# Threat Modeling Approaches for Requirements Specifications and Software Design

Kenneth Dinsmore, Sangam Sahai

Texas Tech University

## Abstract

In today’s times, there are many new techniques for Threat Modelling. This research paper discusses new and novel approaches for Threat Modelling.

This paper first discusses the broad classification of Threat Modelling approaches, and then discusses some new approaches. Finally, this paper proposes a brand new technique for developers who are newbies in the field of Software Security. This technique abstracts the complexities of the classification of Threat Modelling approaches and instead suggests the developers the best approach suitable for them based on some input parameters.

## Intro

There was a time when security requirements for software, with the exception of safety and security critical systems, were typically not given a lot of thoughtful analysis. They were treated as “non-functional” requirements and incorporated into the design and implementation process with varying degrees of thoroughness. Those processes are quickly becoming a thing of the past. Today, there are a plethora of methods for eliciting and documenting security related requirements. In this work, we summarize a small subset of the methods and approach that have been proposed and are in use today. We then proposed an idea that may be useful in advancing the field.

## Related Work

Kibir and Rahman present a brief survey of security Requirements Engineering (RE) [1], Several asset based approaches have been reviewed I this work including Bhattacharjee, Barik and Mazumdar [2], Gurses, Janke, Obry, Onabajo, Santen and Price [3]. Attacker based approaches that we have studied include work by Faily and Flechais [4], Schneider [5], Whittle, Wijesekera and Hartong [6], Yuan, Nuakoh, Beal and Yu [7]. Software tool based methodologies include work by Belani, Car and Caric, [8], Hafiz and Johnson [9], Schaad and Borozdin [10], Jurjens and Houmb [11], Wang, Wong and Xu [12], Kothari, and Smith et al. [13], and Lincke [14]. Finally, we look at two approaches that combine multiple methods Schaad and Garaga [15], and Vanche [16]. Aside from these works included here, there have been thousands of contributions by a multitude of authors and researchers.

### Attacker Based

#### Barry is not the weakest Link Eliciting Secure System Requirements with Personas

The authors present a user-centric approach to requirements engineering using personas, basing this approach on the claim that personas help with understanding both specific users of a system and potential attackers. Personas “represent archetypal users, and embody their needs and goals” [4]. The initial state of the art that is described is very similar to the mis-use case approach to security requirements engineering where the persona is similar to the actor or attacker role. The new contribution of this paper is the idea of combining usability engineering, requirements engineering and security into a single process. This process involved an initial scoping of the system followed by interviews of users to gain context, persona and functional requirements data. The last stage is to conduct workshops to elicit the actual requirements based on the personas, context and goals. It should be noted that in this method, while personas are grounded in real people and real events, the personas themselves become named fictional people who would use the system and the act of referring to the persona by name helped to guide discussions and reach consensus. This process included the use of negative personas,(attackers) to identify security goals, propose countermeasures and assess the impact on the (legitimate) personas.

#### Comparison of Information Security Risk Analysis in the Context of e-Government to Criminal Threat Assessment Techniques

This paper presents a narrowly targeted view on the comparison between traditional risk management approaches and criminological threat assessment. The focus is the specific set of internet facing applications that are categorized as e-government systems and there is considerable focus on the issues that are specific to government as opposed to commercial enterprises. While generalized risk management practices apply, governments face additional categories of risk such as societal, technical, economic, political, and security where these risk categories apply to the effect on the society as a stakeholder group rather than on individual stakeholders. The authors then introduce the idea of using the same kinds of threat assessment techniques that are used in proactive policing. This approach requires the identification of potential perpetrators (attackers), gathering intelligence on the subject and assigning risks based on that intelligence, then managing the case (subject). The author also includes an asset based analysis approach that relies on identifying the consequences of attacks and the likelihood of such an attack.

