2011 Third International Workshop on Security Measurements and Metrics

Security Risk Management by Qualitative Vulnerability Analysis

Golnaz Elahi
University of Toronto
gelahi@cs.toronto.edu

Eric Yu
University of Toronto
yu@ischool.utoronto.ca

Nicola Zannone
Eindhoven University of Technology
n.zannone@tue.nl

***Abstract*—Security risk assessment in the requirements phase is challenging because risk factors, such as probability and damage of attacks, are not always numerically measurable or available in the early phases of development. This makes the selection of proper security solutions problematic because mitigating impacts and side-effects of solutions are not often quantifiable. In the early development phases, analysts need to assess risks in the absence of numerical measures or deal with a mixture of quantitative and qualitative data. We propose a risk analysis process which intertwines security requirements engineering with a vulnerability-centric and qualitative risk analysis method. The proposed method is qualitative and vulnerability-centric, in the sense that by identifying and analyzing common vulnerabilities the probability and damage of risks are evaluated qualitatively. We also propose an algorithmic decision analysis method that considers risk factors and alternative security solutions, and helps analysts select the most cost-effective solution. The decision analysis method enables making a decision when some of the available data is qualitative.**

I. Introduction

Absolute security is impossible. Security is more a cat and mouse game between the defenders and attackers [1]; hence, it is ultimately about mitigating the damage of attacks or increasing the effort and time required for successful exploitations. Requirements analysts and project leaders need to have an objective measure of security risks to select good-enough security countermeasures. They also need to understand consequences of different alternative countermeasures on the probability and potential damage of attacks to decide over alternatives.

The concept of risk: The risk (r) of a security attack is described by the level of damage (d) that the successful attack poses to the system and the probability (p) that the attack occurs: $r = p \times d$ [2]. A security solution may mitigate the damage to a lower level (d') or reduce the probability of an attack to a lower chance (p'), so the risk is reduced to $r' = p' \times d'$. If the damage is expressed in terms of financial costs, then the risk represents the financial damage of the attack. In this way, managers can easily decide whether to apply a security solution, by examining if the solution's cost (c) is lower than the financial damage that they can save ($c < r - r'$). When managers need to decide over a set of alternative solutions, they can select a solution which minimizes the costs while mitigating the risks ($min(c + r' - r)$).

Risk assessment challenges: In an ideal situation, security risks are known, their damage is measurable in terms of financial costs, and the probability of different attacks can be estimated based on the history of previous attacks. Ideally, the mitigating impacts of security solutions are quantifiable with respect to probability p and damage d. However, in practice, quantifying the damage of attacks, estimating attacks probability, and quantifying the mitigating impacts of countermeasures is challenging, error-prone, and involves collecting subjective (and inaccurate) opinion of security experts, and mining previous attacks repositories (which may not be available).

Some requirements or risk factors can be refined into measurable variables, but many non-functional requirements and security metrics have a soft nature, which makes them hard, if not impossible, to measure. Stakeholders may choose to evaluate some factors and requirements numerically, and some requirements qualitatively, on different scales, e.g., absolute values, percentages, ordinal numbers, or qualitative labels. When different requirements are measured (or estimated) on different scales, normalizing inconsistent types of measures into a single scale is troublesome and may not result in a meaningful utility value.

Vulnerabilities analysis challenges: The other problematic side of security requirements engineering is that attackers usually try to get one step ahead of defenders. Therefore, software developers and managers always need to anticipate potential threats that may not have happened previously. In this condition, one source of deriving potential threats is identifying and analyzing vulnerabilities within the system. A vulnerability is a weakness of an asset [2], flaw, or error in the specification or implementation which can be exploited to break into the system [3].

Some vulnerabilities, such as buffer over-flow, are common and well-known to the security community; thus, common mitigations are usually available to reduce the risk of these vulnerabilities. However, the damage and probability of exploitations, even for commonly-known vulnerabilities, are not always quantifiable or available. Vulnerability scoring frameworks such as Common Vulnerability Scoring System (CVSS) [4] provides a framework for evaluating the criticality of vulnerabilities by qualitative metrics. However, vulnerability analysis in general and CVSS specifically, are not widely adopted in security requirements engineering.

978-0-7695-4680-3/12 $26.00 © 2012 IEEE
DOI 10.1109/Metrisec.2011.12

Every day, new vulnerabilities are discovered, while requirements analysts are not able to objectively evaluate the dimension and extent of the damage that can be caused by the exploitation of such vulnerabilities. Nonetheless, preventing the introduction of vulnerabilities will eventually be more economical than removing them after the system has been compromised.

The other major problem is that vulnerabilities are usually flaws, bugs, and errors at the code level, and associating such low-level weaknesses to requirements satisfaction and design objectives is not straightforward. There is a gap between relating immediate effects of a vulnerability exploitation on the running system to ultimate impacts of this exploitation on requirements and high-level business goals.

Summary of problems: Software developers and even security experts face several challenges to assess security risks and select proper countermeasures to satisfy security requirements. A part of these problems are due to the lack of knowledge about vulnerabilities and exploitation scenarios. Another problem stems from the lack of analysis methods:

1) An objective way to measure the probability of exploitations.
2) An objective way to measure the degree of damage that exploitations can cause, and relating it to high-level requirements and goals.
3) An objective way to accurately estimate mitigating consequences (and side-effects) of security solutions.
4) A systematic decision making method that considers risks, impacts of solutions, and stakeholders' preferences to select good-enough security solutions.

The main contribution of this paper is a vulnerability-centric risk assessment method, supported by a qualitative decision aid algorithm for selecting proper countermeasures for critical risks:

Vulnerability-Centric Risk Assessment: In this method, we identify security risks by detecting and analyzing vulnerabilities within the system. We overview common knowledge sources to identify relevant vulnerabilities, and propose a way to evaluate impacts of vulnerability exploitations on high-level goals of stakeholders. We employ the security modeling notation in [5] to analyze vulnerabilities within system requirements and interactions. This notation extends the *i** Framework [6] with the concepts of vulnerability, exploitation, and attack, and helps reason about the damage caused by vulnerability exploitations on requirements and goals. We explain how to incorporate the CWE vulnerabilities into these models (CWE is used as an example of empirical vulnerability knowledge portals).

