A Method of Threat Analysis for Cyber-Physical System using Vulnerability Databases

Yusuke Mishina, Kazuo Takaragi
Information Technology Research Institute (ITRI)
Advanced Industrial Science and Technology (AIST)
Annex, 2-3-26 Aomi, Koto-ku
Tokyo 135-0064, Japan
{yusuke.mishina, kazuo.takaragi}@aist.go.jp

Katsuyuki Umezawa
Department of Information Science
Shonan Institute of Technology
1-1-25 Tsujido-Nishikaigan, Fujisawa
Kanagawa 251-8511, Japan
umezawa@info.shonan-it.ac.jp

***Abstract*—Safety and security are major issues for cyber-physical systems. We propose a threat analysis method effective for the design stage of a safety-critical cyber-physical system, utilizing the fact that similar systems tend to have co-occurrence in the way of their chain in vulnerability. We first utilize a vulnerability database to express known cyber-attack cases on the Fault Tree-Attack Tree (FT-AT). Second, the method uses FT-AT as a kind of teacher data for similar system threat analysis and uses it to find new attacks in similar systems. This makes it possible to efficiently support system design that tries to implement safety and security. The usefulness of the approach is demonstrated by example applications to previously reported attacks on Tesla and Cherokee.**

***Index Terms*—Threat Analysis, Vulnerability Information, Attack Tree**

I. Introduction

Security issues are hot topics, and discussions in the United States about CSF [1] and the like are progressing to concretize requirements and to consider appropriate regulations to ensure security. As a basic countermeasure for them, considering security by design is required. In other words, when designing a system, it is required to estimate the threats that may occur within the assumed product lifetime, incorporate priorities, and include countermeasures in the design specification.

Interference and interruption to safety because of security threats are recognized as a big problem in safety critical systems such as those for electric power, automobiles, aviation, railways, and medical care. Regarding the security of in-vehicle communication in the EVITA project [2], risk analysis, security requirement settings, architecture design, and prototyping and demonstration of HSM by FPGA were conducted. An Attack tree was used for risk analysis in the EVITA project. One way to analyze the causal relationship between safety (hazard) and security (threat) is to express that relationship with a combination of Fault tree (FT) and Attack tree (AT) [3].

The MITER Corporation in the US provides several forms of vulnerability databases. In CVE (Common Vulnerability and Exposure) [4], individual software vulnerabilities are stored in a database. In CWE (Common Weakness Enumeration) [5], common vulnerabilities are cataloged focusing on the cause of the vulnerability.

However, more than 10,000 vulnerabilities have been reported in CVE since 2017. For many of them we need deep insight and tremendous effort to determine whether a new vulnerability can be chained with the remaining vulnerabilities or normal functions in another system and lead to a new attack. It is not easy to create AT that comprehensively captures those possibilities. Furthermore, in safety-critical cyber-physical systems, the problem of mutual interference between safety and security has not been sufficiently analyzed.

This paper proposes a threat analysis method that is effective for solving such problems.

II. Proposed method

A. Outline of our proposed method

The scientific literature related to safety analysis using FT is mature today [3]. On the other hand, in security analysis, the complexity of the problem is

