Threat Modeling

Architecting & Designing with Security in Mind

Venkatesh Jagannathan

OWASP-Chennai Chapter Leader

venki@owasp.org
heyvenki@gmail.com

OWASP

The OWASP Foundation
http://www.owasp.org

Agenda

- Introduction to Threat Modeling
- Precursors to Threat Modeling
- Threat Modeling – How-To
- Test Focused Threat Modeling
- Alternate Threat Models
- Estimating Threat Modeling for Applications

Introduction to Threat Modeling

- Threat Modeling: A systematic & structured security technique, used to identify the security objectives, threats & vulnerabilities of an application, to help make design and engineering decisions, and determine where to prioritize efforts in designing, developing and deploying secure applications
- It’s a day-to-day phenomenon for all of us
  - Assets (e.g. Photos, Jewelry)
  - Architecture/Design of you home
  - Attackers (Burglary)
  - Natural Calamities
- Focus on Architecture/Design driven threat modeling

WHY Threat Model

- Changing Landscape of Security
  - Data from any Application(s) – Hackers target
  - Governmental Regulations
- Brand Protection

Challenges with Threat Modeling

- A mature SDLC
- Time consuming process
- Difficult to show demonstratable ROI
- Fairly dry stuff to do

Precursors to Threat Modeling

- A mature SDLC
- Understanding proper Data classification
- Understand Web App Security Mechanisms

Precursors to Threat Modeling

- Mature SDLC
  - A proper mechanism of gathering requirements (Need a Requirements Phase)
  - A properly defined Design Phase
- Data Classification
  - Assigning level of sensitivity to data when created, modified, deleted or serialized
  - Classification will tell the extent to which the data needs to be controlled or secured
  - It is also an indicative measure of value in terms of Business Assets
  - Determines the Confidentiality – Integrity & Availability
    - Confidentiality – Public, Confidential or Restricted
    - Integrity – Low, Medium or High
    - Availability – Support, Essential or Critical

Web App Security Mechanisms

Mechanism	What is it?
C – Confidentiality	How does the web app protect sensitive data (in all phases)? What cryptographic control measures are built in?
I – Integrity	Does the web app validate the incoming data and ensure that it is not altered and is safe?
A – Availability	Is the web app protected from DoS attacks?
A – Authentication	Is the entity is really who ought to be?
A – Authorization	Does the app provide access to only necessary entities?
A – Auditing	Is all access/modifications etc tracked for security related evidences?
M – Management	Session – Is interaction between the user & the web app (session) secure? Exception – Does the app have a graceful exception handling mechanism? Configuration – Operationally, is the app secure (e.g.: is the app running with least privileges, configuration strings adequately protected etc

- We have dealt with the necessary conditions for Threat Modeling

- New Lets Threat Model

Some Definitions

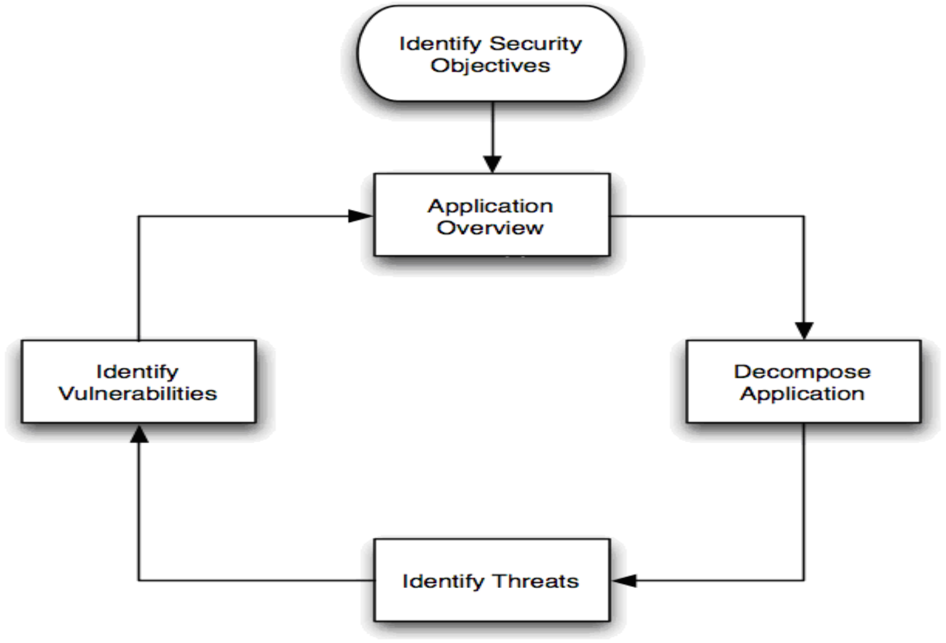
- **Assets** – a resource of value. May be tangible or intangible. Usually reffered to a ‘Object’.
- **Threat** – Undesired act that potentially occurs causing compromise or damage of an asset.
- **Threat Agent** – Something/someone that makes the threat materialize. Usually referred to as ‘Subject’
- **Vulnerability** – Weakness that makes an attack possible
- **Attack** – Act of malicious threat agent. Also known as Exploit.
- **Safeguard (Countermeasure)** – address vulnerabilities (not threats directly); For example – Application Design, Writing Secure Code, deploy with least privilege
- **Probability** – the potential chance of a threat being realized by an attack on an asset
- **Impact** – Outcome of the materialized threat