In this work, we intertwine the CVSS framework with risk assessment, in the sense that the probability and damage of exploitations scenarios on stakeholders' goals are estimated by CVSS vulnerability criticality metrics. We change some of the CVSS metrics for assessing the risks of attacks in untrusted environments. Then, the mitigating impacts of security solutions on the damage and/or probability of attacks are analyzed. Security solutions may have side-effects on other requirements, for instance, may reduce performance and/or threaten privacy. Both financial costs and negative impacts on other requirements are considered in the final decision over alternative solutions.

Decision Analysis: We present a decision analysis algorithm that helps analysts select the best security solution considering costs and benefits of alternatives and preferences of stakeholders over different requirements. This is a challenging decision, specially when numerical risk and mitigation factors are not completely available. The major obstacles in making decisions over security solutions and requirements are:

1) Extracting stakeholders' preferences over multiple criteria in terms of numerical importance weights is time-consuming and error-prone.
2) Different requirements and alternatives can be evaluated in different scales of measurement. Thus, the decision must be made in the presence of mixture of qualitative and qualitative measures.
3) Making an objective decision often requires collecting extensive information from decision stakeholders.
4) Manual decision analysis over numerous requirements and/or alternatives is labor-intensive and error-prone.

The proposed decision analysis method in this paper tackles above mentioned challenges. We adopt the Even Swaps multi-criteria decision analysis approach [7] for deciding over alternative solutions. The Even Swaps process helps make trade-offs between decision criteria by trading one requirement for another; thus, it does not require eliciting importance weights of requirements and risks. Requirements and risks can be evaluated in a mixture of scales and by different measurement methods.

Organization: Section II presents an overview of the risk and decision analysis process. Section III introduces a general Digital Rights Management scenario to illustrate the proposed framework. Section IV presents a qualitative risk assessment method which focuses on identifying and evaluating vulnerabilities. Section V presents a decision analysis method for deciding over alternative security solutions. Related work and existing approaches are reviewed in Section VI and some conclusions are drawn in Section VII.

II. Overview

Figure 1 shows an overview of the risk assessment and decision analysis process. This section overviews the steps of the process. In a previous work [5], we introduced a modeling notation and process for embedding the knowledge of vulnerabilities into *i** goal-oriented models of software requirements [6]. We use these models to evaluate security risks and select security solutions among alternatives for mitigating the risks:

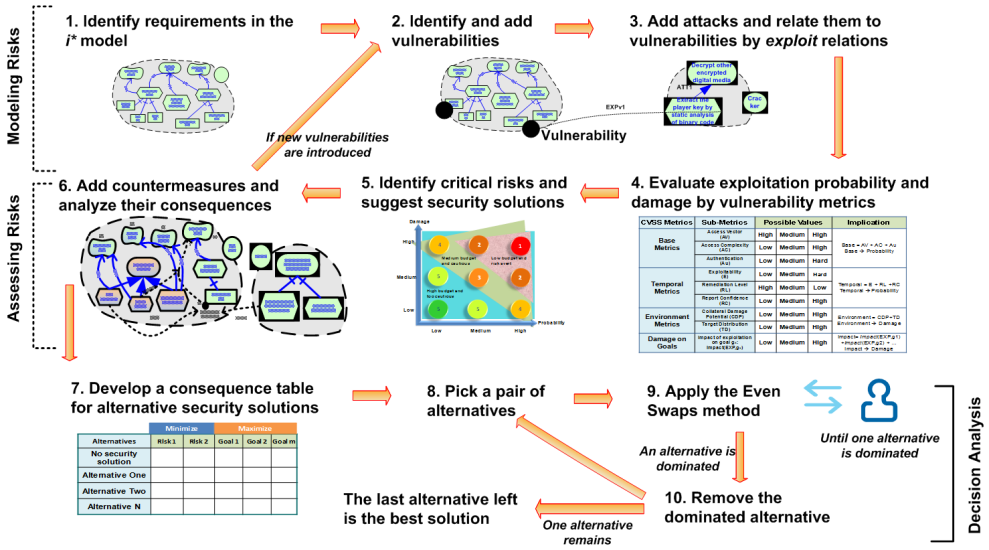


Figure 1. The risk assessment and decision analysis process overview

- *Step 1*: Understand stakeholders' goals and system requirements by developing an *i** model of the system and its main actors.
- *Step 2*: Identify relevant vulnerabilities in empirical knowledge portals and add them to the system model.
- *Step 3*: Identify potential attacks that may exploit the vulnerabilities and analyze the impacts of these attacks on stakeholders' goals and system requirements.
- *Step 4*: Evaluate the probability and damage of vulnerability exploitations by analyzing vulnerability metrics.
- *Step 5*: Identify critical risks that threaten important goals, and suggest security solutions for those risks.
- *Step 6*: Identify security solutions and analyze consequences of solutions on risks and stakeholders' goals.
- *Step 7*: Aggregate consequences of security solutions into a table, called *consequence table*.
- *Step 8*: Select a pair of alternative solutions for each cycle of the decision analysis process.
- *Step 9*: Suggests a chain of goals trading to stakeholders to find the preferred alternative in the pair.
- *Step 10*: Find the dominated alternative (the weaker in the comparison), and remove it from the list of solutions.

The decision analysis algorithm (Steps 7 to 10) is iterative: it selects another pair of alternatives to compare and a new cycle starts. These cycles continue until one alternative remains, which is the best solution overall.

III. Digital Right Management Illustrative Example

We illustrate the vulnerability-centric risk assessment by analyzing a prototypical Digital Rights Managements (DRM) player [1]. The player gets an encrypted media, and given a valid user key, decrypts the media, and decodes the digital content to an audio. The user needs to purchase an activation code for the player, and the player only works if the activation code is valid at the time of using the player.

Although vulnerabilities within DRM systems in general are known and various security mechanisms have been developed to protect digital content and DRM, selecting good-enough and cost-effective solutions need the assessment and judgement of security experts. Throughout this paper, we illustrate how the proposed method helps analyze DRM player risks and select proper countermeasures. The target users of the proposed method are requirements analysts which may not have extensive security knowledge.