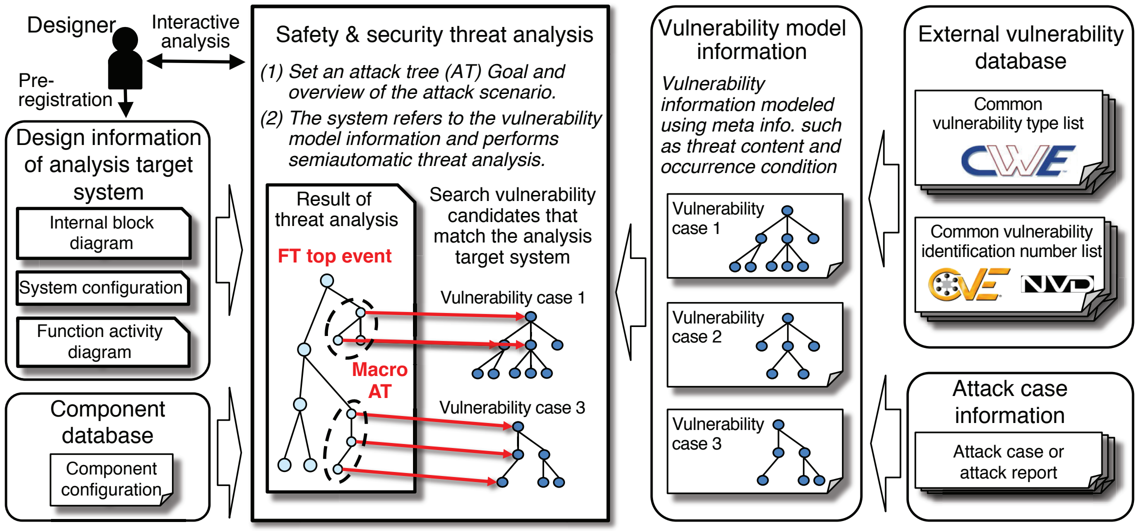

Fig. 1. Overview of proposed method

significantly increased. In addition, elaborate attacks occur with multiple combinations of those vulnerabilities. Furthermore, it is not easy to create AT that comprehensively captures their possibilities.

In this paper, we focus on such problems and propose a threat analysis method based on the following characteristics as a practical approach.

(a) The defenders, that is, the designers, have limitations on how to protect systems. That is, under circumstances where a lot of new vulnerabilities are announced each year, it is practically difficult to check each time whether a new vulnerability can be chained with remaining vulnerabilities or normal functions in the system and lead to a new attack.
(b) Attackers have a tendency to attack. Many attacks are imitations of known attacks and minor changes. Here we calls this a related attack, and it is assumed that it is the substantial cause of the risk increase.
(c) Including these trends in $FT-AT$ can be a useful starting point for analysis.
(d) Expressing cases that occurred in the past with $FT-AT$ makes it possible for the designer to recognize related attacks (recognize the danger).
(e) Continuous application of this approach gradually helps to reduce risk. It should be noted, however, that this approach does not guarantee the discovery and prevention of sophisticated new attacks that are not related attacks.

The overview of the above proposed method is shown in Fig. 1.

B. Creating vulnerability model information

The MITRE Corporation has published several forms of vulnerability databases [4] [5]. For each vulnerability, we will create a rough AT with reference to such databases and previous literature of attack cases. Thus, let AT obtained in such a way be called the first AT (hereinafter, referred to as AT^1). One AT^1 is created for each vulnerability. Although this work volume is large, it can be done efficiently using natural language processing and AI technology.

C. Proposal of component database

In some configurations of embedded systems such as those used in automobiles and IoT devices, COTS applications are not used as they are, and only subprograms of the applications are embedded as needed. On the other hand, a vulnerability database such as CVE describes vulnerability information for certain software, but it does not necessarily refer to the information of subprograms in the software. Therefore, a correspondence table between the software version and the version of the subprograms of the software as shown in Figure 2 would help. This makes it easy to check vulnerability information during the manufacturing of embedded devices.

Figure 2 shows an example of Tesla browser hacking. As shown on the left side of Figure 2, the CVE Detail provided by NVD contains only the version of Chrome including vulnerable WebKit. By referring to the correspondence table, it is possible to predict the vulnerability of Tesla' s browser using vulnerable WebKit.