#### Executable Misuse Cases for Modeling Security Concerns

This paper presents a modeling language for misuse cases that allows the misuse cases to be executed. The methodology described requires the support of an extended UML modeling tool to author the misuse cases using activity and sequence diagrams. The extension includes the relationships <<threatens>> and <<mitigates>> for the use case nodes. The result is a set of executable finite state models based on the misuse cases and the applied attack mitigation strategies. The technique is explained through two case studies, a train control system and an electoral voting system, both of which have stringent security requirements. The main idea of the work is that by transforming the UML along with the mitigation strategy into finite state machines and executing them, which animates the original sequence diagrams; the stakeholders will better understand the potential threats, the systems reactions to the attacks and the new behavior of the system. This will in turn lead to new scenarios to be modeled, executed and analyzed. This attack scenario centered approach qualifies it as an attacker centric approach. The authors note that of the thirteen attacks that they identified against the target systems, nine were able to be modeled using the proposed system and the attacks that could not be modeled were outside of the scope of the initial requirements specification.

#### Retrieving Relevant CAPEC Attack Patterns for Secure Software Development

In this work, the main idea is to codify systematic processes for each phase of the software development lifecycle into an automated tool. This, the authors believe will assist developers in using known attack patterns to identify weaknesses and build the appropriate countermeasures into the software under development. The work describes the current phase of their ongoing efforts in this regard; specifically, the authors work in identifying and retrieving Common Attack Pattern Enumeration and Classification (CAPEC) data that is appropriate and relevant to the target software system. Since the tool is designed to be used with Microsoft’s SDL tool, the paper discusses this tool and the mappings between the CAPEC initiative and the STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege) models in the context of performing an architectural risk analysis. The proposed tool also sorts the identified threats by relevance to the situation at hand, where relevance is a function of completeness, likelihood, severity of impact, attacker skill level, Confidentiality, Integrity and Availability (CIA) impact, and attacker motivation. This work views the system both from an asset perspective and an attacker perspective, with the focus being on attack patterns (attacker behavior) which makes it a good fit for our survey of attacker centric modeling approaches.

#### TAM2 Automated Threat Analysis

Security Analysis is something which should be done as early as possible in the SDLC.

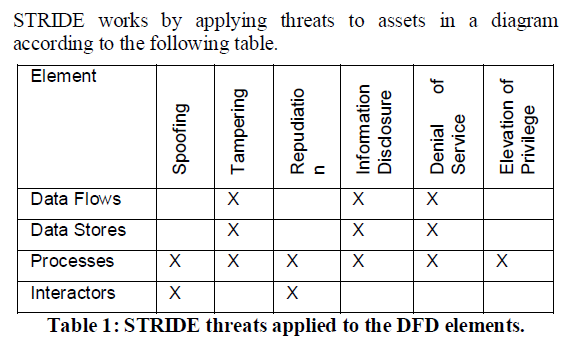
One of the most initial diagrams in the SDLC is the Software Architecture Block Diagram. But there are not many approaches which use this diagram for Threat Modelling.

This approach uses the Software Architecture Block Diagram for Threat Modelling.

This new approach involves 2 solutions -

1. Applying security analysis to the most initial Software Architecture Block Diagram
2. Using Annotations to give precise information about the system.

STRIDE is the Microsoft’s signature method of Threat Modelling, but it is applicable to Data flow Diagrams.

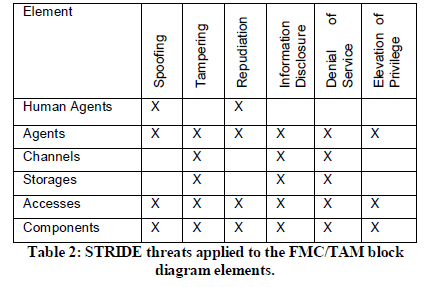


Whereas the most common way of representing the Software Architecture is the UML/ Block Diagram.

So we should think of some way in which we can use this diagram for Threat Modelling.

We apply STRIDE methodology to Block Diagram **comparing the shapes of the Data Flow Diagram to Block Diagram**.

And then we come up with STRIDE template based on DFD.



The second part of the solution involves Annotations.

In Software Diagrams, to add semantics, we add shapes. But shapes always do not help us. In our block diagram, The File System and The Database, both have the storage shape but both these assets have different sets of Vulnerabilities. For example, Database is prone to SQL injection attack but a file system is not! So shapes also add ambiguity.

To fight this ambiguity, we add annotations.

We use the Component Make and the Version number of the asset as the Annotation.

And then we can use these annotations to search for vulnerabilities in all the public vulnerability databases like CVE (Common Vulnerability Enumeration).

