kaâniche, Experimenting with quantitative evaluation tools for monitoring operational security, IEEE Trans. Softw. Eng. 25 (5) (1999) 633–650.
[312] Y.M. Liu, I. Traoré, Properties for security measures of software products, Appl. Math. Inf. Sci. J. 1 (2) (2007) 129–156.
[313] S. Noel, S. Jajodia, B. O'Berry, M. Jacobs, Efficient minimum-cost network hardening via exploit dependency graphs, in: Proceedings of the 19th Annual Computer Security Applications Conference, 2003, IEEE, 2003, pp. 86–95.

[314] M.S. Barik, A. Sengupta, C. Mazumdar, Attack graph generation and analysis techniques, Defence Sci. J. 66 (6) (2016) 559.
[315] N. Chaufette, T. Haag, Vulnerability cause graphs: a case of study, Technical report, Linkopings universitet, Sweden, 2007.
[316] J. Bertin, Semiology of Graphics: Diagrams, Networks, Maps, University of Wisconsin press, Wisconsin, USA, 1983.
[317] G.A. Miller, The magic number seven plus or minus two: some limits on our capacity for processing information, Psychol. Rev. 63 (1956) 91–97.
[318] M. Petre, Why looking isn't always seeing: readership skills and graphical programming, Commun. ACM 38 (6) (1995) 33–44.
[319] M. Wertheimer, A brief introduction to gestalt, identifying key theories and principles, Psychol. Forsch. 4 (1923) 301–350.
[320] I. Hogganvik, A graphical approach to security risk analysis (Ph.D. thesis), 2007.
[321] F. den Braber, I. Hogganvik, M. Lund, K. Stølen, F. Vraalsen, Model-based security analysis in seven steps - a guided tour to the coras method, BT Technol. J. 25 (1) (2007) 101–117.
[322] A.S. Sendi, H. Louafi, W. He, M. Cheriet, Dynamic optimal countermeasure selection for intrusion response system, IEEE Trans. Dependable Secure Comput. 15 (2016) 755–770.
[323] R. Dewri, N. Poolsappasit, I. Ray, D. Whitley, Optimal security hardening using multi-objective optimization on attack tree models of networks, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS'07, ACM, 2007, pp. 204–213.
[324] G.A. Miller, The magical number seven, plus or minus two: some limits on our capacity for processing information., Psychol. Rev. 101 (2) (1994) 343.
[325] J. Rumbaugh, I. Jacobson, G. Booch, Unified Modeling Language Reference Manual, The, Pearson Higher Education, Wolverhampton, UK, 2004.
[326] A. McLean, G. Wiggins, Computer programming in the creative arts, in: Computers and Creativity, Springer, New York, USA, 2012, pp. 235–252.
[327] N. Genon, P. Heymans, D. Amyot, Analysing the cognitive effectiveness of the Bpmn 2.0 visual notation, in: 3rd International Conference on Software Language Engineering, SEL2010, Springer, 2010, pp. 377–396.
[328] S. Palmer, I. Rock, Rethinking perceptual organization: the role of uniform connectedness, Psychon. Bull. Rev. 1 (1) (1994) 29–55, http://dx.doi.org/10.3758/bf03200760.
[329] D.A. Wiegmann, D.F. Dansereau, E.C. McCagg, K.L. Rewey, U. Pitre, Effects of knowledge map characteristics on information processing, Contemp. Educ. Psychol. 17 (2) (1992) 136–155.
[330] S.L. Smith, D.W. Thomas, Color versus shape coding in information displays., J. Appl. Psychol. 48 (3) (1964) 137.
[331] W. Winn, An account of how readers search for information in diagrams, Contemp. Educ. Psychol. 18 (2) (1993) 162–185.
[332] H. Störrle, A. Fish, Towards an operationalization of the "physics of notations" for the analysis of visual languages, in: 16th International Conference on Model Driven Engineering Languages and Systems MODELS2013, Springer, 2013, pp. 104–120.