Version	Release date	Layout engine
…	…	…
11.0.696	2011-04-27	WebKit 534.24
12.0.742	2011-06-07	WebKit 534.30
13.0.782	2011-08-02	WebKit 535.1
14.0.835	2011-09-16	WebKit 535.1
15.0.875	2011-10-25	WebKit 535.2
16.0.912	2011-12-13	WebKit 535.7
17.0.963	2012-02-08	WebKit 535.11
…	…	…

Vulnerability CVE-2011-3928 in Google Chrome before 16.0.912.77

Tesla's browser Mozilla / 5.0 (X11; Linux) Apple WebKit / 534.34

Fig. 2. Example of Component Database

D. Detailed analysis

Next, a detailed threat analysis is given based on the design detail of the target system in addition to the vulnerability model shown in Section II-B and the component database shown in Section II-C.

(1) Designers create a second AT (hereinafter, referred to as AT^2) with a top event (which can be a troublesome accident, safety accident) of the target system (Fig. 3 (s2)). Designers describe the outline of the attack tree in accordance with the attack scenario that they can think of.

(2) The algorithm compares the AT^1 and the AT^2 for each vulnerability (Fig. 3 (s3)). Natural language processing and AI processing are used to detect whether there is a top event of the AT^1 that is close to the intermediate event of the AT^2 [1]. If there is a close one, the AT^1 is ORed (Fig. 3 (s4)). (There is a possibility in terms of words that its vulnerability may cause the failure of the target system.)

(3) Paying attention to the intermediate node of the ORed tree, the algorithm judges whether the intermediate node should be deleted from the AT^2 (different components' versions or contradiction of attacks' consequences) or not. Specifically, it is assumed that the FALSE node is a node unrelated to the component of the AT^2, and the relation of the FALSE node and the relation of the above node of the AND relation, which is just above the FALSE node, are deleted (Fig. 3 (s5)).

(4) This process is repeated for all the AT^1s for one AT^2. Then we evaluate the occurrence probability of the top event of the final AT^2 (Fig. 3 (s6)).

[1]For example, detecting that the same word "executes arbitrary code" appears in the uppermost node of Figure 5 and in the middle node in the third row in Figure 6.

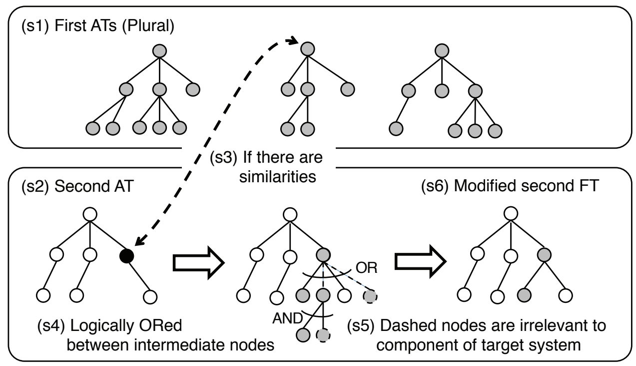

Fig. 3. Threat analysis algorithm

III. Formulation of proposed algorithm

Next, we formulate the algorithm shown in Section II-D.

A. Definition

The notation of the attack tree AT according to reference [3] is shown as follows.

$$\boldsymbol{G} = \{g_i\} : AttackGoals \quad (1)$$
$$\boldsymbol{O} = \{o_i\} : Operations \quad (2)$$
$$\boldsymbol{AS} = \{as_i\} : Assertions \quad (3)$$
$$\boldsymbol{V} = \{v_i\} : Vulnerabilities \quad (4)$$
$$\boldsymbol{R} = \{r_i\} : Relationships \quad (5)$$

An example of AT is shown in Fig. 4.

Here, an attack goal is the goal of all potential cyber attacks, and operations represent all the basic actions (reads, writes, etc.) that can be performed by either the attacker or the operator of the system. An assertion is a statement (for example, "Web server is not patched") representing "conditions to be verified" in order to take account of the actual branching of the attack tree. Vulnerability is a known vulnerability. Relationships are relationships that exist between elements that make up an attack tree (that is, attack goals, operations, assertions, vulnerabilities). The attack tree AT_k is defined as follows.

$$AT_k = \{g_i, \boldsymbol{O}_i, \boldsymbol{AS}_i, \boldsymbol{V}_i, \boldsymbol{R}_i\} \tag{6}$$

Here, $g_i \in \boldsymbol{G}, \boldsymbol{O}_i \subseteq \boldsymbol{O}, \boldsymbol{AS}_i \subseteq \boldsymbol{AS}, \boldsymbol{V}_i \subseteq \boldsymbol{V}, \boldsymbol{R}_i$ is a set of relationships.

