

Article

Method for Attack Tree Data Transformation and Import Into IT Risk Analysis Expert Systems

Donatas Vitkus ^{*,†}, Jonathan Salter [†], Nikolaj Goranin [†] and Dainius Čeponis [†]

Department of Information Systems, Vilnius Gediminas Technical University, Saulėtekio al. 11, 10223 Vilnius, Lithuania; jonathan-stein.salter@stud.vgtu.lt (J.S.); nikolaj.goranin@vgtu.lt (N.G.); dainius.ceponis@vgtu.lt (D.Č.)

* Correspondence: d.vitkus@vgtu.lt

† These authors contributed equally to this work.

Received: 28 October 2020; Accepted: 22 November 2020; Published: 26 November 2020

Featured Application: This paper proposes a novel approach of knowledge base formation for expert systems, dedicated to IT security risk analysis, using attack trees as a source of information. Automating the conversion of attack trees to a format that expert systems can use can be applied for minimizing time expenses while creating the knowledge base of an expert system and keeping it up to date, and for further applications as a risk assessment tool by small–medium enterprises.

Abstract: Information technology (IT) security risk analysis preventatively helps organizations in identifying their vulnerable systems or internal controls. Some researchers propose expert systems (ES) as the solution for risk analysis automation since risk analysis by human experts is expensive and timely. By design, ES need a knowledge base, which must be up to date and of high quality. Manual creation of databases is also expensive and cannot ensure stable information renewal. These facts make the knowledge base automation process very important. This paper proposes a novel method of converting attack trees to a format usable by expert systems for utilizing the existing attack tree repositories in facilitating information and IT security risk analysis. The method performs attack tree translation into the Java Expert System Shell (JESS) format, by consistently applying ATTop, a software bridging tool that enables automated analysis of attack trees using a model-driven engineering approach, translating attack trees into the eXtensible Markup Language (XML) format, and using the newly developed ATES (attack trees to expert system) program, performing further XML conversion into JESS compatible format. The detailed method description, along with samples of attack tree conversion and results of conversion experiments on a significant number of attack trees, are presented and discussed. The results demonstrate the high method reliability rate and viability of attack trees as a source for the knowledge bases of expert systems used in the IT security risk analysis process.

Keywords: expert systems; attack trees; risk analysis; information security; JESS; ATTop; Mitre transformation

1. Introduction

It is well known that IT security risk assessment is a vital and sometimes regulatorily mandated process which helps in identifying risks; prioritizing protective measures; and protecting customers, businesses, and private information. Expert systems play a crucial role in taking the knowledge from a security expert, expressed as rules, and allowing it to be shared effortlessly. Many authors [1–3] emphasize that expert systems are adequate for automating risk assessments, thereby minimizing the need for a company to have a security expert. This is of great importance for small–medium sized enterprises that lack human and financial resources. Although the concept of expert systems

is not new and is widely used in many areas [4,5], their application in information security area, including risk analysis [6], is relatively new; i.e., expert systems and methods of their optimization, including methods of automated knowledge base formation, remain a relevant research topic due to a big dynamics and many method application perspectives [7,8].

Sometimes it becomes a problem, since high-quality knowledge is expensive. On the other hand, there exist many sources of systematic information that can be used for knowledge base creation, which can minimize the process expenses at least partially. The idea of automating knowledge base formation is relatively new; still, some research on the area was already done, and currently, proposed methods have demonstrated the possibility and benefits of such an approach. In our previous research, methods for transformation of ontologies into expert system (ES) knowledge base format [9] and integration of data from websites on regional malware distribution [10] for further use in the expert system-based risk analysis were discussed.

Attack trees can assist in the IT risk analysis process by providing a structure to contemplate an attack against a system; i.e., attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks, which can be very useful while evaluating possible threats and their probabilities during the risk assessment. The primary purpose of attack trees is to model security threats, represent attacks against a system, and analyze attack vectors. Some researchers have demonstrated [11–14] that attack trees can be used as a supportive tool in the fields of defense and vulnerability detection. Attack trees are also successfully used in information security risk analysis [15], and design processes of security and defense systems [16] and their analysis [17]. The versatility of attack trees and their wide areas of application have allowed making an assumption that attack trees can be considered as a reliable source of information that can be applied for increasing the efficiency of expert systems. This paper proposes the idea of using attack trees for IT security risk analysis by converting them into ES knowledge base rules.

By developing a process which streamlines the risk assessment process via expert systems, risk analysis can be brought into a more affordable place for small and medium enterprise (SME) category businesses and individuals while simultaneously increasing the accuracy of the risk assessment process [1,18] for everyone, especially SMEs that cannot afford to hire high-quality security experts for risk analysis. Under current methods, a cybersecurity professional, when using an attack tree, is required to painstakingly proceed through several attack trees to perform a risk analysis for a system or to base his decision on a subjective valuation. This is a task that can take a considerable amount of time and effort, as an expert will have to spend his time going through every attack route possible within the attack trees, some of which can have thousands of nodes [19]. Solving the problem of converting attack trees to formats that an expert system can utilize will help reduce the labor costs associated with performing risk assessments. The creation of a knowledge base should provide a higher level of information security, since countless examples of attack trees already exist, with varying levels of detail and complexity, targeted against different information systems and platforms. Utilizing an expert system for this process would also ensure accuracy, reliability, and repeatability for the risk analyses conducted. By converting attack trees to formats usable by expert systems, the benefits of attack trees and expert systems can be combined and multiplied. Moreover, crucially, this method allows the use of existing resources and repositories of attack trees to assist in building a wide knowledge base.

This paper proposes a novel approach of knowledge base formation for expert systems, dedicated to IT security risk analysis, using attack trees as a source of information. The method proposed in this manuscript is novel and exclusive by approach, in that attack trees are being transformed into the expert system knowledge base rules, in contrast to the earlier approaches, where attempts to perform direct use of attack trees' data without transformation were presented. The approach proposed provides a unique possibility to generate an integral knowledge base of an expert system, by integrating different attack trees taken from various sources. Moreover, the generated rules can be imported into the existing knowledge base, where knowledge on other areas of information security

risk is already systematized. While performing transformations, information on attack directions, probabilities, impacts, and possible countermeasures is imported, thereby providing the possibility to use an expert system for decision making and making recommendations on risk management. The research aims to improve the methods for the creation of a knowledge base for the eventual purpose of assisting with conducting risk analyses on an information system or set of systems.

Automating the conversion of attack trees to a format that expert systems can use is relevant and essential because it will allow small companies and individuals to perform risk assessments at a level comparable to an assessment conducted by someone with the resources of large companies [18]. Findings on attack tree data transformation and import into expert risk analysis systems can be applied to significantly reduce the amount of time, effort, and the monetary investment required for the risk analysis of a given company or agency. Additionally, this will allow organizations with limited security financing to prioritize the defense of the most likely places at which an attack could occur.

This article is organized as follows. Section 1, the current section, is an introduction. Section 2 presents related work on attack trees and expert systems. Section 3 presents an approach of automating the conversion of attack trees into a format that can be accepted by an expert system. Section 4 details the experiments, and the weaknesses and strengths of the methodology. Finally, Section 5 concludes the paper.

2. Prior and Related Work

In them classic case, humans—real experts—form expert system knowledge bases. However, it is a slow and expensive method, causing one of the biggest problems in for broader use of expert systems. Nevertheless, expert systems are used nowadays for solving actual problems, e.g., first-line IT service desks, medicine, and law. Several researchers are working on this topic, and several methods of automating the ES knowledge base formation have been proposed.

Currently, three main ways of ES knowledge base formation can be distinguished:

- Manual—the knowledge base is created by human experts in the field;
- Semi-automated—the knowledge base's formation is partly automated;
- Automated—the knowledge base's formation is fully automated, which is the most interesting for our research.

Automated knowledge base formation can be classified into two approaches:

- The first approach is based on using existing knowledge sources directly in their native formats, e.g., ontologies and databases, without additional transformations.
- The second is applied when existing knowledge sources are transformed into ES knowledge base rules.

The first method is cheaper but is very static. Early studies on using existing knowledge sources directly in their native formats considered it to not be a problem until there is a need to change, update, add, or delete information. Using information sources directly in expert systems needs specific ES modification and lacks flexibility. Finally, it is very difficult or sometimes even not possible to merge several sources of information having different formats. The second approach, when existing information sources are transformed into the ES knowledge base rules, is more superior and flexible, and has more areas of application, compared to the first, but requires additional methods of data transformation, additional time, and is sensitive to the correctness of data transformation.

Automated expert system knowledge base formation methods using existing information sources are currently of great interest to researchers. Unfortunately, not too many methods have been proposed for integrating IT security-related data into expert systems. One such method was presented in our earlier research on the use of existing information security ontologies [9]. This method enables the integration of sources that have a deep structure and keeps valuable information on possible recommended controls in the expert systems' knowledge base. In our further research [10] the

applicability of data importation to the knowledge base from the websites was analyzed; specifically, regional malware distribution was imported to prove the influence of geographical threat prevalence on the risk assessment results. It has also ensured regular (in fact daily) updates for the knowledge base. The main limitation of both methods is that they do not give information on cybersecurity attacks, which is crucial while performing IT security risk analysis. Using attack trees as sources for ES, dedicated for IT security risk analysis, knowledge base creation will allow analyzing not only attack vectors, but also the price, likelihood, and defensive measures of such attacks.