[333] C.P. Gane, T. Sarson, Structured Systems Analysis: Tools and Techniques, Prentice-Hall, Upper Saddle River, New Jersey, 1979.
[334] T. De Marco, Structure analysis and system specification, in: Pioneers and Their Contributions to Software Engineering, Springer, New York, USA, 1979, pp. 255–288.
[335] T. De Marco, Structured analysis and system specification, in: Software Pioneers, Springer, New York, USA, 2002, pp. 529–560.
[336] K. Koffka, Principles of Gestalt Psychology, Routledge, London, UK, 2013.
[337] A. El Kouhen, A. Gherbi, C. Dumoulin, F. Khendek, On the semantic transparency of visual notations: experiments with UML, International SDL Forum, Springer, New York, USA, 2015, pp. 122–137.
[338] P. Karpati, A.L. Opdahl, G. Sindre, Experimental comparison of misuse case maps with misuse cases and system architecture diagrams for eliciting security vulnerabilities and mitigations, in: Sixth International Conference on Availability, Reliability and Security (ARES), IEEE, 2011, pp. 507–514.
[339] P. Caire, N. Genon, P. Heymans, D.L. Moody, Visual notation design 2.0: towards user comprehensible requirements engineering notations, in: 21st IEEE International Conference on Requirements Engineering, RE'13, IEEE, 2013, pp. 115–124.
[340] N. Genon, D. Amyot, P. Heymans, Analysing the cognitive effectiveness of the ucm visual notation, in: 6th International Workshop on System Analysis and Modeling, Springer, 2010, pp. 221–240.
[341] K. Masri, D. Parker, A. Gemino, Using iconic graphics in entity-relationship diagrams: the impact on understanding, J. Database Manag. 19 (3) (2008) 22.
[342] D. Norman, The design of everyday things, Psychol. Everyday Things 20 (1988).
[343] A.v.K. Lemon, O.v.K. Lemon, Constraint matching for diagram design: qualitative visual languages, in: International Conference on Theory and Application of Diagrams, Springer, 2000, pp. 74–88.
[344] P.C.-H. Cheng, R.K. Lowe, M. Scaife, Cognitive science approaches to understanding diagrammatic representations, in: Thinking with Diagrams, Springer, New York, USA, 2001, pp. 79–94.
[345] K.R. Koedinger, J.R. Anderson, Abstract planning and perceptual chunks: elements of expertise in geometry, Cogn. Sci. 14 (4) (1990) 511–550.
[346] T. Halpin, ORM 2 Graphical Notation - Technical Report ORM2–02, Neumont University, 2005.
[347] R. Barker, Case Method: Entity Relationship Modelling, Addison-Wesley Longman Publishing Co., Inc., Boston, Masschusets, USA, 1990.
[348] M.H. Diallo, J. Romero-Mariona, S.E. Sim, T.A. Alspaugh, D.J. Richardson, A comparative evaluation of three approaches to specifying security requirements, in: 12th Working Conference on Requirements Engineering: Foundation for Software Quality, RefsQ'06, 2006.
[349] K. Buyens, B. De Win, W. Joosen, Empirical and statistical analysis of risk analysis-driven techniques for threat management, in: Second International Conference on Availability, Reliability and Security, ARES 2007, IEEE, 2007, pp. 1034–1041.
[350] T. Stålhane, G. Sindre, A comparison of two approaches to safety analysis based on use cases, in: 26th International Conference on Conceptual Modeling, Springer, 2007, pp. 423–437.
[351] O. Flåten, M.S. Lund, How good are attack trees for modelling advanced cyber threats? in: Proceedings of the Norwegian Information Security Conference 2014, 2014.
[352] I. Hogganvik, K. Stølen, On the comprehension of security risk scenarios, in: Proceedings of the 13th International Workshop on Program Comprehension, IWPC 2005, IEEE, 2005, pp. 115–124.
[353] I. Hogganvik, K. Stølen, A graphical approach to risk identification, motivated by empirical investigations, in: International Conference on Model Driven Engineering Languages and Systems, Springer, 2006, pp. 574–588.