All AT has one main goal g, and the output (upper side) of the logic gate becomes an assertion.

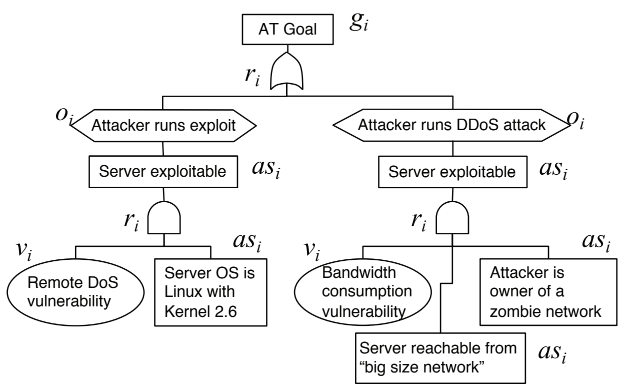

Fig. 4. Example of AT (Quoted from Fig. 2 in reference [3])

B. Formulation of proposed algorithm

The first attack tree AT_k^1 and the second attack tree AT^2 are defined as follows.

$$AT_k^1 = \{g_k, \boldsymbol{O}_k, \boldsymbol{AS}_k, \boldsymbol{V}_k, \boldsymbol{R}_k\} \tag{7}$$
$$AT^2 = \{g_j, \boldsymbol{O}_j, \boldsymbol{AS}_j, \boldsymbol{V}_j, \boldsymbol{R}_j\} \tag{8}$$

Next, look for k, which is $g_j = g_k$ or $as_l \approx g_k$, where $as_l \in \boldsymbol{AS_j}$. In addition, look for n and m, which is $as_n \approx as_m$, where $as_n \in \boldsymbol{AS_k}, as_m \in \boldsymbol{AS_j}$.

Here, $x \approx y$ means "Comparing the descriptions of both sides with words, it is judged that x and y are close."

Next, update the second attack tree AT^2 as follows.

$$\begin{aligned} AT^2 = \{ & g_j, \boldsymbol{O}_j \cup \boldsymbol{O}_k \cup \boldsymbol{O}_n, \boldsymbol{AS}_j \cup \boldsymbol{AS}_k \cup \boldsymbol{AS}_n, \\ & \boldsymbol{V}_j \cup \boldsymbol{V}_k \cup \boldsymbol{V}_n, \boldsymbol{R}_j \cup \boldsymbol{R}_k \cup \boldsymbol{R}_n \backslash \boldsymbol{R}'\} \end{aligned} \tag{9}$$

Here, $X \backslash Y$ represents the set of elements in X but not in Y.

Also, $\boldsymbol{R}'$ is $\boldsymbol{R}' = \boldsymbol{R}'_{OR} \cup \boldsymbol{R}'_{AND}$. $\boldsymbol{R}'_{OR}$ is the relationship of the FALSE node, and $\boldsymbol{R}'_{AND}$ is the relationship of the upper nodes of the AND relationship just above the FALSE node.

Here, FALSE means "The vulnerability in question has no mechanical chain to the target event."

A FALSE node is $o \in \boldsymbol{O}_k \cup \boldsymbol{O}_n, as \in \boldsymbol{AS}_k \cup \boldsymbol{AS}_n, v \in \boldsymbol{V}_k \cup \boldsymbol{V}_n$ that is unrelated to the components of AT^2 (such as different components and different versions).