Several methods for using attack trees for IT security risk assessments were proposed. One such method is IT security risk assessment for a ship control system based on attack trees [20]. This method uses attack trees directly—not transforming them to ES knowledge base rules—and is dedicated to supervisory control and data acquisition (SCADA) systems. Another example is the method to find the optimal set of countermeasures [21]. This method is a framework relying on attack–defense trees and oriented toward modeling in The Attack-Defense Tree Tool (ADTool) 2.0. The main limitation of such methods is that they use attack trees directly as sources and are not as universal as is needed for expert systems.

An attack tree, first expressed by Bruce Schneier as a “formal, methodical way of describing the security of systems based on varying attacks,” is typically a graphical representation of vulnerabilities within an IT system, although it can be expressed in a text format which is not graphical in nature [22,23]. Whether graphical or text format is represented, there are some similarities between these two representations. Every path through the attack tree represents a unique attack vector through the organization and shows what vulnerabilities an attacker can leverage to obtain access to sensitive material [22]. At every level or leaf of the attack tree, there are different vulnerabilities, some of which must be combined to cause a breach and some which are capable of causing breaches by themselves. These are known respectively as “and” and “or” decompositions, and are represented differently based on whether the graphical, text, or combination format of the attack tree is used [22]. It is up to the creator of the attack tree as to how much information is shown and in what formats the attributes are represented, as there are no formal standards in place [23–26]. Despite the lack of formal standards, there are some recommendations for attack tree generation, such as creating the trees, measuring the probabilities of each leaf, removing or marking improbable attack paths, generating countermeasures, and applying the most appropriate countermeasures to leaves [19,27].

An example of the attack tree in a graphical format is shown in Figure 1.

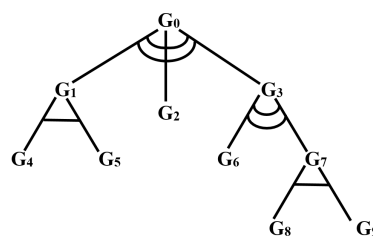


Figure 1. Graphical format attack tree [22].

Each independent level is called a “leaf” node, and vulnerabilities are moved through during an attack from the bottom to the top, which is called the “root” node [23]. For this attack tree, straight horizontal lines represent “and” decompositions, and curved horizontal lines represent “or” decompositions.

There are several programs which can be used to aid with the graphical representation portion of the attack tree creation process. However, there is still a large amount of input required on the part of an expert person with manual attack tree generation [17,28]. Many of these graphing tools also include analysis functions to both ensure that data are entered in an accepted format and to analyze the paths from leaf nodes to the root node.

An example of the text format for an attack tree is shown in Figure 2, which shows an attack tree for intellectual property theft against the fictional ACME, Inc.

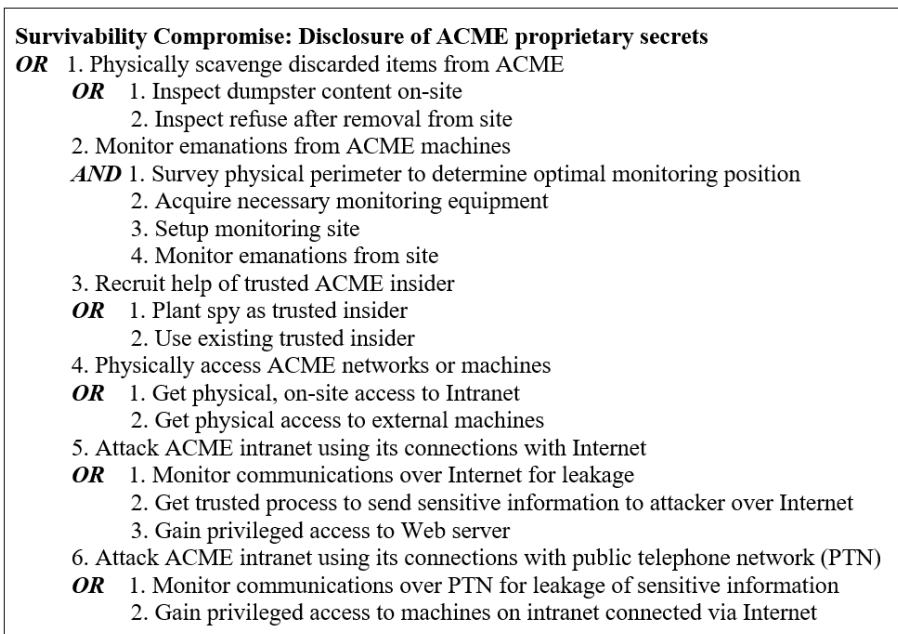


Figure 2. Text format attack tree [22].

Just as with the graphical format, the tasks at each level are shown as “and” or “or” to show whether all steps or only one must be conducted at that level to progress with the attack. At the top of the text format shown in Figure 2, it can be noted that all six nodes are “or” nodes to the root and only node two has “and” nodes for its children. Text formats are often used when dealing with more complex attack trees, as they are easier to read than a complex and branching graphical format with dozens of paths [23].

The easiest way of working with attack trees for a human expert is the graphical format. However, for integration into information systems, text format is preferred. For solving this problem, researchers [29–31] have proposed methods for transforming attack trees from graphical format to text and vice versa. This has increased the usability of attack trees. Nowadays, attack trees are widely used in the fields of defense, risks analysis, threat modeling against information, electronic systems, computer control, and physical systems. Based on the analysis of attack trees’ data, practitioners can define actions to reduce or annihilate risks. A significant barrier in integrating attack trees into the knowledge base of ES is that attack trees can become largely complex and thus hard to specify [32].

3. Attack Trees in XML

The eXtensible Markup Language, or XML, is a language created by the World Wide Web Consortium (W3C) to store and transport data in a format which is readable to both computers and humans (W3Schools, n.d.). It is designed to be extremely flexible and allows creators to define their naming conventions within the XML structure. For example, the attack tree in Figure 3 is an example tree from the ADTree attack tree graphing tool named “RFID Communication Block” that was exported to XML by the tool.

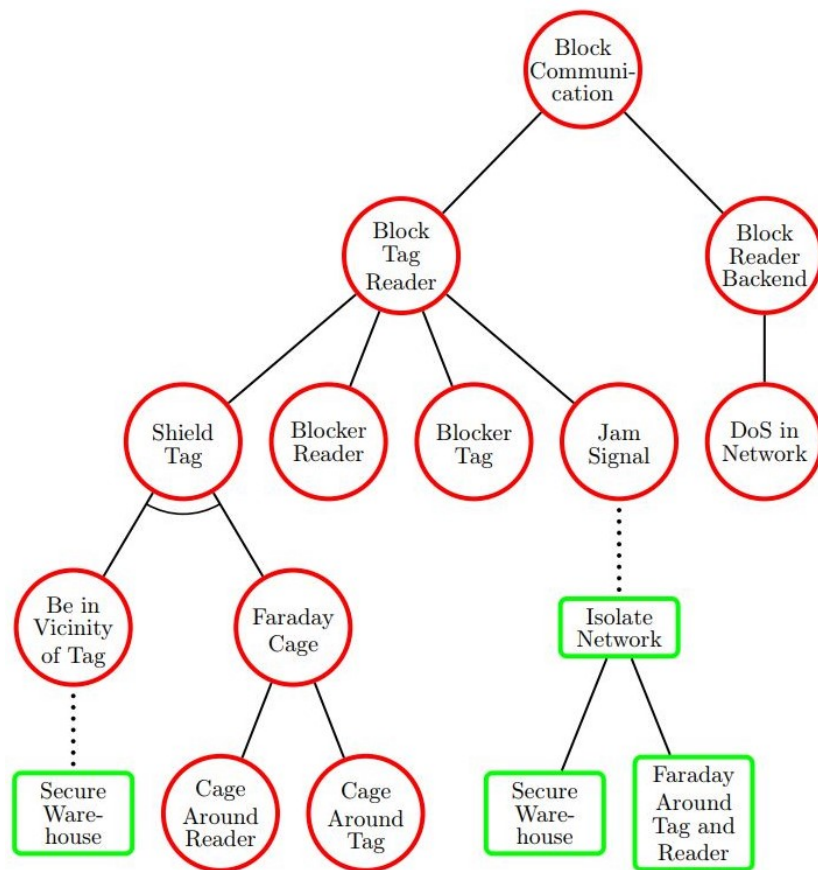


Figure 3. RFID Communication Block attack tree [33].

The result of this export is presented in Figure 4.

```

1  <?xml version='1.0'?>
2  <adtrees>
3  |   <node refinement="disjunctive">
4  |   |   <label>Block Communication</label>
5  |   |   <node refinement="disjunctive">
6  |   |   |   <label>Block Tag Reader</label>
7  |   |   |   <node refinement="conjunctive">
8  |   |   |   |   <label>Shield Tag</label>
9  |   |   |   |   <node refinement="disjunctive">
10 |   |   |   |   |   <label>Be in Vicinity of Tag</label>
11 |   |   |   |   |   <node refinement="disjunctive" switchRole="yes">
12 |   |   |   |   |   |   <label>Secure Warehouse</label>
13 |   |   |   |   |   </node>
14 |   |   |   |   </node>
15 |   |   |   |   <node refinement="disjunctive">
16 |   |   |   |   |   <label>Faraday Cage</label>
17 |   |   |   |   |   <node refinement="disjunctive">
18 |   |   |   |   |   |   <label>Cage Around Reader</label>
19 |   |   |   |   |   </node>
20 |   |   |   |   |   <node refinement="disjunctive">
21 |   |   |   |   |   |   <label>Cage Around Tag</label>
22 |   |   |   |   |   </node>
23 |   |   |   |   </node>
24 |   |   |   </node>
  
```

Figure 4. RFID Communication Block attack tree in XML [33].