IV. Security Risk Assessment by Vulnerability Analysis

This section describes steps 1 to 6 in Figure 1. First we discuss how to elicit risks by analyzing vulnerabilities. Then, we adopt and adapt the Common Vulnerability Scoring System (CVSS) for evaluating vulnerabilities and risks. Finally, we present a risk management method to decide on critical risks and identify proper solutions. The final output of these activities is a set of risks, their probability and damage, a set of solutions and their consequences on risk factors and their side-effects on other requirements.

A. Identifying Relevant Vulnerabilities

Initially, analysts need to identify relevant risks in the target domain and technology. Thus, the risk assessment process requires understanding the domain through modeling and analyzing stakeholders' goals and system requirements. By understanding the system through developing *i** models, important goals that can be threaten by security risks are identified, and vulnerabilities are linked to entities that can introduce vulnerabilities to the system. The extended *i** modeling notation with security concepts [5] helps express potential effects of the vulnerabilities on stakeholders' goals or system requirements.

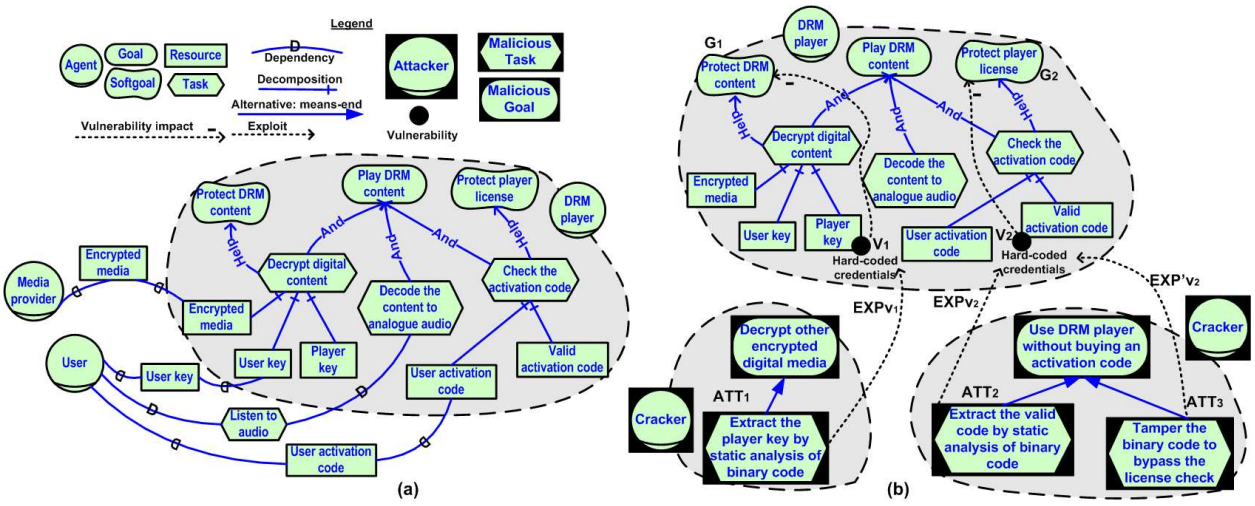


Figure 2. The *i** model of a Digital Right Management player, vulnerabilities, and possible attacks

Eliciting Relevant Vulnerabilities from CWE: Common Weakness Enumeration (CWE) provides a categorization of vulnerabilities, such as authentication issues, buffer errors, code injection, etc. CWE provides a set of software weaknesses in source code and operational systems as well as weaknesses related to architecture and design. The CWE catalogue includes abstract weaknesses, errors, and vulnerabilities that are independent of specific platforms or technologies. This makes CWE a suitable starting point to identify relevant vulnerabilities in a given goal model.

The *i** model makes it possible to link identified vulnerabilities to requirements. Every goal, action (tasks in the i* notation), and resource within the system can potentially introduce a new vulnerability. To detect what vulnerability are relevant, the analyst needs to search the CWE catalogue for relevant terms expressed in the model.

For example, the *i** model in Figure 2(a) shows how the DRM player works. CWE does not explicitly discuss any vulnerability related to `(Valid) activation code` or `Player key`. However, these keys are actually *hard-coded* in the DRM player. In CWE, the weakness ID-798, "Use of Hard-coded Credentials", states that software that contains hard-coded credentials, such as a password or cryptographic key, leads to a significant authentication failure. In the context of the DRM player, two instances of such hard-coded credentials are `Valid activation code` and `Player key`.

Common Attack Pattern Enumeration and Classification (CAPEC) lists possible attacks that exploit CWE vulnerabilities. Figure 2(b) shows some attacks that can exploit the vulnerability of hard-coded credentials. For example, a software cracker can `Use DRM player without buying activation code` either by `Extracting the valid code by static code analysis` or `Tampering the binary code to bypass the license check`.

B. Adapting CVSS for Vulnerability-Centric Risks Assessment

CVSS is an open framework developed by National Institute of Standards and Technology for assessing vulnerabilities criticality across many disparate hardware and software platforms. CVSS in general helps IT managers to prioritize vulnerabilities and suggest solutions for remediating those that pose high risks [4].

CVSS is composed of three metric groups: Base, Temporal, and Environmental, each consisting of a set of metrics. Table I summarizes the vulnerability assessment metrics that are adopted in this work.

In the original CVSS, these metrics groups measure the criticality of vulnerabilities. For the purpose of risk assessment, we use Base and Temporal metrics to assess how hard is exploiting the vulnerabilities, which ultimately indicates how probable the exploitations are. Environmental metrics and negative impacts of exploitation on goals (as additional metrics) are used to evaluate the damage of exploitations.

CVSS provides guidelines to evaluate these metrics objectively as well as equations for evaluating the Base, Temporal and Environmental metric groups and producing an score between 0 and 10 for each of the metric groups [4]. In this adaptation of CVSS, we evaluate and aggregate the metrics on the scale of Low, Medium, High to simplify the evaluation process in the early requirements stage, where more detailed numerical data are not available. Nevertheless, analysts can still apply the proposed method by using any scale of measurement on which they can accurately evaluate vulnerability metrics. The proposed decision analysis method in Section V enables analyzing risk factors regardless of the evaluation scale being used.