And we can also look for the appropriate controls for those vulnerabilities.

And SAP has also implemented a tool to validate this approach.

#### A Threat Model Driven Approach for Security Testing

This approach is also called Scenario specification based Threat Modelling.

This approach is used for detecting undesirable threat behavior at run time.

Sequence Diagram is used to model threats.

UML sequence diagrams describe and interaction by focusing on the sequence of the messages that are exchanged along with their corresponding occurrence with respect to time.

Sequence diagrams are exploited to describe the interactions an attacker would go through, to compromise the system.

Message sequence is the path from the first message to the last message.

Message sequence of threat can be used for security testing.

This is called scenario specification based method.

A threat model represented by a sequence diagram (SDTM) is viewed as a tuple of

(O, M, E) where

– O = {o1, o2, · · · , om} is a finite set of objects.

For any oi ∈ O, let fOC(oi) represent the belonging class of object oi;

– E = {e1, e2, · · · , ek} is a finite set of events. Let

feo be a function from E to O, feo(e) = oi ∈ O means e is the occurrence of the corresponding message in the lifeline of oi;

– M = {m1,m2, · · ·,mn} is a finite set of labelled

Messages. ∀mk ∈ M, let !mk and ?mk represent the send and the receiver of mk, respectively, and ∃ei, ej ∈ E the **corresponding sending and receiving event, and let fem(ei) =!mk and fem(ej) =?mk;**

Each message sequence represents

a single scenario of the threat model.

For a threat model SD\_TM = (O,M,E), msq is a message sequence in the form (m1 → m2 → ... → mj →

mk → ... → mn), ∀i, 1 < i < n,mi ∈ M, mi → mj

The sequence of events should be derived from the sequence diagram which represents the trace of the threat behavior.

Thus the threat trace of event sequences can be generated from the threat model.

At runtime each message is realized by a message sending event and a message receiving event.

Hence, a system behavior that is represented by a message sequence can also be represented at a finer granularity in terms of a sequence of sending and receiving events.

It is a run of a sequence diagram in a certain scenario, we name it as threat trace, denoted by ttrace. Let SDTM =

(O,E,M) be a threat model, ∀ei ∈ E, 1 < i < n, and an event sequence e1 → e2 → · · · → en is an ttrace, which represents one behavior scenario of SDTM,

Where ei = (eid, etype, sender, receiver, method), ∃m ∈

M,feo(ei) =!m or ?m.

1. eid is a serial number, and uniquely denotes an event.
2. etype denotes the message sending event / receiving event represents the message sending event, and \_r\_represents the message receiving and executing event;
3. Sender, receiver represent the message sending object and the message receiving object of the event,
   1. if fem(ei) =!m, then sender = feo(ei), receiver = feo(?m)
   2. if fem(ei) =?m, then sender = feo(!m), receiver = feo(ei)
4. Method is the method called by the message m.

Threat model driven security testing –

1. Derive message sequences from the sequence diagram which represent a threat.
2. Track the corresponding **ttrace**.
3. Match the runtime execution trace with **ttrace.**

Instrumenting the code –

Objective here is to collect the run time information.

So we put probe statements before a method call and during a method execution.

For example –

A ()

{

//probe before execution of method A

//probe statement before calling B

B ();

}

The execution trace is called **etrace.**

And then penetration testing is done to check if for any scenario, **ttrace = etrace** ?

If yes, then a threat scenario is encountered and the developers are reported to fix the issue.

#### Designing Software Security With UML Extensions

Concrete security deployment is very important.

We will discuss a very short example here. Input validation is very important to fight SQL injection attacks. But if the validation is only at the client side, then that is of no use as the user can easily bypass that by disabling javaScript. Hence security here is inadequate.

So besides the presence of the code, LOCATION of the code is equally important.

An exhaustive deployment diagram solves this problem.

In a single picture, it shows how the software is deployed and where is it deployed.

A preliminary MDD (Misuse deployment diagram) shows where attacks are handled in single diagram and where major attacks should be addressed.