This variability in XML formatting raises problems within the attack tree conversion process, as each tool may define attributes using different names. For example, on line 11 of Figure 4 text, the term “switchRole” indicates that the node “Secure Warehouse” is a countermeasure node rather than a vulnerability node. In the universal metamodel for attack trees (UATS) format of this same file,

“switchRole” is only called “role,” but this means any program that wants to extract and work with this XML data must be programmed to find all names for these attributes.

Some tools can be used to aid with the graphical representation portion of the attack tree creation process, although there is still a large amount of input required on the part of an expert person with manual attack tree generation [17,28]. While some researchers have tried to overcome this via crowdsourcing, others have attempted to utilize AI to generate the attack trees while an expert person reviews the automated creation results [34–36]. One common task which requires the use of an expert person is verification that the vulnerabilities expressed by an attack tree correlate with the system the attack tree is designed for [34]. Research has been performed with the goal of reducing the expert-required workload, including attempts at utilizing vulnerability databases such as the National Vulnerability Database (NVD) and Mitre Organization’s Common Attack Pattern Enumeration and Classification (CAPEC) [35]. While this research has shown considerable promise, there remains a need for experts at multiple steps in the attack tree generation, verification, and review stages. By inputting these attack trees into an expert system which has been configured to present an expert opinion, the need for an expert person could be further minimized.

4. Method

The proposed method is a process which utilizes two software bridging programs together to implement the automation of converting information from attack trees into an expert system of knowledge based rules. This method is implemented via a Python-based program named ATES, which will take output in UATS format from the ATTop program mentioned above and will create ES knowledge base rules to the output files. The output of ATES is either one or two files, depending upon the user selection, which are readable by the JESS language. Version one outputs a singular file containing the structure for the JESS expert system and the attack tree data, while version two outputs two files, one containing the structure for the JESS expert system and the other containing the attack tree data. The general proposed method view is presented in Figure 5.

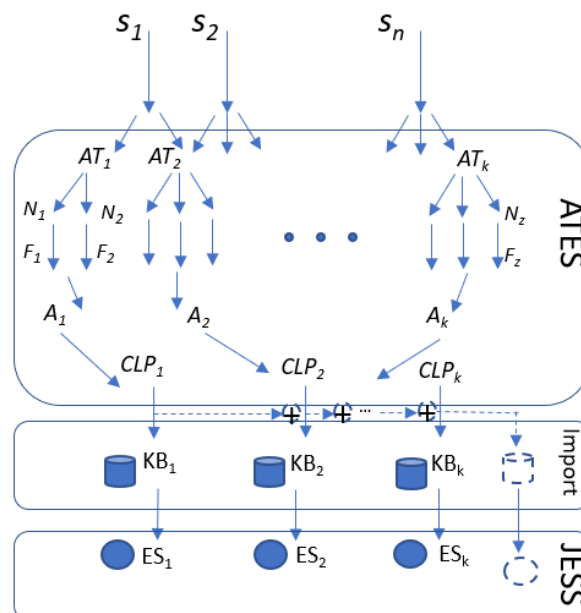


Figure 5. The method proposed.

As was said earlier, the proposed method is based on data transformation from attack trees ($AT_1 \dots AT_k$) collected from different sources ($S_1 \dots S_n$) into ES knowledge base facts. Nodes and leaves ($N_1 \dots N_z$) of the attack trees are transformed into facts ($F_1 \dots F_z$), while the root of the attack tree is used to generate the attack description ($A_1 \dots A_k$). Every attack tree transformed is saved in the form

of C Language Integrated Production System (CLIPS) rules in a newly generated file ($CLP_1 .. CLP_k$). Every generated CLP file is formed as a knowledge base of an expert system ($KB_1 .. KB_k$) that can be later imported into the unique expert system ($ES_1 .. ES_k$) or can be merged with an existing knowledge base (dotted notation on Figure 5).

The method proposed does not depend on the source of attack trees, and it does not limit the amount of attack trees; i.e., S —the source of all attack trees—can be formed of several different sources.

$$S = \{S_1, S_2, \dots, S_n\}$$

Each source can have several attack trees (AT).

$$S_1 = \{AT_1, AT_2, \dots, AT_l\},$$

$$S_2 = \{AT_{l+1}, AT_{l+2}, \dots, AT_j\},$$

...

$$S_n = \{AT_{y+1}, AT_{y+2}, \dots, AT_k\}.$$

Due to that attack trees from different sources can be joined, thereby forming the set of attack trees AT .

$$AT = \sum_{i=1}^n S_i$$

$$AT = \{AT_1, AT_2, \dots, AT_k\}.$$

Each attack tree can be of a different depth and structure, with a different number of leaves (marked as N). Still, every attack tree has a root leaf, called the root node (marked as R).

$$AT_1 = \{R_1, N_1, N_2, \dots, N_o\},$$

$$AT_2 = \{R_2, N_{o+1}, N_{o+2}, \dots, N_p\},$$

...

$$AT_k = \{R_k, N_{x+1}, N_{p+2}, \dots, N_z\}.$$

By joining leaves and root nodes of all attack trees, we will get the set of attack tree elements NR .

$$NR = \sum_{i=0}^k AT_i$$

$$NR = \{R_1, R_2, \dots, R_k, N_1, N_2, \dots, N_z\}.$$

Transformation: Every element in set AT is transformed into elements in set A :

$$AT\{AT_1, AT_2, \dots, AT_k\} \Rightarrow A\{A_1, A_2, \dots, A_k\};$$

i.e., the attack tree is transformed into separate attacks in the knowledge base. Since AT set is formed of several elements, this means that every element should be transformed separately. Set of attack tree elements NR is transformed into a set of knowledge base facts and aims FG :

$$NR\{R_1, R_2, \dots, R_k, N_1, N_2, \dots, N_z\} \Rightarrow FG\{G_1, G_2, \dots, G_k, F_1, F_2, \dots, F_z\}.$$

In other words, every attack set A is formed of the attack aim G and facts F :

$$A_1 = \{G_1, F_1, F_2, \dots, F_o\},$$

$$A_2 = \{G_2, F_{0+1}, F_{0+2}, \dots, F_p\},$$

...

$$A_k = \{G_k, F_{x+1}, F_{p+2}, \dots, F_z\}.$$

After the transformation is performed, it is necessary to prepare data in a format suitable for the expert system. We are using C Language Integrated Production System (CLIPS) rules. As for that, the CLP file is formed, it is composed of three main elements:

- Tree template (*TT*)—information about the converted attack tree (*AT* name, and root node (attack goal) name);
- Node template (*NN*)—a predefined structure to work with facts generated from attack trees;
- Facts about the attack, generated in the transformation process.

That is, the *CLP* is composed of the following elements:

$$CLP_1 = \{TT, NT, A_1\} = \{TT, NT, G_1, F_1, F_2, \dots, F_0\},$$

$$CLP_2 = \{TT, NT, A_2\} = \{TT, NT, G_2, F_{0+1}, F_{0+2}, \dots, F_p\},$$

...

$$CLP_k = \{TT, NT, A_k\} = \{TT, NT, G_k, F_{x+1}, F_{x+2}, \dots, F_z\}.$$

Figure 6 depicts an activity diagram of the attack tree data proceeding by the methodology from creation as an attack tree to the final transformation to the JESS expert system knowledge base rules.

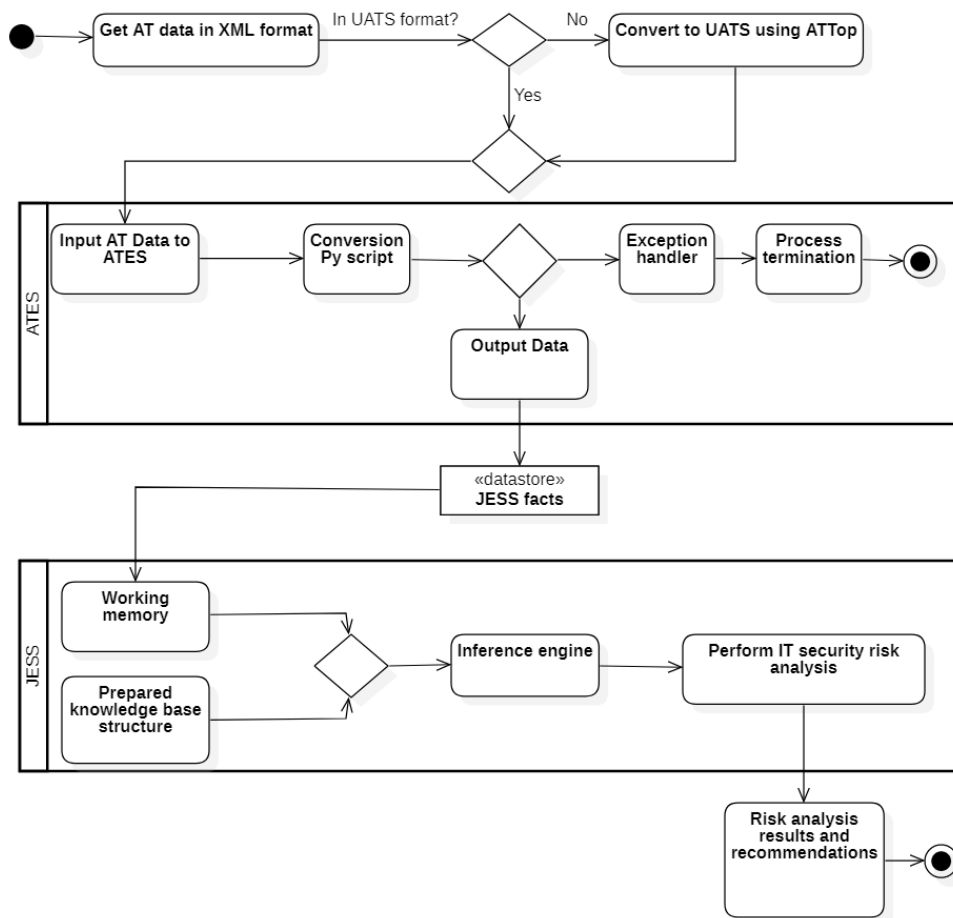


Figure 6. Proposed method activity diagram.