1) Base Metrics: Base metrics represent the intrinsic and fundamental characteristics of a vulnerability that are constant over time and environments. Access Vector (AV) metric in the Base group reflects how the vulnerability is

Table I
ADOPTING AND ADAPTING CVSS METRICS FOR ASSESSING RISK FACTORS

	CVSS Metrics	Sub-Metrics	Possible Values			Implication
Evaluating Probability	Base Metrics	Access Vector (AV)	Locally **(High)**	Adjacent Network **(Medium)**	General network **(Low)**	Base = AV + AC + Au Base → Probability
		Access Complexity (AC)	**Low**	**Medium**	**High**	
		Authentication (Au)	No authentication **(Low)**	Single authentication **(Medium)**	Multiple authentications **(High)**	
	Temporal Metrics	Exploitability (E)	Unproven method **(High)**	Proof-of-Concept **(Medium)**	Functional method **(Low)**	Temporal = E + RL +RC Temporal →Probability
		Remediation Level (RL)	Unavailable fix **(Low)**	Temporary Fix or Workaround approach **(Medium)**	Official Fix **(High)**	
		Report Confidence (RC)	**Low**	**Medium**	**High**	
Evaluating Damage	Environment Metrics	Collateral Damage Potential (CDP)	**Low**	**Medium**	**High**	Environment = CDP+TD Environment → Damage
		Target Distribution (TD)	Low percentage **(Low)**	Medium percentage **(Medium)**	High percentage **(High)**	
	Damage on Goals	Impact of exploitation on goal g_x: Impact(EXP,g_x)	**Low**	**Medium**	**High**	Impact= *Impact*(EXP,g_1) +*Impact*(EXP,g_2) + … Impact → Damage

exploited: Locally, through an Adjacent Network, or general Network Access. A local access is demanding, while through general network is easier and more probable (Table I). Access Complexity (AC) metric measures the complexity of the attack to exploit the vulnerability. Authentication (Au) metric measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability: Multiple, Single, None.

2) Temporal Metrics: Temporal metrics represent the characteristics of a vulnerability that change over time but not among environments. Exploitability (E) measures the current state of exploit techniques or code availability: Unproven, Proof-of-Concept, and Functional code or method. With an unproven exploitation method, the effort needed to successfully exploit the vulnerability would be high, while a functional and available exploitation method reduces the required effort (Table I). Remediation Level (RL) can be Official Fix, Temporary Fix/Workaround approach, and Unavailable fix to patch the vulnerability. When there exists an official fix for the vulnerability, the needed effort to exploit to vulnerability is high. Finally, Report Confidence (RC) measures the degree of confidence in the existence of the vulnerability and the credibility of the technical details collected.

3) Environmental Metrics: Environmental metrics represent the characteristics of a vulnerability that are relevant and unique to a particular environment. Among environmental metrics, Collateral Damage Potential (CDP) measures the potential for loss of life or physical assets or revenue through damage or theft of property or equipment. Target Distribution (TD) is an indicator to approximate the percentage of systems that could be affected by the vulnerability.

4) Vulnerability Impact Metrics: To assess the risks of vulnerabilities we add a tailored metric to the CVSS metric groups: the damage of vulnerability exploitation on stakeholders' goals and system requirements. For example, in the scenario in Figure 2(b), the damage of attack ATT_1 exploiting v_1 (EXP_{v_1}) on goal G_1 is High. This helps customize the CVSS metrics per project and in terms of goals that are of importance.

5) Tailoring CVSS for security in un-trusted environment: Potential attacker has full access to the hardware and the binary executable of a software application that resides in an un-trusted environment, and can inspect, reverse engineer, and tamper the application being run in the un-trusted environment. For example, game consoles, mobile applications, entertainment content, and any other application that is being run at the client side, reside in un-trusted environments, and potential attackers have full access to the hardware and application binary in the memory.

CVSS metrics are not designed considering attacks on the binary code in an un-trusted environment. For example, Access Vector values from the Base group are either locally, through an adjacent network, or through a global network. While for the case of attacks in un-trusted environments, the access to the binary code is already provided, and the attacker can even own the device or software. The Authentication Metric (Au) in the Base group is irrelevant as well, since attackers usually have full access to the system that they try to crack and authentication is not required. In case of attacks in un-trusted environments, the size of the application binary can affect effort needed to crack the application (See Table II).

In the Environmental metrics, Target Distribution is expressed as the percentage of damage distribution, while in case of attacks in un-trusted environments, the important factors is how many copies of illegal content or application are distributed (See Table II).

Table III summarizes the evaluation of vulnerability metrics for EXP_{v_2} in the DRM player scenario. The DRM

Table II
TAILORING CVSS METRICS FOR ASSESSING RISKS OF ATTACKS AGAINST THE BINARY CODE IN UN-TRUSTED ENVIRONMENTS

	CVSS Metrics for Un-trusted Environment	Sub-Metrics	Possible Values		
Probability	Base Metrics	Binary Size	Large (High)	Medium (Medium)	Small (Low)
		Binary Code Complexity	Low	Medium	High
	Temporal Metrics	Same as other types of attacks (Table I)			
Damage	Environment Metrics	Collateral Damage Potential	Same as other types of attacks (Table I)		
		Number of copies distributed	Nearly all revenue space (High)	Considerable revenue space (Medium)	Insignificant revenue space (Low)
	Damage on Goals	Same as other types of attacks (Table I)			

player is an example of attacks in un-trusted environments. For example, the Binary Size of the player is small (Low), and its complexity is Medium. By exploiting this vulnerabilities, a considerable revenue space will be affected by the (illegal) distribution of copies.