C. Calculation of attack probability [3]

According to the formulation in the previous section, it is possible to calculate the probability of attack with the following formula using the calculation method of the conventional research [3].

If the inputs to the logic gates are independent, the probability of the output value from the ith AND gate $P_{out}AND_i$ and the probability of the output value from the ith OR gate $P_{out}OR_i$ are as follows.

$$P_{out}AND_i = \prod_{k=1}^{n} P_{in}(k,i) \tag{10}$$

$$P_{out}OR_i = \sum_{k=1}^{n} P_{in}(k,i) \tag{11}$$

However, $P_{in}(k,i)$ is the probability of the input of the kth input to the ith gate with n inputs ($1 \leq k \leq n$).

In addition, in reference [3], calculation formulas when the inputs to the logic gates are not independent are also shown.

Furthermore, reference [3] suggests rewriting the operation node with an AND gate and an assertion in order to obtain the probability of the top event (attack goal) of the attack tree. As a result of rewriting, the description of the operation disappears from the attack tree, and the probability of the top event can be calculated by sequentially calculating the above expression (10) and expression (11).

IV. Application of Proposed Method to Actual Case

In this section, we apply the proposed algorithm of Section II-D to examples of Tesla [9] and Cherokee [10].

A. Application to Tesla case

In September 2016, Tencent used a plurality of vulnerabilities of the Tesla Model S to invade the in-vehicle system via WiFi, injected a malicious CAN message into the CAN bus, and caused malfunction of the vehicle.

1) Create the first attack tree AT^1: We create the first AT (hereinafter, referred to as AT^1) on the basis of the existing vulnerability database and concrete attack case. The more the AT^1s are created, the more the threat analysis algorithm can predict many attacks. As an example, Figure 5 shows a AT^1 of the CVE-2011-3928 that is actually used in the Tesla hack. Attacks on this vulnerability cause "Execution of Arbitrary Code" or "Denial of Service" in specific versions of web browsers.

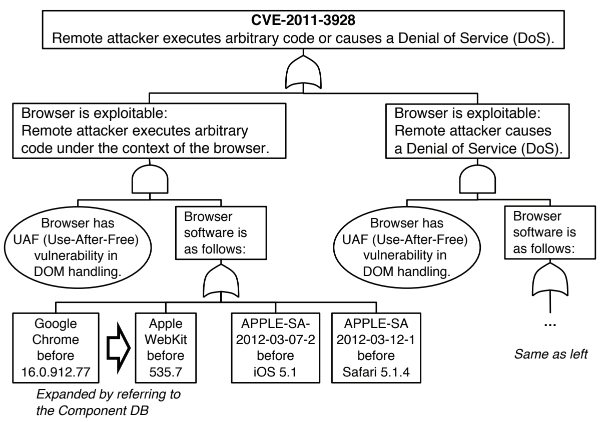

Fig. 5. First AT of CVE-2011-3928

2) Create the second attack tree AT^2: We construct the second AT (hereinafter, referred to as AT^2) on the basis of the vulnerability database (AT^1) and the component database. A designer sets an attack goal and creates an outline of the attack scenario he/she can think of. Figure 6 shows an AT^2 of the Tesla hack. The top node shows an attack goal, and the second row shows the attack scenario across 4 function blocks of the Tesla connected vehicle system. The algorithm splits the outline of the attack scenario into atomic actions and compares AT^1 with these actions. If there are similar nodes, AT^1 is combined with AT^2. By referring to the Component Database, overlooking the vulnerability CVE-2011-3928 of the Google browser that contains WebKit is avoided. Finally, the node indicating the component not used by Tesla is deleted from AT^2.

Figure 7 shows a threat analysis result of the attack on Tesla. For each software module allocated in the above-described four functional blocks, the vulnerability included in each module and the flow of data at the time of attack execution are shown. As a result of the cyber attack, it is understood that the boundary between the information system displaying the web information outside the vehicle and the information system controlling the in-vehicle equipment is broken.