The main steps presented in Figure 6 are described below.

- **Get AT data in XML format.** The initial transformation of AT into XML format is performed due to its versatility.
- **In UATS format.** XML UATS format enables the use of a larger variety of AT sources, so our tool is made to work with XML UATS from the beginning. Examples of AT in XML UATS and graphical formats are given in Figure 7.

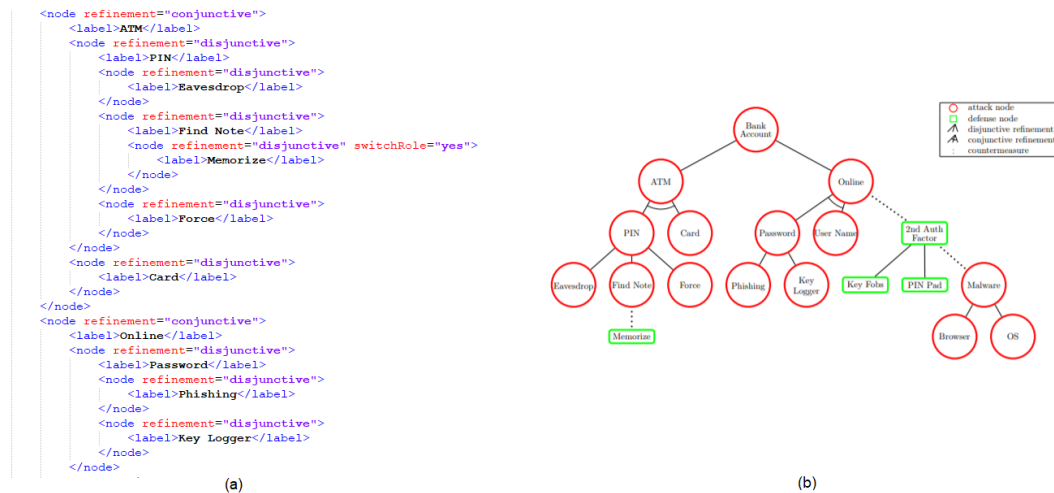


Figure 7. Examples of attack trees (a) in XML UATS format and (b) in graphical format.

- **Convert to UATS using ATTop.** The attack tree can be directly exported to XML UATS or converted from various formats. The tool we propose using in this step is ATTop [37].
- **Input AT Data to ATES.** ATES is our Python-based software for the method’s practical implementation that performs conversion of UATS into the JESS format.
- **Conversion Py script.** Conversion engine which converts XML UATS to CLP rules and DAT facts by applying defined patterns. Examples of resulting ES knowledge base facts are presented in Figure 8.

```

(MAIN::tree (tname "Bank Account UATSv2") (rootnode "Bank Account"))
(MAIN::node (id 0) (name "Bank Account") (parent nil) (connector OR) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children 1 2))
(MAIN::node (id 1) (name "ATM") (parent 0) (connector AND) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 3) (countermeasure FALSE) (children 3 4))
(MAIN::node (id 2) (name "PIN") (parent 1) (connector OR) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children 3 4 5))
(MAIN::node (id 3) (name "Eavesdrop") (parent 2) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children nil))
(MAIN::node (id 4) (name "Find Note") (parent 2) (connector OR) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children 5))
(MAIN::node (id 5) (name "Memorize") (parent 4) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure TRUE) (children nil))
(MAIN::node (id 6) (name "Card") (parent 1) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children nil))
(MAIN::node (id 7) (name "Online") (parent 0) (connector AND) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children 8 12 13))
(MAIN::node (id 8) (name "Password") (parent 7) (connector OR) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children 10 11))
(MAIN::node (id 9) (name "Phishing") (parent 8) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children nil))
(MAIN::node (id 10) (name "Key Logger") (parent 8) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children nil))
(MAIN::node (id 11) (name "User Name") (parent 8) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure TRUE) (children nil))
(MAIN::node (id 12) (name "2nd Auth Factor") (parent 7) (connector OR) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure TRUE) (children 14 15 16))
(MAIN::node (id 13) (name "Key Fobs") (parent 12) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children nil))
(MAIN::node (id 14) (name "PIN Pad") (parent 12) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure TRUE) (children nil))
(MAIN::node (id 15) (name "Malware") (parent 12) (connector OR) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children 17 18))
(MAIN::node (id 16) (name "Browser") (parent 15) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children nil))
(MAIN::node (id 17) (name "OS") (parent 15) (connector nil) (cost 0) (time 0) (exploited FALSE) (probability 0.0) (difficulty 0) (countermeasure FALSE) (children nil))
  
```

Figure 8. Example of an attack tree converted to ES facts.

- **Exception handler.** The exception handler, a “try–except” code block that will terminate the processes and return an error message to the user in case any errors occur.
- **Process termination.** Terminates the process if any error occurs.
- **Output data.** Converted rules are exported to the CLP and DAT files (or one combined file if the user selects this option), supported by the JESS ES platform.
- **JESS facts.** The result of the transformation process is a JESS-based expert system knowledge base of rules and facts.
- **Working memory.** Automatically generated ES knowledge base rules and facts in CLP format are imported into ES working memory.
- **Prepared knowledge base structure.** ES should be able to work with the given (imported) facts. Our method utilizes the prepared knowledge base structure (Figure 9), which accepts facts generated by ATES.

```
(deftemplate node "Data about nodes"
  (slot id
    (type INTEGER))
  (slot name
    (type STRING)
    (default ""))
  (slot parent
    (type STRING)
    (default ""))
  (slot connector
    (type STRING)
    (default nil)
    (allowed-values nil AND OR XOR SAND))
  (slot cost
    (type INTEGER)
    (default 0))
  (slot time
    (type INTEGER)
    (default 0))
  (slot difficulty
    (type INTEGER)
    (default 0))
  (slot probability
    (type FLOAT)
    (default 0.0))
  (slot exploited
    (type SYMBOL)
    (default FALSE)
    (allowed-values TRUE FALSE))
  (slot countermeasure
    (type SYMBOL)
    (default FALSE)
    (allowed-values TRUE FALSE))
  (multislot children
    (type INTEGER)
    (default nil)))
```

Figure 9. Example of the prepared knowledge base structure in JESS.

- **Inference engine.** We use the JESS inference engine, which utilizes the RETE algorithm.
- **Perform IT security risk analysis.** At this moment, our ES is prepared for IT security risk analysis, utilizing automatically generated knowledge base rules from the attack trees.
- **Risk analysis results and recommendations.** The expert system, interacting with the user, performs risk analysis, and gives the results.

While performing the conversion of UATS into the ES knowledge base facts, each attack tree node is transformed into one fact, as presented in Figure 10.

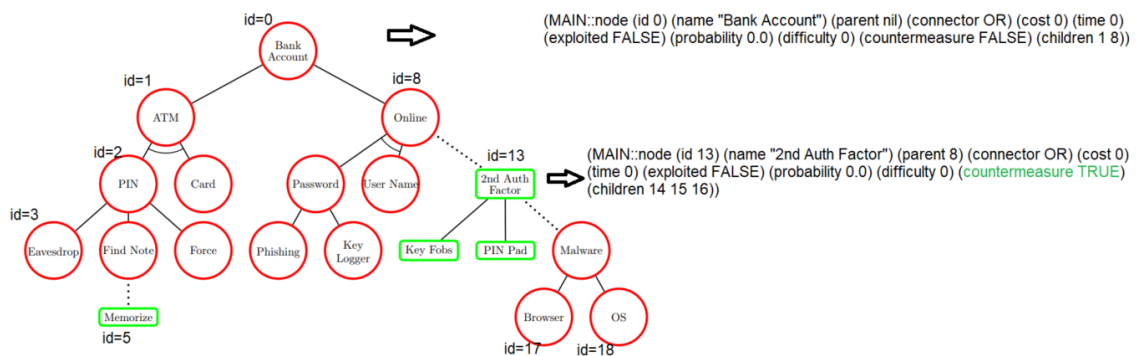


Figure 10. Example of the attack tree transformation to ES knowledge base facts.

On the left side of Figure 10 is an attack tree, and on the right side—two samples of generated ES facts. Every generated fact has the following structure:

- **ID.** It is the ID of the node. Count from top to bottom and from left to right.
- **Name.** The name of the node is read from the attack tree.
- **Parent.** Depicts which node is the parent of the current node. The value is the node ID number.

