Christopher Foley
CIS6375 – Assignment 01

Question 1: Compare misuse cases to our way to enumerate threats (Chapter 1 in slides, Chapter 2 in notes).

With modern software development techniques, a Use Case indicates how a user will use the system. A misuse case [Alex] is a use case developed to show how a user hostile to the system will interact with the system. As such a misuse case has all the attributes of a use case. Misuse cases may be extended to create the Nonfunctional specifications and high level end to end testing and timing specifications. As the table below shows, misuse cases are high level and may involve multiple systems.

Threats are individual attacks on data and system resources. A method for cataloging threats is noted in *The design and evaluation of secure systems: an approach using patterns* [Fer18]. Both approaches show threats, however each has different purposes as shown in the table below:

Misuse Case	***Threat Enumeration***
Created from use case [Fire]	Created from a specific object model.
Shows how an actor can misuse the system	Shows specific threats to system
Used to create Non Functional requirements	Can be used to create a check list to derive threat/defense patterns
High Level	Lower level
Motivates discussion of other misuse cases [Nus]	May show threats not considered
User Centered	Object and implementation centered.
Paired with functional requirements	Derived from object model.
Written in non-technical user friendly language	UML
Can be used to generate “blackbox” testing	The use of UML patterns allows development of code based on a tested pattern.
Can be used to track development progress	
Requires actors	The parallelogram equals threat, triangle equals countermeasure allows **visual validation** of design model (Trapezoid equals known threat defense).
Allows threat/gap from non actor subsystems	May show gaps in model/pattern interactions

An example of a misuse case may be found in *Misuse Cases: Use Cases with Hostile Intent* as Figure 1 [Alex]. The use case describes a medical practice where an practitioner creates and updates an appointment. The hostile user changes the appointment or alters the updated data. In the use case diagram the actors are on the left (light circle with dark text) interacting with the use cases and the hostile actors are on the right (drawn as dark circle with light text) interacting with the same use case, with malicious intent (change appointment, alter medical record).

Question 2: In the paper on tactics “Architectural tactics for security and their realizations” [Fer18a], the authors argue that their tactic set is “better” than the original set. Prove or disprove the assertion.

In their paper the authors never assert that their tactic set is “better”, rather they correctly and justifiably assert that their combination of tactics and architecture is both a refinement and more useful than the original set. The paper in question addresses some overlaps in terminology, definitions, tactics and security patterns. The key contributions of the authors are to clarify the terminology used and significantly, take the Classification of Security tactics, according to Bass, et al. [Bass] and refine the tactics.

Numerous authors have explored the relationships between patterns, architecture and tactics. Patterns and architectures provide a framework for development to occur. However, tactics are different in that they are more specific (e.g. authenticate user, validate access, log changes, etc.). The authors revised the Bass classifications and produced a hierarchy similar to the following:

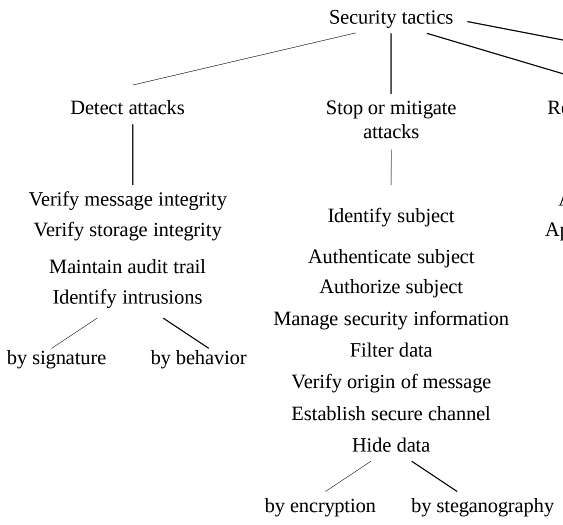

Figure 1: Hierarchy of Tactics (Fernandez, et. al.)