Table III
THE VULNERABILITY METRICS FOR ONE OF THE RISKS IN THE DRM PLAYER SCENARIO

	Metrics	Value for $EXPv_1$	Aggregated Value
Probability	Binary Size	Small (Low)	l = # Low = 2 m = # Medium = 3 h = # High = 0 $m > l + h$ thus a = Medium Probability = Medium
	Binary Code Complexity	Medium	
	Exploitability (E)	Functional method (Low)	
	Remediation Level (RL)	Workaround (Medium)	
	Report Confidence (RC)	Medium	
Damage	Collateral Damage Potential (CDP)	Medium	Max(Medium, Medium, High) = High Damage = High
	Number of copies distributed	Considerable revenue space (Medium)	
	Impact of exploitation on goal G_1: Impact($EXPv_1$,G_1)	High	

Aggregating Probability Metrics: Once the Base and Temporal metrics are evaluated, the assessments are aggregated into one value of probability score. The metrics and impacts are evaluated by ordinal scores of Low, Medium, High. The average (mean) of ordinal values is mathematically meaningless, but the mode is a valid value for describing the central tendency of ordinal measures.

Given a set of ordinal values $V = \{v_1, v_2, ..., v_n\}$, where a value, v_x, can be Low, Medium, or High, l, m, and h are the number of metrics in the set V which are Low, Medium, High respectfully. All of the CVSS factors in Base and Temporal groups (e.g., access, authentication, and exploitability) must be present for a successful exploitation. The central tendency of these values indicates the difficulty of exploiting the vulnerability overall.

To find the central tendency (a), the following rules are applied:

Rule 1: if $l \geq h + m$ then a = Low
Rule 2: if $h \geq l + m$ then a = High
Rule 3: if $m \geq l + h$ then a = Medium

In this way, the aggregation of metrics is the value that the majority of metrics have. If none of Low, Medium, High values is the majority of values, the Medium is taken. The higher the Base and Temporal metrics are, the lower the probability of the exploitation is, because Base and Temporal metrics measure the capabilities and effort needed to exploit the vulnerability, thus

If a = Low then probability = High
If a = Medium then probability = Medium
If a = High then probability = Low

For example, the third column of Table III shows the aggregation of vulnerability metrics to calculate the probability and damage scores for EXP_{v_1}. For the Base and Temporal metrics, $l = 2$, $m = 3$, $h = 0$, Medium is the majority of metrics value; thus the aggregation of Base and Temporal metrics is Medium, and the probability of EXP_{v_1} is Medium.

Assessing Damage Metrics: The higher the Environmental metrics and exploitation impacts on goals are, the higher the damage is. If even one aspect of the damage is High, the risk ultimately is high impact, and thus, we take the worst case scenario (pessimistic) as such:

Let $d_1, d_2, ..., d_n$ be the values of damage metrics
Damage = $Max(d_1, d_2, ..., d_n)$

For example, in Table III, damage would be $Max(Low, Medium, High) = High$.

Discussion: As discussed in Section I, measuring (or estimating) the probability and impacts of vulnerability exploitations in numerical measures is often challenging, or unreliable, even if possible. Evaluating these factors require subjective opinion of stakeholders and domain experts while they usually cannot provide explicit rationale for their evaluations. In this section, we introduced a risk assessment method which helps stakeholders and analysts systematically reason about the damage and probability of vulnerability exploitations in terms of vulnerability criticality metrics. Although analysts and stakeholders' (subjective) judgement is ultimately polled to evaluate CVSS metrics, the proposed method breaks down the risk value into fine-grained vulnerability metrics, instead of directly asking for a high-level risk value for each vulnerability. The breakdown provides measurable variables which helps analysts justify and support their judgement objectively.

C. Risk Treatment

1) Risk Ranking: Requirements analysts may identify several vulnerabilities and potential risks that require employing remediation. Some of the risks are insignificant and thus negligible. To identify critical risks, which require employing security solutions, we suggest a risk ranking approach which is tailorable according to the available budget and risk-taking attitude of analysts and project owners.

Figure 3 shows the ranking of risks according to their probability and damage. When the budget is low and project owners are risk-avert, only the top-ranked risks can be analyzed. If project owners are cautious and the budget is enough, less critical risks can be considered as well.

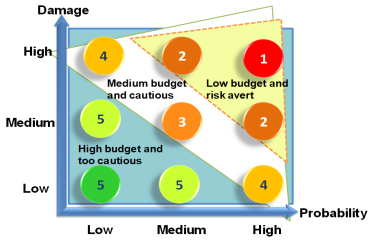


Figure 3. Risk ranking guidance

For example, in Figure 2(b), we identified two main vulnerabilities and three potential exploitation scenarios. EXP'_{v_2} is the most critical risk, and EXP_{v_1} and EXP_{v_2} are less critical:

Risk	Probability	Damage	Risk Rank
EXP_{v_1}	Medium	High	2
EXP_{v_2}	Medium	Medium	3
EXP'_{v_2}	Medium	High	2

Let us assume that the DRM player project does not have enough security budget and the product owners take risks, thus we only focus on the risk of EXP'_{v_2} and EXP_{v_1} (`Tampering the binary code to bypass the license checks` and `Extracting the player key by static analysis of binary code`).

2) Identifying and Analyzing Security Solutions: In the final step of the risk analysis process, requirements analysts brainstorm alternative security solutions for critical risks and re-evaluate risks with the presence of each solution.

Identifying Alternative Security Solutions: To protect the system against an attack or vulnerability exploitation, often alternative protection mechanisms are available. Each alternative has some advantages and disadvantages. For example, in the DRM player scenario, the main protection strategies against `Static code analysis and tampering` attacks are [1]:

1) `Code Obfuscation`: By obfuscating a program, the code is transformed into a form that is more difficult for an adversary to understand or change than the original code. Obfuscation adds an overhead to the code which causes performance drop downs.
2) `Tamper Proofing`: Even with a strong obfuscation, crackers may be able to reverse engineer the code and try to modify the program. Tamper proofing algorithms detect that the program has been modified (usually by computing a checksum over the code and comparing it to an expected value). Once tampering has been detected, a tamper response is executed which usually causes the program to fail.
3) `Distribution with Physical Token`: Physical tokens are hardware-based protections that try to provide a safe environment for data, code and execution. By employing a physical token, the user needs to show possession of a token to use the software.