- **Connector.** It defines the node type. Possible values are OR, AND.
- **Cost.** It is the cost of the attack if this value is specified in the attack tree.
- **Time.** It is the time needed to perform the attack if this value is given in the attack tree.
- **Exploited.** It is the state of the node. By default, all values have FALSE values, and they can be changed during the risk analysis process in the ES. A value can become TRUE based on the user’s answers.
- **Probability.** It is the probability of the attack if this value is given in the attack tree.
- **Difficulty.** It is the difficulty of the attack if this value is given in the attack tree.
- **Countermeasure.** It identifies whether the node already has countermeasures applied or not. Possible values are TRUE, FALSE.
- **Children.** It shows what children this current node has. The value is node ID numbers.

5. Results and Discussion

The method proposed was practically implemented, and several experiments were performed to demonstrate method reliability and applicability. For method implementation, the ATES tool, which converts AT to ES knowledge base rules, was developed. Later, automatically generated ES knowledge base rules and facts were imported into the created ES (based on JESS), which allows us to perform risk analysis using the generated knowledge base. Figure 11 provides details on the path that the attack tree passes through in the ATES program.

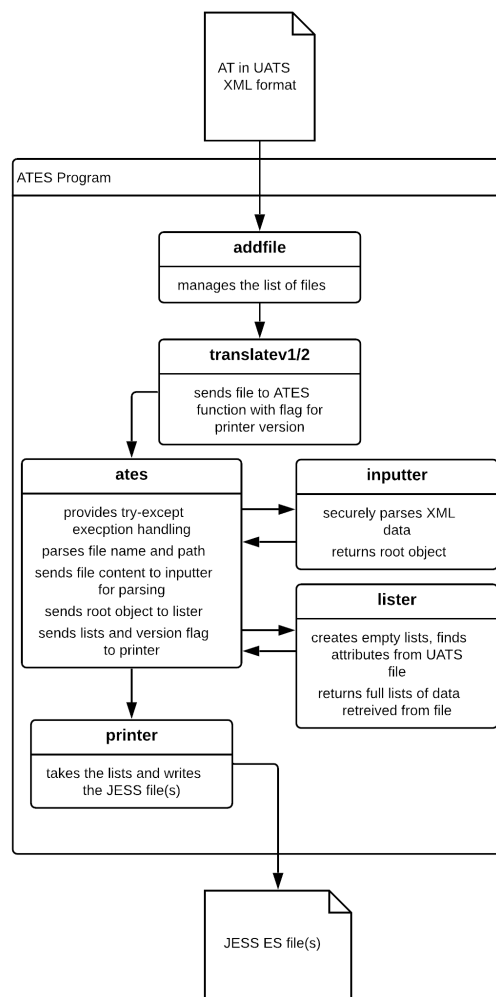
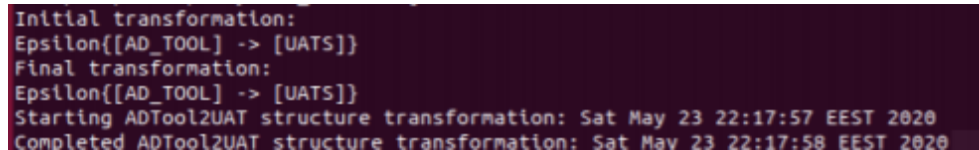


Figure 11. Conceptual model of ATES.

The ATES was written in Python version 3.8 and should be compatible with any Python version greater than 3. Below the short description of functions implemented in ATES is provided:

- **Preparing attack tree data (data pre-processing).** The creation of an attack tree in one of the various formats accepted by ATTop or direct creation of a UATS XML file must be done by the user. Once the UATS file is returned from ATTop, the ATES program is ready to accept the file for translation. Successful transformation to UATS via ATTop is demonstrated in Figure 12.



```
Initial transformation:
Epsilon([AD_TOOL] -> [UATS])
Final transformation:
Epsilon([AD_TOOL] -> [UATS])
Starting ADTool2UAT structure transformation: Sat May 23 22:17:57 EEST 2020
Completed ADTool2UAT structure transformation: Sat May 23 22:17:58 EEST 2020
```

Figure 12. Successful transformation via ATTop.

- **Add file.** Once the UATS file is returned from ATTop, the ATES program is ready to accept the file for translation. ATES has a simple GUI created using the Tkinter package in Python.
- **Translatev1/2.** This function is wholly encased in a “try–except” block which is a built-in function of Python which is designed for error handling. Any errors from any of the functions within this block result in the deletion of the data and, because of how this “try–except” block is configured, an error message gets displayed on the screen.
- **Ates.** The ates function first takes the files in input and retrieves the filename; then sends the files to the inputter function for parsing; then sends the parsed data to the inputter function to extract the necessary node information into lists, before finally sending the lists to the printer function which will write the files. The ates function will also take the value sent with file data from the translate functions, called pvalue, and will modify the new file name to differentiate between version one and version two if the pvalue is 2.
- **Inputter.** The inputter function parses the XML file using the defused XML ElementTree package built into Python. Using the defused XML package prevents code execution from maliciously crafted XML files. The function returns an XML document object model, or DOM, for use by the lister function.
- **Lister.** The lister function takes the XML DOM from the inputter function and extracts the necessary data from it. It does this by first creating empty lists, then finding the attributes to fill each list. It then returns the full lists to the ates function to be sent to the printer function.
- **Printer.** The printer function has a “try–except” block for creating a directory to store the output files from the conversion process. The files created by this function will also expect this directory to exist and will provide the user with the command to run if the files are in this directory. Next, the function will open the new .clf file in write mode, or create it in write mode if it does not exist, and write the JESS program code which has been written. If pvalue is 1 then the function will write necessary JESS code and the list data from lister into the same file, but, if pvalue is 2, then it will write an additional JESS code and create a new .dat file for containing the attack tree’s node data. All the files will be named automatically based upon the original input file name. Once these processes have been completed, the files will be saved and closed.

Successful attack tree conversion to JESS expert system knowledge base is demonstrated in Figure 13. Converted files are automatically exported to the directory “Attack_trees/JESS” in CLP format.

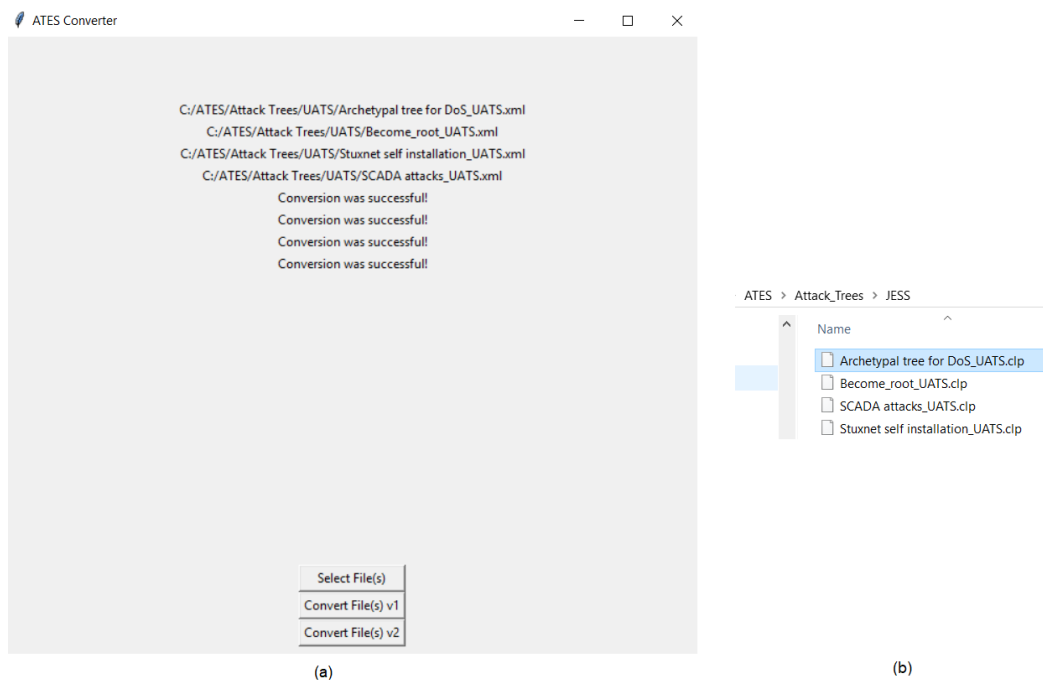


Figure 13. Example of (a) successful conversion to JESS expert system and (b) automatically exported clp knowledge base files.