However, simultaneously, a different classification, as noted in the related works section, was presented by Ryoo et al.. [Ryoo12].   Their classification is shown below in Figure 2: Revised Hierarchy (Ryoo, Laplante, Kazman). Both classifications have their advantages, however I would suggest that they be combined into a hierarchy that reduces the number of main categories and simplifies the categories in a manner similar to that shown in Figure 3: Revised Hierarchy Of Tactics (Foley). The revised classification assumes logging of all interactions and that institution policies are designed to stop/mitigate attacks and define a recovery policy before the intrusion is detected. Message and system integrity are checked to ensure attacks are stopped/limited. Reaction to attack is assumed as part of stopping/mitigating and recovery since it is assumed that a commercial system under attack will as part of its design wish to stop or mitigate the attack.

In Section 11 of their paper the authors present a rationale for (non)validation of their tactics. The difficulty of measuring a hierarchy of tactics is that there is no known measure for a hierarchy and tactics are simply a measure of what controls make a system secure. [Fer18b] The work of Ryoo, et al., attempted to validate the use of tactics in Open Source software and found inconsistent use of tactics other than encryption. [Ryoo16] A concern with measuring tactics and their impact is subjecting a system to an attack, measuring the impact of the attack, then measuring the impact after tactics are applied. One option might be to establish a ratio of tactics to threats as noted in the work of one of the authors. [Fer18b] This work creates a ratio between known threats and threats countered by patterns used in the system. A similar ratio might be used to evaluate a system. Additionally of note is the work of Abdelkareem M. Alashqar, Hazem M. El-Bakry and Ahmad Abo Elfetouh [Ala]where they adopted the Hierarchy of Tactics and assigned specific techniques to the categories and then tested their impact on a specific system. Their intent appears to extend the work in the paper by providing empirical data with respect to the tactics. Thus a developer and architect can establish categories with specific quality parameters as a goal. Their tool will allow the developer/architect to judge the relative impact of each tactic. Although their classification is not entirely complete, it provides a measure that may be used to judge the impact of known security techniques.

Selection of one tactic may preclude additional tactics. No security tactic is perfect, each tactic is a countermeasure against one type/class of attack. A developer will select and use tactics in conjunction with the patterns and architectures chosen for development. However as recently demonstrated by Mel Brooks in “Spaceballs” (search YouTube for the term “Spaceballs12345”) and recently by the pop icon Kanye West [Naked], no system is secure without the cooperation and active use of its users.

By behavior

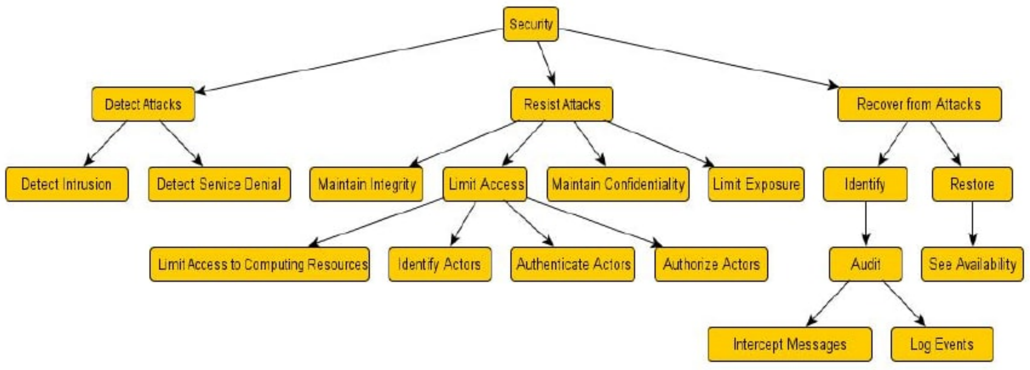

Figure 2: Revised Hierarchy (Ryoo, Laplante, Kazman)