Figure 4(a) shows these alternative security solutions and their side-effects on a number of factors such as portability, delay, and cost. The model also depicts that by applying `Code Obfuscation`, the vulnerability of `Hard-coded credentials` gets patched (although hard-coded credentials are still in the code, they are obfuscated and thus hard to find). By `Tamper Proofing` the impacts of the vulnerability exploitation are alleviated. Finally, the use of `Physical Tokens` forces the `Cracker` to `Clone existing physical tokens` in addition to successful tampering, which increases the effort for a successful attack dramatically. Modeling and analyzing the mitigating impacts of solutions later provide explicit rationale for analysts to re-assess the probability and damage of vulnerability exploitations with the presence of security solutions.

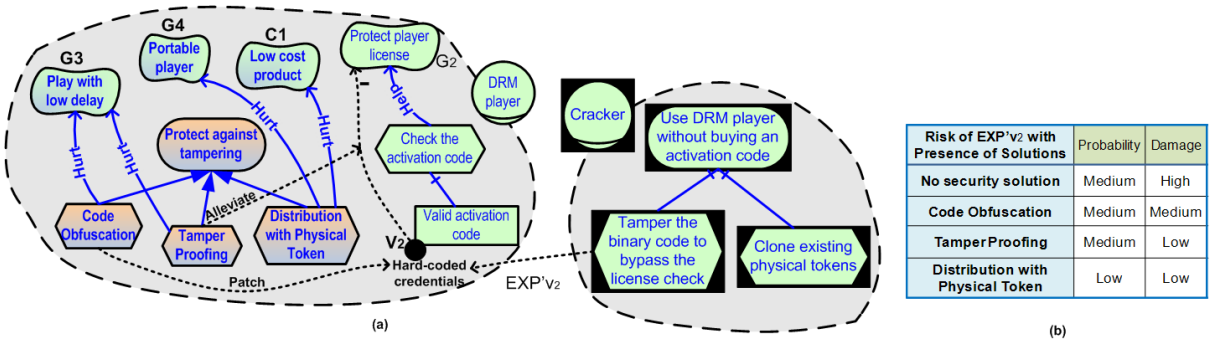
Re-Assessing Risks with the Presence of Security Solutions: Once alternative security solutions are identified and analyzed, their mitigating effects on the probability and damage of exploitations are evaluated. The risk assessment process using the CVSS metrics in Table I is repeated with the presence of security solutions. For example Figure 4(b) shows the risk factors for EXP'_{v_2} with the presence of each solution.

Consequences of alternatives on risk factors (Figure 4(b)) is not the only factor for selecting a solution among alternatives. As depicted in Figure 4(a), different solutions have side-effects on the cost of the product, portability, and player delays. In the next section, we introduce a decision analysis method for analyzing and comparing different options. We consider consequences of solutions on risk factors, their side-effects on goals, and preferences of stakeholders over security objectives and other goals.

V. Selecting Security Solutions: The Decision Algorithm

This section explains steps 7 to 10 of the process introduced in Figure 1. Once security solutions and their mitigating impacts and side-effects are identified, the analyst structures the impacts of solutions into a consequence table. For example, Table IV shows the consequence table for different alternative security solutions for the critical risks in the DRM player scenario. Note that the first option in the table in Figure 4(b) is neglecting the risks. A DRM player without any security mechanism faces high damage exploitations, while financial and performance costs are avoided.

We adopt the Even Swaps multi-criteria decision analysis approach [7] to make trade-offs among requirements. The Even Swaps method works by trading-off one requirement for another, thus, it does not require eliciting importance weights of requirements for determining the best solution. Requirements can be evaluated in a mixture of scales and by different measurement methods (as illustrated in the consequence table in Table IV).



Risk of EXP'v2 with Presence of Solutions	Probability	Damage
No security solution	Medium	High
Code Obfuscation	Medium	Medium
Tamper Proofing	Medium	Low
Distribution with Physical Token	Low	Low

Figure 4. Alternative security solutions for the Digital Rights Management player

Table IV
CONSEQUENCE TABLE OF ALTERNATIVE SECURITY SOLUTIONS FOR THE DRM PLAYER

	Minimize				Maximize	Minimize	
Alternative	Prob EXP'v2	Damg EXP'v2	Prob EXP v1	Damg EXP v1	G4 Portable	G3 Delay	C1 Cost
No security solution	Medium	High	Medium	High	✓	50 ms	$ 0
Code Obfuscation	Medium	Medium	Low	High	✓	200ms	$20K
Tamper Proofing	Medium	Low	Medium	Medium	✓	500ms	$15K
Distribution with Physical Token	Low	Low	Low	Low	✓	100ms	$40K

✓ Fully satisfied

A. *Basics of the Even Swaps Decision Analysis*

The Even Swaps decision analysis method implicitly extract stakeholders' value trade-offs, i.e., how much stakeholders would give up on one goal for more satisfaction of another. For example, the first option is to neglect the risk of EXP'_{v_2} and EXP_{v_1} and another alternative is `Code Obfuscation`. For the sake of brevity, lets call them A_1 and A_2 respectfully.

A_2 has no impact on the probability of EXP'_{v_2} and on the damage of EXP_{v_1}. It also does not affect the portability of the DRM player. Thus, we can limit comparing the alternatives on the relevant criteria: their risk factors, delay, and costs. Without applying any security solution (A_1), \$0 is paid and the delay is insignificant (50ms), while the damage of EXP'_{v_2} is High and probability of EXP_{v_1} is Medium. A_2 costs \$20K, and adds to the delay but reduces the damage of EXP'_{v_2} to Medium and decreases the probability of EXP_{v_1} to Low.

	EXP'_{v_2} Damage	EXP_{v_1} Probability	Delay	Cost
A_1	High	Medium	50 ms	\$0
A_2	Medium	Low	200 ms	\$20K

Deciding between A_1 and A_2 involves making a trade-off between the risk factors and other criteria (delay and costs). Making such trade-offs depends on stakeholders' preferences. For example, analysts need to understand how much stakeholders are willing to pay for reducing security risks. In an even swap, analysts ask stakeholders a hypothetical question: *If we could reduce the damage caused by exploiting v_2 (EXP'_{v_2}) from High to Medium, how much you would be willing to pay for A_1?*

Let us assume stakeholders declare that they would be willing to pay \$15K, if the EXP'_{v_2} damage was reduced to Medium. A solution with Medium risk and cost of \$15K is not available. This is a virtual, hypothetical alternative which we call A'_1. A'_1 is as preferred as A_1, and can be used as its substitute.