- JESS ES file(s).** The produced JESS programs have an initial welcome message at the top of each file with instructions on where to place the files and with what JESS command to run the files. Successful JESS operation is shown in Figure 14.

```

JESS> (batch "Attack_Trees/JESS/RFID_Comm_Block_UATSv2.clp")
F-0 (MAIN::initial-fact)
F-1 (MAIN::program (phase initialization))
F-2 (MAIN::question (ident node-select) (text "which node ID would you like to select?"))
F-3 (MAIN::question (ident slot-select) (text "which slot would you like to modify? "))
F-4 (MAIN::question (ident modify-value) (text "what should the value be? ")) (answer n
F-5 (MAIN::question (ident main-menu) (text "what would you like to do? (l)ookup name,
F-6 (MAIN::tree (name "RFID Comm Block UATSv2") (rootnode "Block Communication"))
F-7 (MAIN::node (id 0) (name "Block Communication") (parent nil) (connector OR) (cost
rmeasure FALSE) (children 1 14))
F-8 (MAIN::node (id 1) (name "Block Tag Reader") (parent 0) (connector OR) (cost 0) (t
ure FALSE) (children 2 8 9 10))
F-9 (MAIN::node (id 2) (name "Shield Tag") (parent 1) (connector AND) (cost 0) (time 0
ALSE) (children 3 5))
F-10 (MAIN::node (id 3) (name "Be in Vicinity of Tag") (parent 2) (connector OR) (cost
ermeasure FALSE) (children 4))
F-11 (MAIN::node (id 4) (name "Secure Warehouse") (parent 3) (connector nil) (cost 0)
asure TRUE) (children nil))
F-12 (MAIN::node (id 5) (name "Faraday Cage") (parent 2) (connector OR) (cost 0) (time
FALSE) (children 6 7))
F-13 (MAIN::node (id 6) (name "Cage Around Reader") (parent 5) (connector nil) (cost 0
measure FALSE) (children nil))
F-14 (MAIN::node (id 7) (name "Cage Around Tag") (parent 5) (connector nil) (cost 0) (
sure FALSE) (children nil))
F-15 (MAIN::node (id 8) (name "Blocker Reader") (parent 1) (connector nil) (cost 0) (t
ure FALSE) (children nil))
F-16 (MAIN::node (id 9) (name "Blocker Tag") (parent 1) (connector nil) (cost 0) (time
FALSE) (children nil))
F-17 (MAIN::node (id 10) (name "Jam Signal") (parent 1) (connector OR) (cost 0) (time
FALSE) (children 11))
F-18 (MAIN::node (id 11) (name "Isolate Network") (parent 10) (connector OR) (cost 0)
asure TRUE) (children 12 13))
F-19 (MAIN::node (id 12) (name "Secure Warehouse") (parent 11) (connector nil) (cost 0)
measure TRUE) (children nil))
F-20 (MAIN::node (id 13) (name "Faraday Around Tag and Reader") (parent 11) (connector
LSE) (countermeasure TRUE) (children nil))
F-21 (MAIN::node (id 14) (name "Block Reader Backend") (parent 0) (connector OR) (cost
ermeasure FALSE) (children 15))
F-22 (MAIN::node (id 15) (name "Dos in Network") (parent 14) (connector nil) (cost 0)
asure FALSE) (children nil))
For a total of 23 facts in module MAIN.
what would you like to do? (l)ookup name, (m)odify data, (v)iew AT, (q)uit:
    
```

Figure 14. Generated rules were successfully imported into JESS.

The attack trees used for testing ATEs’ accuracy and capability for error handling have been sourced through numerous scientific articles covering various topics relating to attack trees, attack trees examples from attack tree tools, and results in Internet searches for attack tree examples. Additionally, the Mitre organization has a plethora of information which is easily turned into example attack trees, such as what is available within the CAPEC [38] and ATT&CK [39] databases. The citations for the

use of these attack trees have been listed beneath the root node of the attack tree as a comment within the pre-ATTop conversion XML file for that attack tree. Applicability of the method proposed for automated formation of an expert system’s knowledge base was evaluated and approved experimentally. Table 1 indicates the number of attack trees used in this process and their top-level sources. The uniqueness of the approach proposed is the provided possibility to generate integral knowledge base of an expert system, by integrating different attack trees taken from various sources.

In the experiment, 22 different free sources with attack trees were analyzed, and 49 different attack trees were transformed. One attack tree, in general, is dedicated to one main attack, so it allows ES to assess the risk for 49 different types of attacks. During the experiment, a ruleset composed of 875 facts was created. These facts allow us to assess the risk of 49 attacks and to analyze 826 controls (Figure 15). This can be considered as a significant achievement in the area of information security risk management, since effective and source-independent utilization of existing and reliable information sources and their adaption for use in expert systems creates preconditions for performing information security risk analysis even by non-security specialists.

Table 1. Attack tree sources.

Attack Tree Sources	Quantity of Attack Trees
A safety/security risk analysis approach of Industrial Control Systems: A cyber bowtie—combining new version of attack tree with bowtie analysis [40]	7
ADTool example attack trees [41]	10
Attack Modelling for Information Security and Survivability [22]	2
Attack tree-based evaluation of physical protection systems vulnerability [42]	1
Attack trees with sequential conjunction [43]	2
Automated Generation of Attack Trees by Unfolding Graph Transformation Systems [24]	2
Beyond attack trees : dynamic security modelling with Boolean logic Driven Markov Processes (BDMP) [44]	2
Bridging two worlds: Reconciling practical risk assessment methodologies with theory of attack trees [18]	1
Crowdsourcing Computer Security Attack Trees [36]	1
DAG-based attack and defense modelling: Don’t miss the forest for the attack trees [25]	1
Effective Analysis of Attack Trees: A Model-Driven Approach [45]	1
Is my attack tree correct? [34]	1
Mission oriented risk and design analysis of critical information systems [46]	1
Multi-vendor Penetration Testing in the Advanced Metering Infrastructure [47]	3
Producing and Evaluating Crowdsourced Computer Security Attack Trees [48]	1
Risk assessment of cyber attacks in ECPS based on attack tree and AHP [49]	1
SANS ICS Attack Surfaces [50]	1
Security risk assessment framework for smart car using the attack tree analysis [51]	5
Semi-automatically Augmenting Attack Trees using an Annotated Attack Tree Library [35]	2
Studying Cyber Security Threats to Web Platforms Using Attack Tree Diagrams [52]	1
Survivability analysis of SOA based on attack tree models [53]	1
Understanding risk through attack tree analysis [54]	2
The total quantity of sourced attack trees:	49

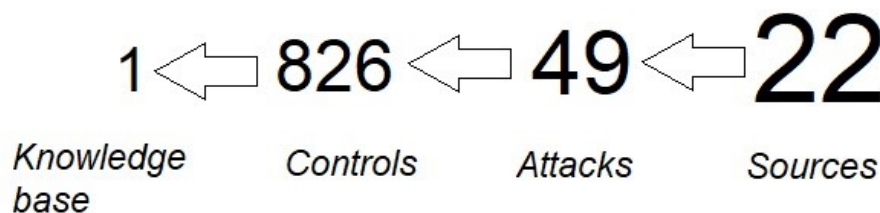


Figure 15. Summary of transformation results.

The experiment has shown that, out of approximately 50 attack trees, there were zero errors with the ATES conversion program when inputs were correctly configured. When a non-XML file is submitted, a syntax error is raised and the data are destroyed by the program. If, however, a file is in XML but does not contain the expected formatting, then a syntax error is displayed, and a file is

created with pythonic “none” data instead of the attack tree node data. Additionally, the resulting JESS expert systems were functioning as intended.

Development of an expert system, including development of an inference engine, was out of scope of this research. Still, in order to demonstrate that automatically generated rules are practically applicable, the earlier expert system prototype developed by our team and utilizing the OWASP Risk Rating Methodology [55] was used. The system prototype is JESS based and was presented in [10]. The generated rules, including information attack success rates, were imported from the tree, presented in Figure 16.

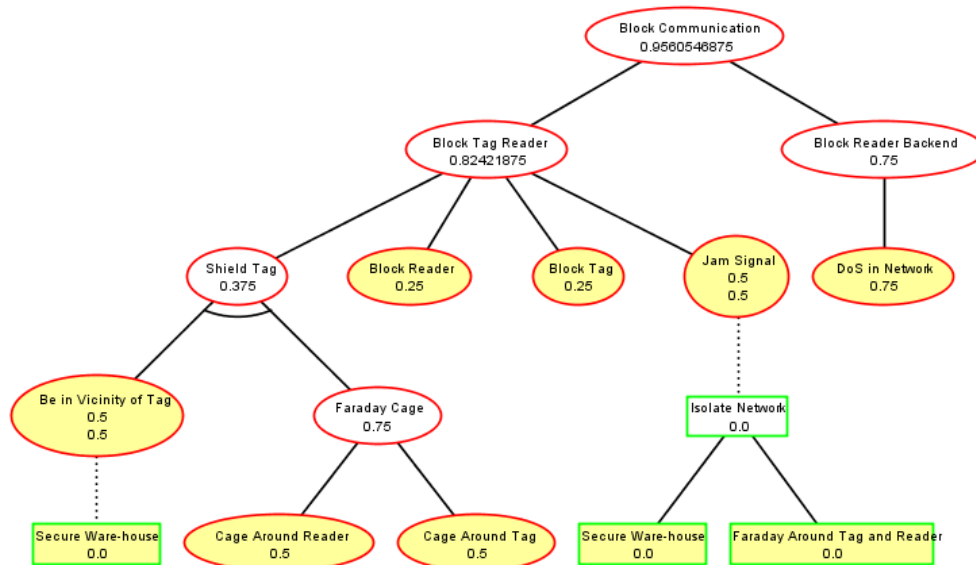


Figure 16. Attack tree: “How to block communication between an RFID reader and tag.”