B. Application to Cherokee case

Our proposed algorithm can also be applied to the remote hacking case against Jeep Cherokee published by C. Miller et al. in 2016. Like the Tesla system, the Cherokee system is composed of four functional units: cellular phone, CID, Communication Gateway, and ECU. Cherokee uses the "Uconnect System" for CID. Just searching for "Uconnect System" in the vulnerability database results in only one hit. By referring to the component database, it is found that D-bus is configured as interprocess communication software of the Uconnect System. By doing this, searching the vulnerability database with "D-bus" reveals many vulnerabilities and led to the discovery of the vulnerability "opened to anonymous" exploited this time.

In addition, let's assume that the case of Tesla is a case that occurred before this Cherokee analysis[2]. The AT^2 obtained in Section IV-A2 can then be used as a candidate AT^1 to be used in Cherokee's case.

V. Considerations

As seen in the case of Tesla this time, the attacker performs attacks on WiFi (V1), attacks on browsers (V2), and attacks on console displays (V3) in order from the outside of the defense. These V1 and V2, V2 and V3 are high-frequency attacks that appear in combination with the attack "to make abnormal operation of the connected car." Hereinafter, such an attack is called a "co-occurrence attack." In addition, in the case of the Jeep Cherokee, an attack is also carried out on the mobile phone network

[2]In fact, the attack on Cherokee was carried out before that of Tesla.

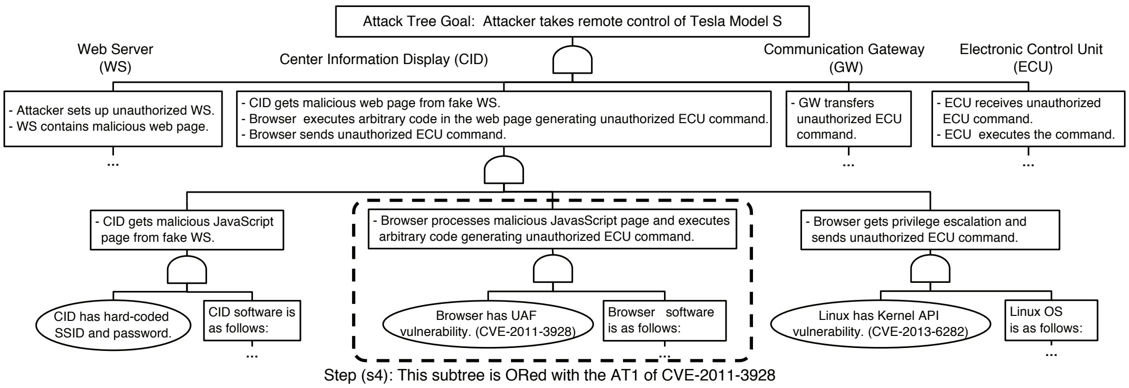

Fig. 6. Second AT of Tesla (only relevant parts)