	EXP'_{v_2} Damage	EXP_{v_1} Probability	Delay	Cost
A'_1	**Medium**	Medium	50 ms	**\$15K**
A_2	Medium	Low	200 ms	\$20K

Now deciding between A'_1 and A_2 is easier, because the damage of EXP'_{v_2} is irrelevant and we can focus only on three decision criteria.

In general, in an even swap, the decision analyst, collaborating with the stakeholders, hypothetically changes the consequence of an alternative on one requirement, and compensates this change with a preferentially equal change in the satisfaction level of another requirement. Swaps aim to either make criteria irrelevant, in the sense that both alternatives have equal consequences on the criteria, or create a dominant alternative. Alternative A dominates alternative B, if A is better than (or equal to) B on every criteria [8]. Irrelevant goals and dominated alternatives can both be eliminated, and the process continues until only the most preferred alternative remains [8].

For example, in the DRM scenario, after swapping the damage of EXP'_{v_2} with costs, neither of the alternatives become dominant. Therefore, the swapping process will continue. Analysts ask stakeholders another hypothetical question to extract another value trade-off: *If we could reduce the probability of EXP_{v_1} from Medium to Low, how much more you would be willing to pay for A_1?*

Let us assume stakeholders declare that they would be willing to pay another extra \$10K if the probability of exploiting v_1 could have been reduced to a Low level, in total paying \$25K. After making this swap, the probability of EXP_{v_1} is irrelevant in the final decision and stakeholders only need to swap costs and the delay:

	EXP_{v_1} Probability	Delay	Cost
A'_1	**Low**	50 ms	**\$25K**
A_2	Low	200 ms	\$20K

Analysts ask stakeholders another hypothetical question to extract another value trade-off: *If we could reduce the costs of A_1 from \$25K to \$20K, how much delay would you tolerate (in addition to current 50 ms)?* Let us assume stakeholders can tolerate an additional 200 ms delay to save \$5K (in total 250 ms), thus A_1 and A_2 will become indifferent with respect to costs and based on delay, A_2 would be a better solution. Note stakeholders' goal is to minimize the costs and the player's delay.

	Delay	Cost
A_1'	**250 ms**	**\$20K**
A_2	200 ms	\$20K

This process continues by comparing `Tamper Proofing` and `Code obfuscation` in which `Tamper Proofing` is recognized as the dominant one. Then `Tamper Proofing` and `Distribution with physical token` are compared in the same way. The results of this analysis shows that `Tamper Proofing` is more cost-effective than using `Physical tokens` due to the costs of physical tokens and their portability issues.

B. Discussion: Even Swaps in Practice

Although the Even Swaps method helps identify stakeholders' value trade-offs and compare alternatives systematically, analysts need to decide which goals to swap in each step based on their personal judgement. For example, in the Even Swaps example for the DRM player scenario, the analyst first asked stakeholders to swap reducing the damage of vulnerability with the costs, while stakeholders had other swap options (e.g., swapping probability and delay).

When several requirements and risks are considered, determining the right swaps to make among numerous possibilities is hard for human decision makers [8], and thus non-expert users may not know which goals to swap next, and the process can become tedious without creating a dominated alternative.

To tackle the Even Swaps problems in practice, in a previous work [9], we introduced a tool that semi-automates the Even Swaps process, in the sense that the process is still interactive with stakeholders while an algorithm suggests the next goals to swap to decision analysts in each step of the process. A prototypical tool is developed to support the process and a demo is available in [10].

VI. Related Work

Attack and Vulnerability Modeling and Analysis. Various security-related modeling techniques help analyze attacks and vulnerabilities within the system. For example, Attack trees [11] represent attacks as a hierarchical tree-structure. The notions of obstacle in [12] captures exceptional behavior and anti-goals to model obstacles set up by attackers to threaten security goals. Jürjens proposes a UML profile, called UMLsec [13], to express security-related issues in UML models.

Security Risks Analysis. Some security requirements engineering methods focus on integrating risk analysis into the requirements engineering mainstream. Matulevicius et al. [14] improve the Secure Tropos [15] modeling language for risk management purposes, where risk is defined as the combination of a threat with vulnerabilities leading to negative impacts on assets. Mayer et al. [16] intertwine requirements engineering activities with risk analysis by considering impacts of risks on business assets and eliciting security requirements for mitigating the risks. They define risk metrics to support risk assessment at the modeling language level [17]. Security Attributes Evaluations Method [18] assesses risks and benefits of security technologies and analyzes threat index before and after applying security technologies, in order to compare alternative technologies. CORAS [19] provides a framework for model-based risk assessment in the form of a UML profile. STRIDE [20] provides a threat classification for characterizing known threats based on the exploitation.

Our work fills various gaps in existing risk assessment approaches. In this work, the modeling process helps understand how vulnerabilities are introduced to the system, how they can get exploited, and how critical the risk of exploitations are. In most of existing methods, evaluating the extent of risks are assumed to be done by expert analysts. This work brings together the use of an existing vulnerability scoring system and a requirements modeling notation for assessing the risks of vulnerabilities. The final output of existing risk assessment methods is a list of risks and countermeasures. We extend the analysis by providing a practical decision analysis method to select a solution among alternatives.

Requirements Decision Analysis. From the decision analysis point of view, this work relies on the Even Swaps method. The Even Swaps method was first applied in requirements engineering for qualitative trust trade-offs analysis [21].

Existing requirements decision analysis methods mostly rely on the availability, accuracy, and meaningfulness of quantitative estimations of costs, benefits, and requirements satisfaction levels [22], [23], [24], [25]. For example, Feather et al. [22] propose a quantitative model for strategic decision analysis and making trade-offs among quality requirements. This approach is based on "coarse quantification" of relevant risk factors and their interactions. In some requirements analysis approaches [26], [27], [28], requirements and alternatives are quantified by using ordinal measures or a probabilistic layer for reasoning about partial goal satisfaction.