This example utilizes the “How to block communication between an RFID reader and tag” attack tree. In each tree, information on attack probability success is stored. Nodes marked in red represent attack nodes, while green nodes define countermeasures, i.e., defense nodes. As stated earlier, this attack tree, stored in XML format, was converted using the proposed method and later imported into the existing expert system prototype (used for risk analysis). Impact estimation at the current step was obtained from the user input. Risk calculation was done by the classical equation: $risk = likelihood \times impact$. The sample of the risk evaluation obtained is presented on Figure 17.

```

*****
Technical Impact HIGH: 8.0
*****

*****
Business Impact MEDIUM: 5.0
*****

Probability of success: | 0.9560546875
Technical risk:         | 7.6484375
Business risk:         | 4.7802734375

Attack name:           | Block communication
Description:           | How to block communication between an RFID reader and tag
    
```

Figure 17. Example of the risk calculation.

The green color marks the impact, entered by the expert system user. It can be also imported from the attack tree if such information is included in it. The blue color denotes the likelihood that is imported from the attack tree. The likelihood can vary depending on the availability or absence of countermeasures. In this sample the likelihood is calculated for the case when no countermeasures are applied. The red color is used to highlight the risk, calculated by the expert system according to the equation provided above. Since the ES, evaluating both technical and business risks, was utilized for the experiments, they both can be seen on the screen.

The provided example should not be seen as finite or unconditional. It is just used to demonstrate that the rules generated by the method proposed can be integrated and used in the new or existing expert system dedicated for information security risk analysis. The knowledge base created can be utilized not only for the risk calculation, but also for evaluating other criteria and estimations important in the risk assessment process. The method proposed can transform and import any kind of information stored in a tree, such as likelihood, price of the attack, weight coefficients, impacts, and others. Moreover, information imported can be used for detecting the most probable attack vector or, as shown on Figure 16, for making suggestions of possible controls and countermeasures. Still, this is already a part of the specific expert system decision making process, which is out of scope of this research, and can be considered as a perspective research topic.

5.1. Current Issues and Controls Applied

As with any methodology, there are some limitations and issues. Below the short descriptions of such issues are provided with comments on the controls that were or could be applied to minimize their influences on the quality of results:

- One potential issue of the method proposed is that there is a small possibility that data are altered during the transformation processes. To resolve this issue, the data were visually compared during the heuristic phase from the input to the resulting files to verify the accuracy.
- Another weakness of the proposed method is that the output data will only be as reliable as the input data. That is to say that, if an inaccurate attack tree is converted to the expert system rules, then the expert system will not provide accurate information regarding any eventual risk assessments. There is, ultimately, not any way around this issue beyond double-checking that the attack trees are accurate, come from the reliable sources, and apply to the system in question. This will require that users of this method would have some knowledge of their systems and that they are receiving accurate attack tree information.
- Potentially, the greatest issue in this method lies in the extreme variability within XML. This has been mitigated somewhat by using the output from ATTop to standardize the expected input to ATEs. However, the naming conventions of various tools being capable of significant differences for the same attributes (e.g., the attribute identifier “duration” referring to the length of time required for an attack could also be called “time” or even “length” by specific tools) means that the only solution is to code each tool’s naming convention into the lister function.

5.2. Advantages of the Methodology

The main advantages of the method proposed are related to the flexibility in data processing and output:

- Due to the popularity and ease of use of the Python programming language, it is relatively easy to increase the acceptable attack tree formats by determining the attribute identifiers used within that tool’s XML format. Utilizing an “if-then” statement within the lister function would allow for the conversion of these attack trees as well.
- The configuration of version two allows for a relatively small set of changes which would mean a single JESS expert system could read the .dat files for multiple attack trees. As the data are already in a separate file, it would be possible to select multiple .dat files to minimize consumed

storage space on a device. This was not pursued by the authors due to a lack of necessity in modern systems.

- Finally, there are several points where data can be configured by the user, and it will be accepted throughout. It is possible to set the attribute values either before translation in ATTop, after translation from ATTop and before input to ATES, after output from ATES by manually editing the data files, or inside the JESS program created by ATES. The most efficient and accurate method is to either modify the final JESS .dat files or the XML files after being output from ATTop.

5.3. Future Work

Further research on the methodology improvement will be concentrated on:

- Incorporation of PDF files as a source of attack tree;
- Minimizing the time for the file inputting process by creating the automated input file batch process;
- Resolving the issues related to the variability of attack tree representation and naming formats.

6. Conclusions

The analysis performed has shown that current methods of attack tree analysis involve human experts performing expert analyses of attack trees and the systems being utilized by a company to determine the best method of defense for that system or company. The vast amounts of data about possible attacks against information systems collected in the form of attack trees can be seen as prospective sources of data for the automatic creation of knowledge bases for expert systems, dedicated for IT security risk analysis, thereby minimizing the knowledge base creation process's expenses and ensuring process reliability.

A method was proposed that utilizes several tools: already existing ATTop, which performs the initial conversion of attack tree data into the unified UATS XML format, and the newly developed ATES tool, which performs further transformation into the CLIPS rules supported by the JESS expert system platform, which was chosen for the experiments due to its broad adoption.

The experiments performed have covered data importation from attack trees from 22 different sources and allowed creating the knowledge base that could be used to assess risks of 49 cyber attacks and propose over 826 possible controls. The method has demonstrated reliability while converting attack trees of a supported type. Throughout the heuristic portion of experiments, the data conversion correctness was confirmed to be valuable by JESS and was tested on the JESS platform to verify that accidental attack tree data modification will not occur during any phase of the translation process. Still, the recent trends in attack tree representation, transformation, and generation show a large amount of variability in attack tree representation and generation. This variability leads to some difficulties in the transformation that could require further research.

Author Contributions: All authors contributed to investigation, methodology, software, measurements, data analysis, scientific discussions and writing the article. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This research received no support which is not covered by the Author Contributions and Funding sections.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Janulevičius, J.; Čenys, A. Development of a Risk Assessment Model for IT Risk Self-Assessment Expert System for SMEs. *Int. J. Comput. Commun. Eng.* **2016**, *3*, 306–309. [[CrossRef](#)]
2. Abraham, A. Rule-Based Expert Systems. In *Handbook of Measuring System Design*; John Wiley and Sons: New York, NY, USA, 2005.

3. Akerkar, R.A.; Sajja, P.S. *KnowledgeBased Systems*; Jones & Bartlett Publishers: Toronto, ON, Canada, 2010.
4. Batista, L.O.; de Silva, G.A.; Araujo, V.S.; Araujo, V.J.S.; Rezende, T.S.; Guimaraes, A.J.; Souza, P.V.D.C. Fuzzy neural networks to create an expert system for detecting attacks by sql injection. *Int. J. Forensic Comput. Sci.* **2018**, *1*, 8–21. [[CrossRef](#)]
5. Dheir, I.; Abu-Naser, S.S. Knowledge Based System for Diagnosing Guava Problems. *Int. J. Acad. Inf. Syst. Res. (IJAIRS)* **2019**, *3*, 9–15.
6. Atymtayeva, L.; Kozhakhmet, K.; Bortsova, G. Building a Knowledge Base for Expert System in Information Security. In *Soft Computing in Artificial Intelligence; Advances in Intelligent Systems and Computing Series.*; Springer: Cham, Switzerland, 2014; pp. 57–76.
7. Ivkin, A.N.; Burlakov, M.E. Realization of expert intrusion detection system based on the results of datasets and machine learning algorithm analysis. *Prikaspijskij Zurnal Upravelnije Vysok. Technol.* **2020**, *2*, 100–107.
8. Pinto, F.J. Application of the Bayesian Model in Expert Systems. In Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, Ávila, Spain, 26–28 June 2019; Springer: Cham, Switzerland, 2019; pp. 117–124.
9. Vitkus, D.; Steckevičius, Ž.; Goranin, N.; Kalibatiénė, D.; Čenys, A. Automated Expert System Knowledge Base Development Method for Information Security Risk Analysis. *Int. J. Comput. Commun. Control* **2019**, *14*, 743–758. [[CrossRef](#)]
10. Vitkus, D.; Jezukevičiūtė, J.; Goranin, N. Dynamic Expert System-Based Geographically Adapted Malware Risk Evaluation Method. *Int. J. Comput. Commun. Control* **2020**, *15*. [[CrossRef](#)]
11. Kim, D.; Shin, D.; Shin, D.; Kim, Y.H. Attack detection application with attack tree for mobile system using log analysis. *Mob. Netw. Appl.* **2019**, *24*, 184–192. [[CrossRef](#)]
12. Kornecki, A.J.; Liu, M. Fault tree analysis for safety/security verification in aviation software. *Electronics* **2013**, *2*, 41–56. [[CrossRef](#)]
13. Rios, E.; Rego, A.; Iturbe, E.; Higuero, M.; Larrucea, X. Continuous Quantitative Risk Management in Smart Grids Using Attack Defense Trees. *Sensors* **2020**, *20*, 4404. [[CrossRef](#)]
14. Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of artificial intelligence adversarial attack and defense technologies. *Appl. Sci.* **2019**, *9*, 909. [[CrossRef](#)]
15. Ingoldsby, T.R. *Attack Tree-Based Threat Risk Analysis*; Amenaza Technologies Limited: Calgary, AB, Canada, 2010; pp. 3–9.
16. Lohner, B. Attack-Defense-Trees and other Security Modeling Tools. *Network* **2018**, *97*, 97–103.
17. Kordy, B.; Kordy, P.; Mauw, S.; Schweitzer, P. ADTool: Security analysis with attack-defense trees. In *Quantitative Evaluation of Systems; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin, Germany, 2013; pp. 173–176. [[CrossRef](#)]
18. Gadyatskaya, O.; Harpes, C.; Mauw, S.; Muller, C.; Muller, S. Bridging two worlds: Reconciling practical risk assessment methodologies with theory of attack trees. *Artif. Intell. Lect. Notes Bioinform.* **2016**, *9987*, 80–93. [[CrossRef](#)]
19. Shostack, A. *Threat Modeling: Designing for Security*; Wysopal, C., Ed.; John Wiley & Sons, Inc.: Indianapolis, IN, USA, 2014.
20. Wenli, S.; Tianyu, G.; Chunyu, C.; Jing, H.; Peng, Z. Information Security Risk Assessment Method for Ship Control System Based on Fuzzy Sets and Attack Trees. *Secur. Commun. Netw.* **2019**. [[CrossRef](#)]
21. Fila, B.; Wideł, W. Exploiting attack–defense trees to find an optimal set of countermeasures. In Proceedings of the 2020 IEEE 33rd Computer Security Foundations Symposium (CSF), Boston, MA, USA, 22–26 June 2020; pp. 395–410.
22. Moore, A.P.; Ellison, R.J.; Linger, R.C. *Attack Modelling for Information Security and Survivability*; Carnegie Mellon University, Software Engineering Institute: Pittsburgh, PA, USA, 2001.
23. Schneier, B. Attack Trees. *Dr. Dobb's J.* **1999**, *24*, 21–29.
24. Huistra, D.J. Automated generation of Attack Trees by Unfolding Graph Transformation Systems. Master's Thesis, University of Twente, Enschede, The Netherlands, 2016.
25. Kordy, B.; Pietre-Cambacedes, L.; Schweitzer, P. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Comput. Sci. Rev.* **2014**, *13–14*, 1–38. [[CrossRef](#)]
26. Saini, V.; Duan, Q.; Paruchuri, V. Threat modeling using attack trees. *J. Comput. Sci. Coll.* **2008**, *23*, 124–131.