For example –

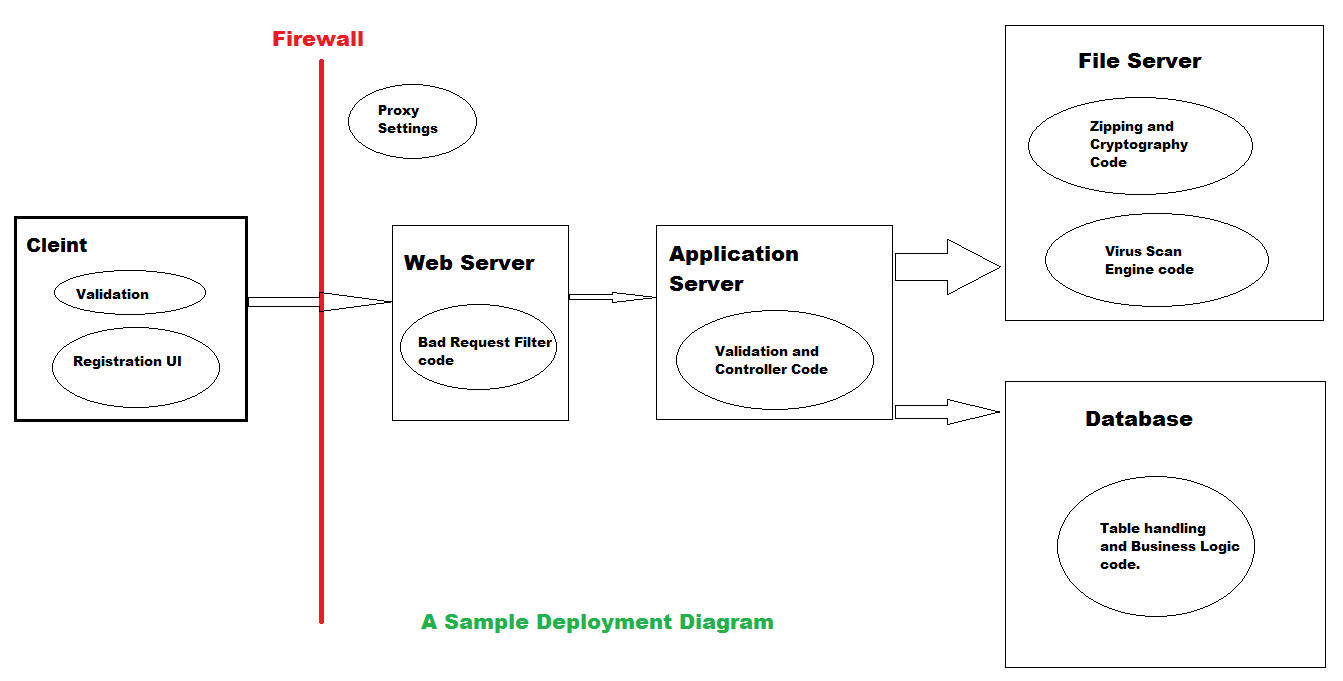
Captcha related code should be on server side.

Validation code should be on both client and server side.

Proxy related code should be in the firewall.

Business Logics should be mostly in Databases so that error logging is easy and the logic is not written in the server side controller.

For file intensive applications, file scan engine code should be places in the file server.



### Combined

#### Vulnerability Analysis for a Qualitative Security Evaluation

This work describes a statistical analysis of software systems based on a database of vulnerabilities, the vulnerability lifecycle, known behaviors of attackers and the ways in which these factors influence the security posture of the system under study. The authors describe the vulnerability lifecycle with specific attention given to the relationship and temporal distance between vulnerability discovery, exploit and disclosure, and the way in which these events affect the systems security posture. Given that this work is focused on the lifecycle events related to known vulnerabilities in existing software, it falls somewhat outside of our stated domain; however, it should be noted that the results of this work are important for understanding the relative risk that each class of vulnerability poses to the system under study. Furthermore, this risk information can be used by designers and architects who are building and maintaining other software components. In other words, by researching the existing vulnerabilities and understanding how they affect the security posture of the affected systems, we can better understand and prioritize the risks to systems that are stil under development and maintenance.

#### Automating Architectural Security Analysis

## Current State-of-the-art in this Area

Currently, there are dozens of approaches to eliciting and documenting security related requirements. Some of these methods are specific to the security aspect of software, some encompass all or most of the aspects that were once classified as “non-functional” requirements and others are all-encompassing methodologies. The list in Table 1 provides a sample of the methodologies that are available. This work touched on some of these briefly.