In practice, however, formal and rigorous methods to effectively estimate risk factors are often not feasible [29]. Hence, faced with the typical absence of reliable quantitative data, systematic security trade-off decision analysis methods using qualitative values are needed. Some decision analysis approaches such as AHP [30] and Even Swaps [7] circumvent the need to measure requirements and consequences of

solutions or extract preferences directly from stakeholders. As opposed to AHP, the Even Swaps method does not require collecting explicit preferences from stakeholders. Our work adopts the Even Swaps method for making security decisions when multiple alternative solutions are available for risks of vulnerabilities exploitation.

VII. Conclusions and Limitations

We adapt the CVSS for evaluating risks of vulnerabilities within *i** models of system requirements and stakeholders' goals. This helps analyze the damage of risks in terms of negative impacts of exploitations on goals and requirements. We adopt and enhance the Even Swaps [7] method for analyzing requirements trade-offs to decide over multiple alternative security solutions.

Even though we simplified the CVSS metrics and calculations, analysts may find evaluating vulnerabilities in terms of CVSS metrics complex. Metrics evaluations may be inaccurate and unreliable, and the metrics aggregation method may reduces the accuracy even more by aggregating vulnerability scores into one value. However, aggregating the scores is inevitable because decision analysis is not practically possible if we analyze all CVSS metrics in the Even Swaps process individually.

Making trade-offs by Even Swaps may need substantial cognitive abilities and domain knowledge. In practice, stakeholders may reject every suggested swap, which results in several swap queries to find the one that stakeholders are able to make. The process may become time-consuming or never ends. Nevertheless, if stakeholders are not able to specify how much they would sacrifice on one goal for satisfaction of another, they will not be able to numerically specify preferences over the goals either.

References

[1] C. Collberg and J. Nagra, *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Addison-Wesley Professional, 2009.

[2] M. J. Kenning, "Security management standard — iso 17799/bs 7799," *BT Technology Journal*, vol. 19, no. 3, pp. 132–136, 2001.

[3] E. R. Kissel, *Glossary of key information security terms*. NIST IR 7298, 2005.

[4] P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System Version 2.0," tech. rep., NIST, 2007.

[5] G. Elahi, E. Yu, and N. Zannone, "A vulnerability-centric requirements engineering framework," *Requirements Engineering Journal, Special Issue*, 2009.

[6] E. Yu, *Modeling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, 1995.

[7] J. S. Hammond, R. L. Keeney, and H. Raiffa, *Smart choices : a practical guide to making better life decisions*. Broadway Books, 2002.

[8] J. Mustajoki and R. P. Hämäläinen, "Smart-swaps - a decision support system for multicriteria decision analysis with the even swaps method," *Decis. Support Syst.*, vol. 44, no. 1, pp. 313–325, 2007.

[9] G. Elahi and E. Yu, "A semi-automated decision support tool for requirements trade-off analysis," *To apear in proceeding of COMPSAC'11*, 2011.

[10] "Decision analysis tool demo, release 1, dcs, univ. of toronto, available at http://www.cs.toronto.edu/~gelahi/Release1/Release1.html," 2011.

[11] B. Schneier, "Attack trees," *Dr. Dobb's Journal*, vol. 24, no. 12, pp. 21–29, 1999.

[12] A. van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proc. of ICSE'04*, pp. 148–157, IEEE Computer Society, 2004.

[13] J. Jürjens, *Secure Systems Development with UML*. Springer, 2004.

[14] R. Matulevicius, N. Mayer, H. Mouratidis, E. Dubois, P. Heymans, and N. Genon, "Adapting secure tropos for security risk management in the early phases of information systems development," in *CAiSE*, pp. 541–555, 2008.

[15] H. Mouratidis and P. Giorgini, "Secure Tropos: a Security-Oriented Extension of the Tropos Methodology," vol. 17, pp. 285–309, 2007.

[16] N. Mayer, A. Rifaut, and E. Dubois, "Towards a risk-based security requirements engineering framework," in *In Proc. of REFSQ05*, 2005.

[17] N. Mayer, E. Dubois, R. Matulevicius, and P. Heymans, "Towards a Measurement Framework for Security Risk Management," in *Proc. of Modeling Security Workshop*, 2008.

[18] S. A. Butler, "Security attribute evaluation method: a cost-benefit approach," in *Proc. of ICSE'02*, pp. 232–240, ACM, 2002.

[19] F. den Braber, T. Dimitrakos, B. A. Gran, M. S. Lund, K. Stolen, and J. O. Aagedal, "The CORAS methodology: model-based risk assessment using UML and UP," *UML and the unified process*, pp. 332–357, 2003.

[20] The STRIDE Threat Model, MSDN library. http://msdn.microsoft.com/library/ms954176.aspx.

[21] G. Elahi and E. Yu, "Trust trade-off analysis for security requirements engineering," *In Proc. of RE'09*, pp. 243–248, 2009.

[22] M. S. Feather, S. L. Cornford, K. A. Hicks, J. D. Kiper, and T. Menzies, "A broad, quantitative model for making early requirements decisions," *IEEE Software*, vol. 25, pp. 49–56, 2008.

[23] H. P. In, D. Olson, and T. Rodgers, "Multi-criteria preference analysis for systematic requirements negotiation," in *COMPSAC '02*, pp. 887–892, 2002.

[24] W. Ma, L. Liu, H. Xie, H. Zhang, and J. Yin, "Preference model driven services selection," in *Proc. of CAiSE'09*, pp. 216–230, 2009.

[25] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Softw.*, vol. 14, no. 5, pp. 67–74, 1997.

[26] P. Giorgini, G. Manson, and H. Mouratidis, "On security requirements analysis for multi-agent systems.," in *SELMAS'03*, 2003.

[27] Y. Asnar and P. Giorgini, "Modelling risk and identifying countermeasure in organizations," pp. 55–66, 2006.

[28] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *SIGSOFT '04/FSE-12*, pp. 53–62, 2004.

[29] Microsoft Corporation: Microsoft Solutions for Security and Compliance and Microsoft Security Center of Excellence, *Security Risk Management Guide*, 2006.

[30] T. Saaty, *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. New york: McGraw-Hill, 1980.