[354] F.D. Davis, A technology acceptance model for empirically testing new end-user information systems: theory and results (Ph.D. thesis), 1985.
[355] F. Abed, Cultural influences on visual scanning patterns, J. Cross-Cult. Psychol. 22 (4) (1991) 525–534.
[356] S. Chokron, M. De Agostini, Reading habits influence aesthetic preference, Cogn. Brain Res. 10 (1–2) (2000) 45–49.
[357] Y. Ishii, M. Okubo, M.E. Nicholls, H. Imai, Lateral biases and reading direction: a dissociation between aesthetic preference and line bisection, Brain Cogn. 75 (3) (2011) 242–247.
[358] ISO, Information processing – documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts (ISO 5807:1985), Technical Report, ISO, 1985.
[359] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: an agent-oriented software development methodology, Auton. Agents Multi-Agent Syst. 8 (3) (2004) 203–236.
[360] P. Fonseca Casas, X. Pi Palomés, J. Casanovas Garcia, J. Jové, Definition of virtual reality simulation models using specification and description language diagrams, in: SDL 2013: Model-driven Dependability Engineering, in: Lecture Notes in Computer Science, vol. 7916, Springer, 2013, pp. 258–274.
[361] J. Kontio, V.R. Basili, Risk knowledge capture in the riskit method, in: SEW Proceedings, SEL-96-002, University of Maryland, 1996.
[362] B.M. Michelson, Event-driven architecture overview, 2006.
[363] V. Parondzhanov, Visual syntax of the drakon language, Program. Comput. Softw. 21 (3) (1995).
[364] O. Omojola, Using symbols and shapes for analysis in small focus group research, Qual. Rep. 21 (5) (2016) 832.
[365] G.J. Gorn, A. Chattopadhyay, J. Sengupta, S. Tripathi, Waiting for the web: how screen color affects time perception, J. Mark. Res. 41 (2) (2004) 215–225.
[366] I. Benbasat, A.S. Dexter, P. Todd, An experimental program investigating color-enhanced and graphical information presentation: an integration of the findings, Commun. ACM 29 (11) (1986) 1094–1105.
[367] G. Ghinea, J.P. Thomas, Quality of perception: user quality of service in multimedia presentations, IEEE Trans. Multimed. 7 (4) (2005) 786–789.
[368] J. Xin, K. Cheng, G. Taylor, T. Sato, A. Hansuebsai, Cross-regional comparison of colour emotions part ii: qualitative analysis, Color Res. Appl. 29 (6) (2004) 458–466.
[369] J.H. Xin, K. Cheng, G. Taylor, T. Sato, A. Hansuebsai, Cross-regional comparison of colour emotions part i: quantitative analysis, Color Res. Appl. 29 (6) (2004) 451–457.
[370] D. Kliger, D. Gilad, Red light, green light: color priming in financial decisions, J. Socio-econ. 41 (5) (2012) 738–745.
[371] A.H. Chan, A.W. Ng, Perceptions of implied hazard for visual and auditory alerting signals, Saf. Sci. 47 (3) (2009) 346–352.
[372] M. Zedda, C. Piras, F. Pinna, Road signs: walking among shapes and colors, Int. J. Res. Eng. Technol. 2 (10) (2013) 568–573.
[373] M.A. Rodriguez, What makes a warning label salient? in: Proceedings of the Human Factors Society Annual Meeting, vol. 35, SAGE Publications Sage CA, Los Angeles, CA, 1991, pp. 1029–1033.
[374] Colour Blind Awareness, Colour blindness, 2018, http://www.colourblindawareness.org/. (Accessed 20 September 2018).
[375] C. Ware, Information Visualization: Perception for Design, Elsevier, Amsterdam, The Netherlands, 2012.

[376] S.M. Kosslyn, Understanding charts and graphs, Appl. Cogn. Psychol. 3 (3) (1989) 185–225.
[377] R.E. Mayer, Models for understanding, Review of Educational Research 59 (1) (1989) 43–64.