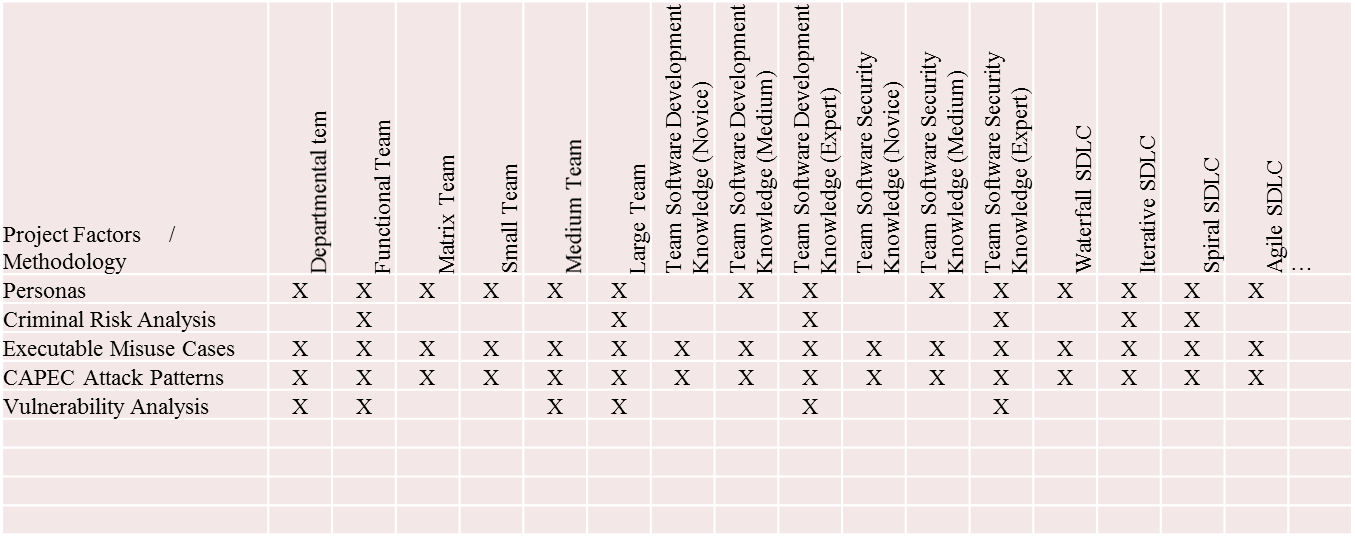
|  |
| --- |
| Abuse Cases |
| Abuser Stories |
| Accelerated Requirements Method (ARM) |
| Agile Security Requirements Engineering |
| Anti-models |
| Attack Trees |
| Common Criteria |
| Comprehensive Lightweight Application Security Process (CLASP) |
| Controlled Requirements Expression (CORE) |
| Critical discourse analysis (CDA) |
| Fault Trees |
| i\* Security Requirements |
| Issue-based information systems (IBIS) |
| Joint Application Development (JAD) |
| Knowledge Agent-oriented System (KAOS) |
| Misuse Cases |
| Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) |
| Problem Frames |
| Quality Function Deployment (QFD) |
| Risk Analysis |
| Secure TROPOS |
| Security Design Analysis (SeDaN) |
| Security Development Lifecycle Tool (SDL) |
| Security Models |
| Security Patterns |
| Security Problem Frames |
| Security Use Cases |
| Simple Reuse of Software Requirements (SIREN) |
| Software Cost Reduction |
| System Quality Requirements Engineering (SQUARE) |
| Threat Modeling for Security Requirements |
| Threat Trees |
| Usage-centric Security Requirements Engineering (USeR) |

Table 1: List of current Security Requirements Elicitation and Documentation Methodologies [1]