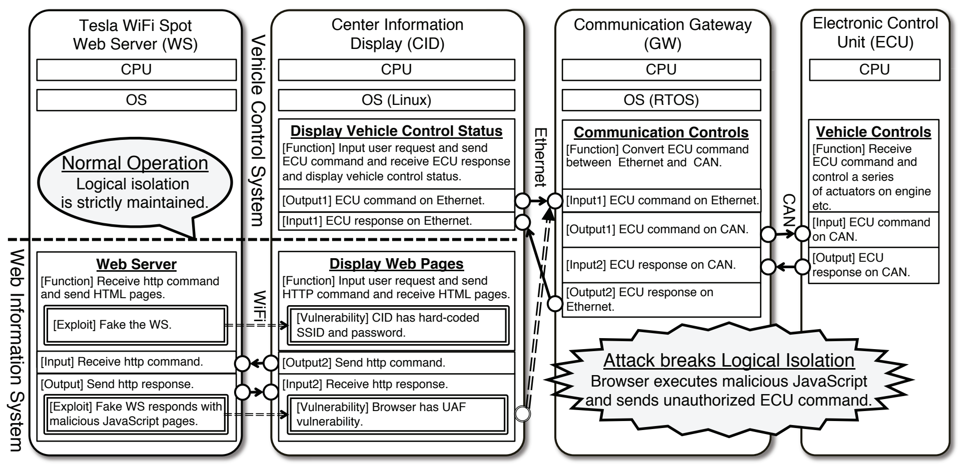

Fig. 7. Threat Analysis Result of Attack on Tesla

(V1') and the console display (V3'), and V1' and V3' are co-occurrence attacks. We consider co-occurrence attacks such as (V1 to V2 to V3) and (V1' to V3') on the basis of the Tesla and Cherokee cases so that they can be used for unknown cases, and we think they should be described at the higher concept level.

In this paper, we propose an analysis method based on the hypothesis that the same kind of vulnerability is easily exploited in attacks against different objects (e.g., cars). A series of attack chains of the Cherokee case actually appear as a chain of attacks of the Tesla case.

Furthermore, the cyber-attack chains discovered in this way will be expressed on FT – AT, which integrates safety and security aspects, and analysis will continue. In the Cherokee case, defending security infringement with a safety mechanism was reported. The mutual interference is analyzed on the FT – AT as the security infringement depicted in AT and as a consideration of the safety mechanism depicted in FT. Thus, the proposed threat analysis system executes a process of learning attacks of similar systems from past attack cases as teacher data on the premise that a series of attacks having co-occurrence will occur.

VI. Conclusion

In general, creating abstract and appropriate meta-information applicable to vulnerable attacks on similar systems requires deep insight and effort on security. However, in this method, as the created tree of cases of Tesla and Cherokee can be used as the first tree of future analysis, the more useful the database is, the more useful it is to add more attack cases for related attacks. Therefore, we believe that this method is promising in analyzing safety and security.

Acknowledgments

This work was supported by Council for Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP), “Cyber-Security for Critical Infrastructure” (funding agency: NEDO).

References

[1] NIST, ”Framework for Improving Critical Infrastructure Cybersecurity Version 1.1,” https://www.nist.gov/cyberframework, April 2018.

[2] A. Ruddle, et al., ”Deliverable D2.3: Security requirements for automotive on-board networks based on dark-side scenarios,” Seventh Research Framework Programme of the European Community, July 2008.

[3] I. N. Fovino, et al., ”Integrating cyber attacks within fault trees,” Reliability Engineering and System Safety 94 (2009) p.p.1394–1402.

[4] MITRE Corporation, ”CVE - Common Vulnerability and Exposure,” https://cve.mitre.org/

[5] MITRE Corporation, ”CWE List - Common Weakness Enumeration,” https://cwe.mitre.org/data/

[6] A. Applebaum, D. Miller, B. Strom, C. Korban and R. Wolf, ”Intelligent, automated red team emulation,” ACSAC ’16 Proceedings of the 32nd Annual Conference on Computer Security Applications, December 05 - 08, 2016, pp 363-373.

[7] Antti Levomaki, Olli-Pekka Niemi, Christian Jalio, ”Automatic Discovery of Evasion Vulnerabilities Using Targeted Protocol Fuzzing,” Briefing, Black Hat Europe 2017, Dec. 2017.

[8] Geoffrey Biggs, et al, ”A profile and tool for modelling safety information with design information in SysML,” Software & Systems Modeling 15, 1 (Jan 2016), pp147–178.

[9] Sen Nie, et al., ”FREE-FALL: HACKING TESLA FROM WIRELESS TO CAN BUS,” Briefing, Black Hat USA 2017, July 2017.

[10] C. Miller and C. Valasek. ”Remote Exploitation of an Unaltered Passenger Vehicle,” Briefing, Black Hat USA 2015, pp 1–91.