Threat Agents

- **Accidental Discovery** – An ordinary user stumbles across a functional mistake in an application and gains access to privileged information/functionality
- **Curious Attacker** – Ordinary user who notices something wrong with the application and decides to explore further
- **Insider** – An employee/contractor within the organization

Threat Modeling Process

- Threat Mdeling is Iterative (continuous)
- Threat Modeling takes inputs and generates Outputs for each step in the process



OWASP

Threat Modeling Process

- Step 0: Identify Security Objectives
  - Use the CI4AM security mechanism
    - Confidentiality
    - Integrity
    - Availability
    - Authentication
    - Authorization
    - Auditing
    - Management
  - Examples
    - Prevent Data Theft
    - Protect IP
    - Attain Compliance
    - Provide high availability
- Use the Policies & standards, Legal/Compliance directives & business requirements to list the security objectives

Threat Modeling Process

- Step 1: Profile the Application
  - Where will the application be deployed
    - DMZ/Internal – complete end to end scenarios
  - Who will be the Users (Actors)
    - Customers, sales agents, public users, administrators, DBAs
  - What are the Data Elements?
    - User account data, credit card info, patient disease information…
  - What rights will the actors have?
    - C, R, U D
  - What Technologies will be used?
    - OS, Web/App Servers, Databases, Architectures (SOA/EJB), Programming Language
  - What security mechanisms apply?
    - CI4AM
- Use the use cases, functional specifications & architectural diagrams for profiling the application

OWASP

Threat Modeling Process

- Step 2: Decompose Your Application (Generate Application Context & Scenarios)
  - Trust Boundaries (indicates where trust level changes)
    - Firewall (Internet -> Intranet)
    - Webserver -> Database
    - Your App -> External 3rd party Services
  - Entry Points (Principal attack Targets)
    - Ports, Pages, Components, APIs, Stored Procedures etc
  - Exit Points
    - Pages that display data, functions sending out values
  - Data Flows
    - Should we validate data at each node?
- Use the use cases, functional specifications, DFDs & architectural diagrams for decomposing the application

Threat Modeling Process

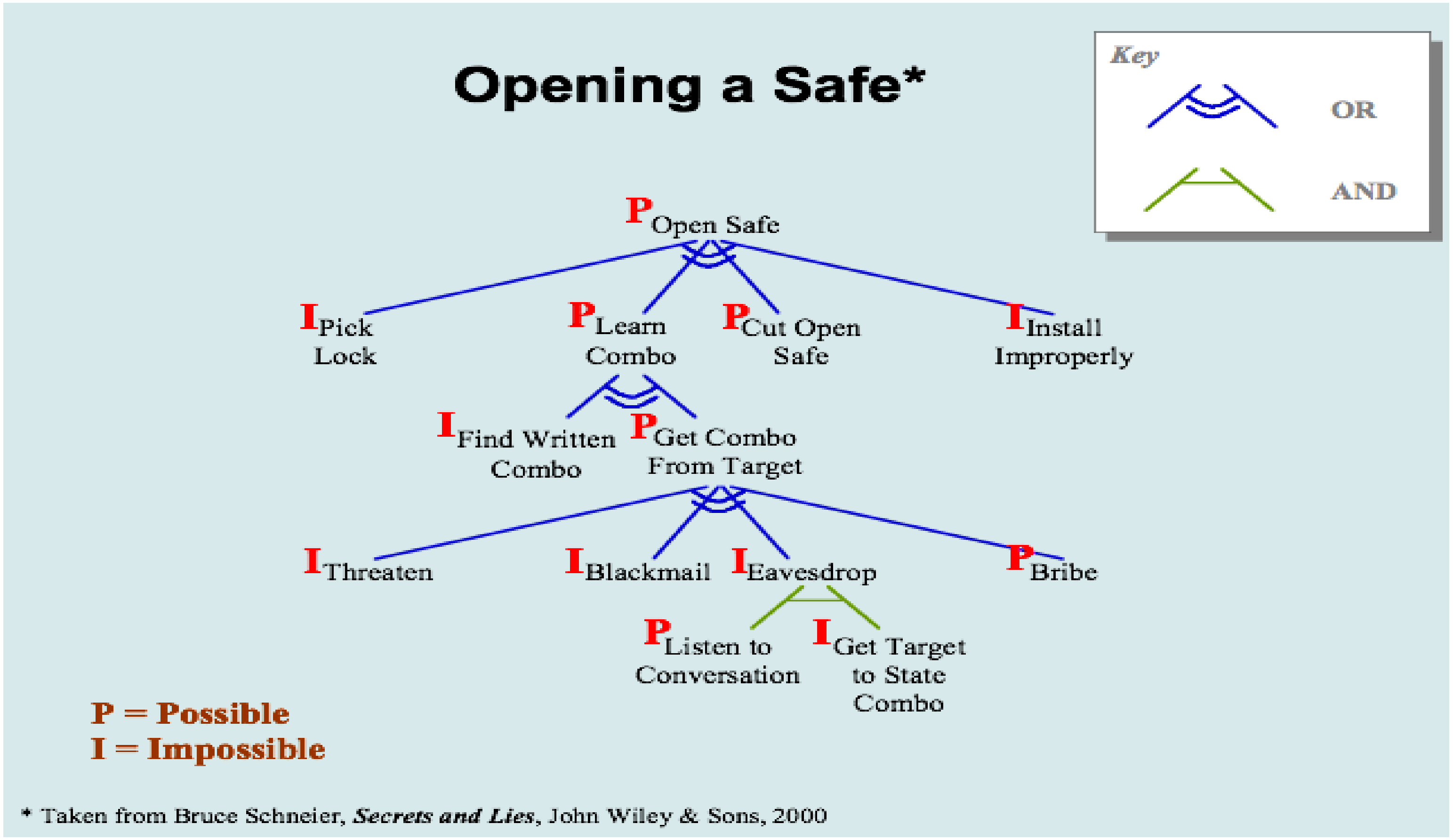
- Step 3: Identifying Threats
  - Identify threats that apply only to the application Context & scenarios generated in the previous step
  - 2 Approaches to Identify Threats
    - Use Attack Trees (CI4AM)
    - Think like an Attacker (STRIDE/DREAD, OCTAVE etc)
  - Create the threat list
    - SQL Injection
    - XSS
    - Replay Attacks
    - MITM
    - Eavesdropping