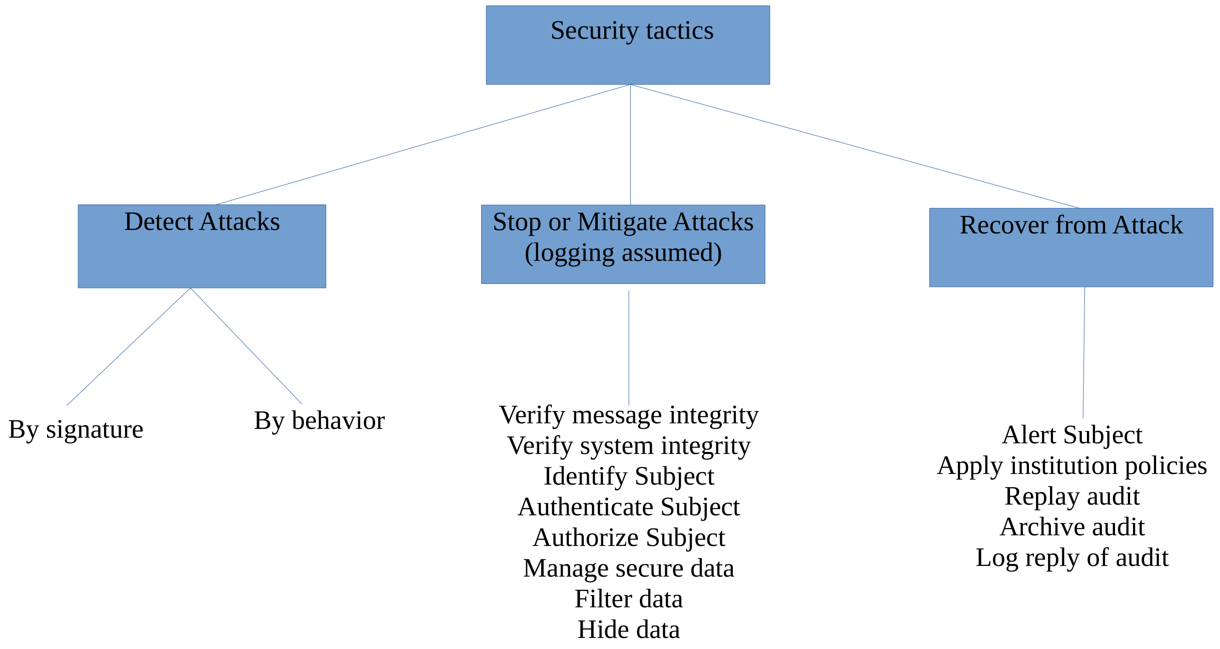

Figure 3: Revised Hierarchy Of Tactics (Foley)

Bibliography

Ala: Abdelkareem M. Alashqar, Hazem M. El-Bakry and Ahmad Abo Elfetouh, A Framework For Selecting Archetectural Tactics Using Fuzzy Measures, 2017
Alex: Alexander, Ian F., Misuse Cases: Use Cases with Hostile Intent, 2003
Fer18: Eduardo B. Fernandez, Security Patterns in Practice - Designing Secure Architectures Using Patterns, 2013
Fer18a: Eduardo B. Fernandez, Hernan Astudillo, Gilberto Pedraza-Garcia, Architectural tactics for security and their realizations, 2018
Fer18b: Eduardo B. Fernandez, Nobukazu, Hironori Washizaki, Evaluating the degree of security of a system built using security patterns, 2018
Fire: Firesmith, D., Use Cases: Pros and Cons, downloaded 30-Sept-2018, https://www.cs.hmc.edu/~mike/courses/mike121/readings/reqsModeling/firesmith.htm
Ryoo16: J. Ryoo, B. Malone, P. A. Laplante, P. Anad, The use of security tactics in open source software projects, 2016
Ryoo12: J. Ryoo, P. Laplante and R. Kazman, Revising a security tactics hierarchy through decomposition, reclassification and derivation, 2012
Naked: John E. Dunn, What Kanye West Can Teach Us About Passcodes, 2018, https://nakedsecurity.sophos.com/2018/10/12/what-kanye-west-can-teach-us-about-passcodes/?utm_source=Naked+Security+-+Sophos+List&utm_campaign=5055a1a16d-Naked+Security+daily+news+email&utm_medium=email&utm_term=0_31623bb782-5055a1a16d-455307865
Bass: L. Bass, P. Clements and R. Kazman, Software Architecture in Practics, 2012
Nus: Nuseibeh, B., Weaving the Software Development Process Between Requirements and Architecture, post 2001, https://pdfs.semanticscholar.org/c80f/02e8dfc4803efb8f4c309a8066daefb9bbb4.pdf \