## Possible Solutions

One thing that many of the researchers noted [17] is the lack of education possessed by developers and the lack of focus on security related requirements engineering goals and practices. It has also been noted that software developers should be able to incorporate security related requirements engineering into their processes without learning or memorizing security knowledge, changing their work habits and they should have tool support [17]. With so many approaches and methodologies for capturing security related requirements, and considering the restrictions on impact stated above, it seems that what is needed is a methodology for deciding with approach, methodology, or tool would be appropriate for a given software development or maintenance project. Our initial idea is that the factors that would need to be taken into consideration include environmental factors such as the development team structure, size and level of education with respect to software development in general and security related requirements specifically; project factors such as the software development lifecycle that the project is following; application factors such as the type of software that is being developed, for instance is it an internal application, a security and safety critical system, a public facing web application, and the type of data that it collects, stores and transmits; and customer related factors including any security related legislation, regulation or standards that must be applied to the development process. The idea is to create a decision tree (possibly automated) that would take in each of the parameters listed above and indicate which security requirements engineering approaches, processes methodologies and tools would be best suited to the project.

Based on our knowledge and research, we have come up with the following data template which suggests the methodologies based on the input parameters.



## Conclusions

This research paper briefed the classification of threat modelling approaches and then discussed some new and novel approaches of threat modelling. Then we discussed that there are so many approaches available that it can be at time confusing for the newbies on the field of software security to decide which technique to use. Our new approach targets to clear that air of confusion and give the developers a tool which can guide them and then suggest them the appropriate approach based on some input parameters. The research work which we conducted to come up with the data for the tool is not exhaustive. It is still open to a whole lot of work and new research. And the more we research on the tool and expand our database, the more accurate our tool will become.

# References

|  |  |
| --- | --- |
| [1] | A. Kabir and M. Rahman, "A Survey on Security Requirements Elicitation and Presentation in Requirements Engineering Phase," *American Journal of Engineering Research,* vol. 02, no. 12, pp. 360-366, 2013. |
| [2] | J. Bhattacharjee, A. Sengupta, M. S. Barik and C. Mazumdar, "A Two-Phase Quantitative Methodology for Enterprise Information Security Risk Analysis," ACM, Maharashtra, India, 2012. |
| [3] | S. Gurses, J. H. Janke, C. Obry, A. Onabajo, T. Santen and M. Price, "Eliciting Confidentiality Requirements in Practice," 2005. |
| [4] | S. Faily and I. Flechais, "Barry is not the weakest link: Eliciting Secure System Requirements with Personas," Oxford, UK, 2010. |
| [5] | R. M. Schneider, "A Comparison of Information Security Risk Analysis in the Context of e-Government to Criminological Threat Assessment Techniques," ACM, Kennesaw, GA, 2010. |
| [6] | J. Whittle, D. Wijesekera and M. Hartong, "Executable Misuse Cases for Modeling Security Concerns," ICSE, Leipzig, Germany, 2008. |
| [7] | X. Yuan, E. B. Nuakoh, J. S. Beal and H. Yu, "Retrieving Relevant CAPEC Attack Patterns for Secure Software Development," ACM, Oak Ridge, TN, 2008. |
| [8] | H. Belani, Z. Car and A. Caric, "RUP-Based Process Model for Security Requirements Engineering in Value-Added Service Development," SESS, vancouver, Canada, 2009. |
| [9] | M. Hafiz and R. E. Johnson, "Security-oriented Program Transformations," CSIIRW, Oak Ridge, Tennessee, 2009. |
| [10] | A. Schaad and M. Borozdin, "TAM2: Automated Threat Analysis," SAC, Riva del Garda, Italy, 2012. |
| [11] | J. Jurjens and S. H. Houmb, "Dynamic Secure Aspect Modeling with UML: From Models to Code," Springer-Verlag, Berlin Heidelberg, 2005. |
| [12] | L. Wang, E. Wong and D. Xu, "A Threat Model Driven Approach for Security Testing," Third International Workshop on Software Engineering for Secure Systems, 2007. |
| [13] | V. Kothari, S. Smith, J. Blythe and R. Koppel, "Agent-Based Modeling of User Circumvention of Security," ACM, Paris, France, 2014. |
| [14] | S. J. Lincke, "Designing Software Security with UML Extensions," in *JCSC*, 2012. |
| [15] | A. Schaad and A. Garaga, "Automating Architectural Security Analysis," ACM, Newark, New Jersey, 2012. |
| [16] | G. Vanche, "Vulnerability Analysis for a Quantitative Security Evaluation," IEEE, 2009. |
| [17] | H. Meland and J. Jensen, "Secure Software Design in Practice," IEEE, 2008. |