[378] T.R. Green, Cognitive dimensions of notations, in: Proceedings of the 5th Conference of the British Computer Society, 1989, pp. 443–460.
[379] R. Schuette, T. Rotthowe, The guidelines of modeling–an approach to enhance the quality in information models, in: International Conference on Conceptual Modeling (ER98), Springer, 1998, pp. 240–254.
[380] J. Krogstie, G. Sindre, H. Jørgensen, Process models representing knowledge for action: a revised quality framework, Eur. J. Inf. Syst. 15 (1) (2006) 91–102.
[381] J. Sweller, Cognitive load during problem solving: effects on learning, Cogn. Sci. 12 (2) (1988) 257–285.
[382] J. Sweller, P. Chandler, Why some material is difficult to learn, Cogn. Instr. 12 (3) (1994) 185–233.
[383] D.L. Moody, P. Heymans, R. Matulevičius, Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation, Requir. Eng. 15 (2) (2010) 141–175.
[384] D. Moody, J. van Hillegersberg, Evaluating the visual syntax of uml: an analysis of the cognitive effectiveness of the uml family of diagrams, in: International Conference on Software Language Engineering, Springer, 2008, pp. 16–34.
[385] W. Li, R.B. Vaughn, Cluster security research involving the modeling of network exploitations using exploitation graphs, in: Sixth IEEE International Symposium on Cluster Computing and the Grid, CCGRID06, vol. 2, IEEE, 2006, p. 26.
[386] A. Xie, G. Chen, Y. Wang, Z. Chen, J. Hu, A new method to generate attack graphs, in: 3rd IEEE International Conference on Secure Software Integration and Reliability Improvement, SSIRI 2009, IEEE, 2009, pp. 401–406.
[387] K.S. Edge, G.C. Dalton, R.A. Raines, R.F. Mills, Using attack and protection trees to analyze threats and defenses to homeland security, in: Military Communications Conference, 2006. Milcom 2006, IEEE, 2006, pp. 1–7.
[388] K.S. Edge, A framework for analyzing and mitigating the vulnerabilities of complex systems via attack and protection trees (Ph.D. thesis), Air Force Institute of Technology, Ohio, USA, 2007.
[389] A. Roy, D.S. Kim, K.S. Trivedi, Cyber security analysis using attack countermeasure trees, in: Proceedings of the 6th Annual Workshop on Cyber Security and Information Intelligence Research (STAST2010), in: CSIIRW '10, ACM, 2010, p. 28, http://doi.ACM.org/10.1145/1852666.1852698. (visited 13 April 2017).
[390] A. Roy, D.S. Kim, K.S. Trivedi, Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees, Secur. Commun. Netw. 5 (8) (2012) 929–943.
[391] A. Roy, D.S. Kim, K.S. Trivedi, Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees, in: 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2012, pp. 1–12.
[392] F. Dai, Y. Hu, K. Zheng, B. Wu, Exploring risk flow attack graph for security risk assessment, IET Inf. Secur. 9 (6) (2015) 344–353.
[393] Z. Ning, C. Xin-yuan, Z. Yong-fu, X. Si-yuan, Design and application of penetration attack tree model oriented to attack resistance test, in: International Conference on Computer Science and Software Engineering (CSSE2008), vol. 3, IEEE, 2008, pp. 622–626.
[394] F. Arnold, H. Hermanns, R. Pulungan, M. Stoelinga, Time-dependent analysis of attacks, in: Third International Conference, POST2014, vol. 14, 2014, pp. 285–305.
[395] F. Arnold, D. Guck, R. Kumar, M. Stoelinga, Sequential and parallel attack tree modelling, in: International Conference on Computer Safety, Reliability, and Security, SAFECOMP2015, Springer, 2015, pp. 291–299.
[396] A. Buldas, P. Laud, J. Priisalu, M. Saarepera, J. Willemson, Rational choice of security measures via multi-parameter attack trees, Lecture Notes in Comput. Sci. 4347 (2006) 235–248.