27. Salter, C.; Saydjari, O.S.; Schneier, B.; Wallner, J. Toward a secure system engineering methodology. In Proceedings of the 1998 Workshop on New Security Paradigms—NSPW '98, Charlottesville, VA, USA, 22–25 September 1998; pp. 2–10. [CrossRef]
28. Sheyner, O.; Wing, J. Tools for generating and analyzing attack graphs. In Proceedings of the International Symposium on Formal Methods for Components and Objects, Leiden, The Netherlands, 4–7 November 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 344–371. [CrossRef]
29. Gadyatskaya, O.; Jhawar, R.; Kordy, P.; Lounis, K.; Mauw, S.; Trujillo-Rasua, R. Attack trees for practical security assessment: Ranking of attack scenarios with ADTool 2.0. In Proceedings of the International Conference on Quantitative Evaluation of Systems, Glasgow, UK, 10–12 September 2016; pp. 159–162.
30. Vigo, R.; Nielson, F.; Nielson, H.R. Automated generation of attack trees. In Proceedings of the 2014 IEEE 27th Computer Security Foundations Symposium, Vienna, Austria, 19–22 July 2014; pp. 337–350.
31. Wang, P.; Lin, W.H.; Kuo, P.T.; Lin, H.T.; Wang, T.C. Threat risk analysis for cloud security based on Attack-Defense Trees. In Proceedings of the 2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT), Seoul, Korea, 24–26 April 2012; Volume 1, pp. 106–111.
32. Pinchinat, S.; Acher, M.; Vojtisek, D. Towards synthesis of attack trees for supporting computer-aided risk analysis. In Proceedings of the International Conference on Software Engineering and Formal Methods, Amsterdam, The Netherlands, 14–17 September 2014; pp. 363–375.
33. RFID Communication Block. Security and Trust of Software Systems. Available online: https://satoss.uni.lu/projects/atrees/trees/block_communication.pdf (accessed on 14 April 2020).
34. Audinot, M.; Pinchinat, S.; Kordy, B. Is my attack tree correct? In *Computer Security—ESORICS 2017*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2017; pp. 83–102. [CrossRef]
35. Jhawar, R.; Lounis, K.; Mauw, S.; Ramírez-Cruz, Y. Semi-automatically Augmenting Attack Trees Using an Annotated Attack Tree Library. In *Security and Trust Management*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2018; pp. 85–101. [CrossRef]
36. Tentilucci, M.; Roberts, N.; Kandari, S.; Johnson, D.; Bogaard, D.; Stackpole, B.; Markowsky, G. Crowdsourcing Computer Security Attack Trees. In Proceedings of the 10th Annual Symposium on Information Assurance (ASIA'15), Albany, NY, USA, 2–3 June 2015; pp. 19–23.
37. Attack Tree Translator and Analyzer ATTop. University of Twente—Formal Methods and Tools. Available online: <https://github.com/utwente-fmt/attop> (accessed on 18 March 2020).
38. Common Attack Pattern Enumeration and Classification. Mitre. Available online: <https://capec.mitre.org> (accessed on 20 April 2020).
39. Knowledge Base of Adversary Tactics and Techniques. Mitre. Available online: <https://attack.mitre.org> (accessed on 20 April 2020).
40. Abdo, H.; Kaouk, M.; Flaus, J.M.; Masse, F. A safety/security risk analysis approach of Industrial Control Systems: A cyber bowtie—Combining new version of attack tree with bowtie analysis. *Comput. Secur.* **2018**, *72*, 175–195. [CrossRef]
41. Library of Trees. Security and Trust of Software Systems. Available online: <https://satoss.uni.lu/projects/atrees/library.php> (accessed on 14 April 2020).
42. Vintr, Z.; Valis, D.; Malach, J. Attack tree-based evaluation of physical protection systems vulnerability. In Proceedings of the 2012 IEEE International Carnahan Conference on Security Technology (ICCST), Boston, MA, USA, 15–18 October 2012; pp. 59–65. [CrossRef]
43. Jhawar, R.; Kordy, B.; Mauw, S.; Radomirović, S.; Trujillo-Rasua, R. Attack trees with sequential conjunction. *JIFIP Adv. Inf. Commun. Technol.* **2015**, *455*, 339–353. [CrossRef]
44. Pietre-Cambacedes, L.; Bouissou, M. Beyond attack trees: Dynamic security modeling with Boolean logic Driven Markov Processes (BDMP). In Proceedings of the 2010 European Dependable Computing Conference, Valencia, Spain, 28–30 April 2010; pp. 199–208.
45. Kumar, R.; Schivo, S.; Ruijters, E.; Yildiz, B.M.; Huistra, D.; Brandt, J.; Stoelinga, M. Effective Analysis of Attack Trees: A Model-Driven Approach. In *Fundamental Approaches to Software Engineering*; Russo, A., Schurr, A., Eds.; Springer: Cham, Switzerland, 2018; pp. 56–73. [CrossRef]
46. Buckshaw, D.L.; Parnell, G.S.; Unkenholz, W.L.; Parks, D.L.; Wallner, J.M.; Saydjari, O.S. Mission oriented risk and design analysis of critical information systems. *Mil. Oper. Res.* **2005**, *10*, 19–38. [CrossRef]

47. McLaughlin, S.; Podkuiko, D.; Miadzvezhanka, S.; Delozier, A.; McDaniel, P. Multi-vendor penetration testing in the advanced metering infrastructure. In Proceedings of the 26th Annual Computer Security Applications Conference, Austin, TX, USA, 6–10 December 2010; pp. 107–116. [[CrossRef](#)]
48. Bogaard, D.; Johnson, D. Producing and Evaluating Crowdsourced Computer Security Attack Trees. In Proceedings of the 2016 IEEE Symposium on Technologies for Homeland Security (HST), Waltham, MA, USA, 10–11 May 2016; pp. 1–4. [[CrossRef](#)]
49. Ru, Y.; Wang, Y.; Li, J.; Liu, J.; Yang, G.; Yuan, K.; Liu, K. Risk assessment of cyber attacks in ECPS based on attack tree and AHP. In Proceedings of the 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD 2016), Changsha, China, 13–15 August 2016; pp. 465–470. [[CrossRef](#)]
50. Cole, E. SANS ICS Attack Surfaces. Available online: <https://www.sans.org/summit-archives/file/summit-archive-1493739527.pdf> (accessed on 14 April 2020).
51. Kong, H.K.; Hong, M.K.; Kim, T.S. Security risk assessment framework for smart car using the attack tree analysis. *J. Ambient Intell. Humaniz. Comput.* **2018**, *9*, 531–551. [[CrossRef](#)]
52. Petrica, G.; Axinte, S.; Bacivarov, I.C.; Firoiu, M.; Mihai, I.C. Studying Cyber Security Threats to Web Platforms Using Attack Tree Diagrams. In Proceedings of the 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Targoviste, Romania, 29 June–1 July 2017; pp. 1–6.
53. Xiao, Y.; Wang, Y.J.; Huang, Z.G. Survivability analysis of SOA based on attack tree models. In Proceedings of the International Conference on Communication Technology Proceedings, Chengdu, China, 9–11 November 2012; pp. 819–823. [[CrossRef](#)]
54. Ingoldsby, T.R. Understanding risk through attack tree analysis. *Comput. Secur. J.* **2004**, *20*, 33–59.
55. OWASP Risk Rating Methodology. Available online: <https://owasp.org/www-project-risk-assessment-framework/> (accessed on 17 February 2020)



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).