[397] A. Buldas, R. Stepanenko, Upper bounds for adversaries utility in attack trees, in: International Conference on Decision and Game Theory for Security, Springer, 2012, pp. 98–117.
[398] O. Gadyatskaya, R. Jhawar, P. Kordy, K. Lounis, S. Mauw, R. Trujillo-Rasua, Attack trees for practical security assessment: ranking of attack scenarios with adtool 2.0, in: International Conference on Quantitative Evaluation of Systems, QEST2016, Springer, 2016, pp. 159–162.
[399] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, R. Lutz, A software fault tree approach to requirements analysis of an intrusion detection system, Requir. Eng. 7 (4) (2002) 207–220.
[400] J.B. Hong, D.S. Kim, Performance analysis of scalable attack representation models, in: 28th IFIP International Conference on Security and Privacy Protection in Information Processing Systems, SEC2013, Springer, 2013, pp. 330–343, 11.
[401] R. Jhawar, B. Kordy, S. Mauw, S. Radomirović, R. Trujillo-Rasua, Attack trees with sequential conjunction, in: IFIP International Information Security Conference, Springer, 2015, pp. 339–353.
[402] K. Karppinen, Security measurement based on attack trees in a mobile ad hoc network environment, Technical Report, VTT Publications, 2007.
[403] B. Kordy, S. Mauw, M. Melissen, P. Schweitzer, Attack–defense trees and two-player binary zero-sum extensive form games are equivalent, in: International Conference on Decision and Game Theory for Security, Springer, 2010, pp. 245–256.
[404] S. Mishra, K. Kant, R. Yadav, Multi tree view of complex attack–stuxnet, in: Proceedings of the Second International Conference on Advances in Computing and Information Technology, ACITY, Springer, 2012, pp. 171–188.
[405] E. Morakis, S. Vidalis, A. Blyth, Measuring vulnerabilities and their exploitation cycle, Inf. Secur. Tech. Rep. 8 (4) (2003) 45–55.
[406] M. Niitsoo, Optimal adversary behavior for the serial model of financial attack trees., in: 5th International Workshop on Security (IWSEC2010), Springer, 2010, pp. 354–370.
[407] H. Pardue, A. Yasinsac, J. Landry, Towards internet voting security: a threat tree for risk assessment, in: Risks and Security of Internet and Systems (crisis), 2010 Fifth International Conference on, IEEE, 2010, pp. 1–7.
[408] G.-Y. Park, C.K. Lee, J.G. Choi, D.H. Kim, Y.J. Lee, K.-C. Kwon, Cyber security analysis by attack trees for a reactor protection system, in: Proceedings of the Korean Nuclear Society (KNS) Fall Meeting, 2008.
[409] H. Peine, M. Jawurek, S. Mandel, Security goal indicator trees: a model of software features that supports efficient security inspection, in: 11th IEEE High Assurance Systems Engineering Symposium, 2008. Hase 2008, IEEE, 2008, pp. 9–18.
[410] W. Pieters, M. Davarynejad, Calculating adversarial risk from attack trees: control strength and probabilistic attackers, in: 9th International Workshop on Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance, DPM 2014, Springer, 2015, pp. 201–215.
[411] S. Pinchinat, M. Acher, D. Vojtisek, Towards synthesis of attack trees for supporting computer-aided risk analysis, in: International Conference on Software Engineering and Formal Methods, Springer, 2014, pp. 363–375.
[412] I. Ray, N. Poolsapassit, Using attack trees to identify malicious attacks from authorized insiders, in: European Symposium on Research in Computer Security, Springer, 2005, pp. 231–246.
[413] K. Reddy, H.S. Venter, M. Olivier, I. Currie, Towards privacy taxonomy-based attack tree analysis for the protection of consumer information privacy, in: Sixth Annual Conference on Privacy, Security and Trust, 2008. PST'08, IEEE, 2008, pp. 56–64.
[414] C.-W. Ten, C.-C. Liu, M. Govindarasu, Vulnerability assessment of cybersecurity for scada systems using attack trees, in: Power Engineering Society General Meeting, 2007, IEEE, 2007, pp. 1–8.
[415] M. Tentilucci, N. Roberts, S. Kandari, D. Johnson, D. Bogaard, B. Stackpole, G. Markowsky, Crowdsourcing computer security attack trees, in: 10th Annual Symposium on Information Assurance, ASIA'15, 2015, p. 19.
[416] D. Fall, T. Okuda, Y. Kadobayashi, S. Yamaguchi, Towards a vulnerability tree security evaluation of openstack's logical architecture, in: International Conference on Trust and Trustworthy Computing, Springer, 2014, pp. 127–142.
[417] U. Franke, T. Sommestad, M. Ekstedt, P. Johnson, Defense graphs and enterprise architecture for information assurance analysis, Technical Report, Royal Institute of Technology Stockholm (Sweden), 2008.
[418] IEC, Code for designation of colours.
[419] I. Cervesato, C. Meadows, One picture is worth a dozen connectives: a fault-tree representation of npatrl security requirements, IEEE Trans. Dependable Secure Comput. 4 (3) (2007) 216.
[420] A. Marback, H. Do, K. He, S. Kondamarri, D. Xu, Security test generation using threat trees, in: ICSE Workshop on Automation of Software Test (AST'09.), IEEE, 2009, pp. 62–69.
[421] C. Fung, Y.-L. Chen, X. Wang, J. Lee, R. Tarquini, M. Anderson, R. Linger, Survivability analysis of distributed systems using attack tree methodology, in: Military Communications Conference. MILCOM 2005, IEEE, 2005, pp. 583–589.
[422] V. Saini, Q. Duan, V. Paruchuri, Threat modeling using attack trees, J. Comput. Sci. Coll. 23 (4) (2008) 124–131.
[423] S. Vidalis, A. Jones, et al., Using vulnerability trees for decision making in threat assessment, Technical Report, University of Glamorgan, School of Computing, Technical Report. CS-03-2, 2003.
[424] P. Ongsakorn, K. Turney, M. Thornton, S. Nair, S. Szygenda, T. Manikas, Cyber threat trees for large system threat cataloging and analysis, in: 4th Annual IEEE Systems Conference, 2010, IEEE, 2010, pp. 610–615.
[425] L. Wang, A. Singhal, S. Jajodia, Toward measuring network security using attack graphs, in: Proceedings of the 2007 ACM Workshop on Quality of Protection, ACM, 2007, pp. 49–54.
[426] L. Wang, S. Jajodia, in: R. Di Pietro, L.V. Mancini (Eds.), Intrusion Detection Systems, vol. 38, Springer Science & Business Media, New York, USA, 2008.
[427] M. Keramati, A. Akbari, An attack graph based metric for security evaluation of computer networks, in: Telecommunications (ist), 2012 Sixth International Symposium on, IEEE, 2012, pp. 1094–1098.

[428] W. Nichols, P. Hawrylak, J. Hale, M. Papa, Introducing priority into hybrid attack graphs, in: Proceedings of the 12th Annual Conference on Cyber and Information Security Research, ACM, 2017, p. 12.
[429] L. Williams, R. Lippmann, K. Ingols, An interactive attack graph cascade and reachability display, in: Symposium on Visualization for Cyber Security, VIZSEC 2007, Springer, 2008, pp. 221–236.
[430] Y. Zhang, Y. Xiang, L. Wang, Power system reliability assessment incorporating cyber attacks against wind farm energy management systems, IEEE Trans. Smart Grid 8 (5) (2017) 2343–2357.
[431] C. Zhang, S. Wang, D. Zhan, A protocol vulnerability analysis method based on logical attack graph, in: 13th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP2017, Springer, 2017, pp. 309–317.
[432] J.L. Wynekoop, N.L. Russo, Studying system development methodologies: an examination of research methods, Inf. Syst. J. 7 (1) (1997) 47–65.
[433] D.L. Moody, The method evaluation model: a theoretical model for validating information systems design methods, in: Proceedings of the 2013 European Conference on Information Systems, ECIS2013, 2003, p. 79.