Threat Modeling Process

- Attack Tree
  - Multiple factors realize a threat
    - Weakness at multiple places
    - Attack Tress help in identifying these combination
  - Consider target as the destination
    - Often multiple steps needed in some logical sequence
    - Often multiple routes can be traveled to reach it
    - Describe attacks as a tree of nodes (sub-trees may be shared among attack scenarios)
  - Attack has to start out on that route
    - Often the most difficult
    - Can be a long way from the destination (those who need to protect the destination must understand this)

Threat Modeling Process

- Attack Tree – Sample



* Taken from Bruce Schneier, ***Secrets and Lies***, John Wiley & Sons, 2000

Threat Modeling Process

- STRIDE – For Threat Modeling
- DREAD – For Threat Ranking

	STRIDE means	
S	Spoofing	Impersonating another person/process
T	Tampering	Unauthorized Alterations
R	Repudiation	Denying claims/unproven actions
I	Information Disclosure	Exposure to unauthorized person/process
D	Denial of Service	Service unavailability
E	Elevation of Privileges	Increasing person/process access level

	DREAD means	
D	Damage Potential	What will be the impact on exploitation?
R	Reproducibility	What is the ease of recreating the attack/exploit?
E	Exploitability	What minimum skill level is needed to launch?
A	Affected Users	How many users will be potentially impacted?
D	Discoverability	What is the ease of finding the vulnerability?

Threat Modeling Process

- Common Attacks related to STRIDE

STRIDE	Attack
Spoofing	Cookie Replay Session Hijacking CSRF
Tampering	XSS SQL Injection
Repudiation	Audit Log Deletion Insecure Backup
Information Disclosure	Eavesdropping Verbose Exception
Denial of Service	Website defacement
Elevation of Privilege	Logic Flow Attacks

Threat Modeling Process

- Step 4: Identifying Vulnerabilities
  - Identify vulnerabilities that apply only to the threats generated in the previous step
  - Vulnerabilities identified should be factored to
    - Shape the design of the application
    - Generate security test cases for testing in development stage (Typical vulnerabilities may be):
      - Weak Encryption
      - Clear Text Credentials
      - Unhandled Exception
      - Dynamic SQL
      - Long Session Timeouts

Threat Modeling Process

- Step 5: Rank The Threats
  - Ranking provides the necessary insight to the risk of the threat
  - Risk can be
    - Accepted
    - Mitigated
    - Transferred
    - Ignored (Avoid this)
  - Two methods for ranking
    - Probability * Impact Ranking
    - DREAD Average Ranking

Threat Modeling Process

- Probability * Impact Ranking
  - Heuristic
  - Define a probability of attack/exploit
    - High (15)
    - Medium (10)
    - Low (5)
  - Define Impact (Seriousness when attacked/exploited)
    - High (15)
    - Medium (10)
    - Low (5)

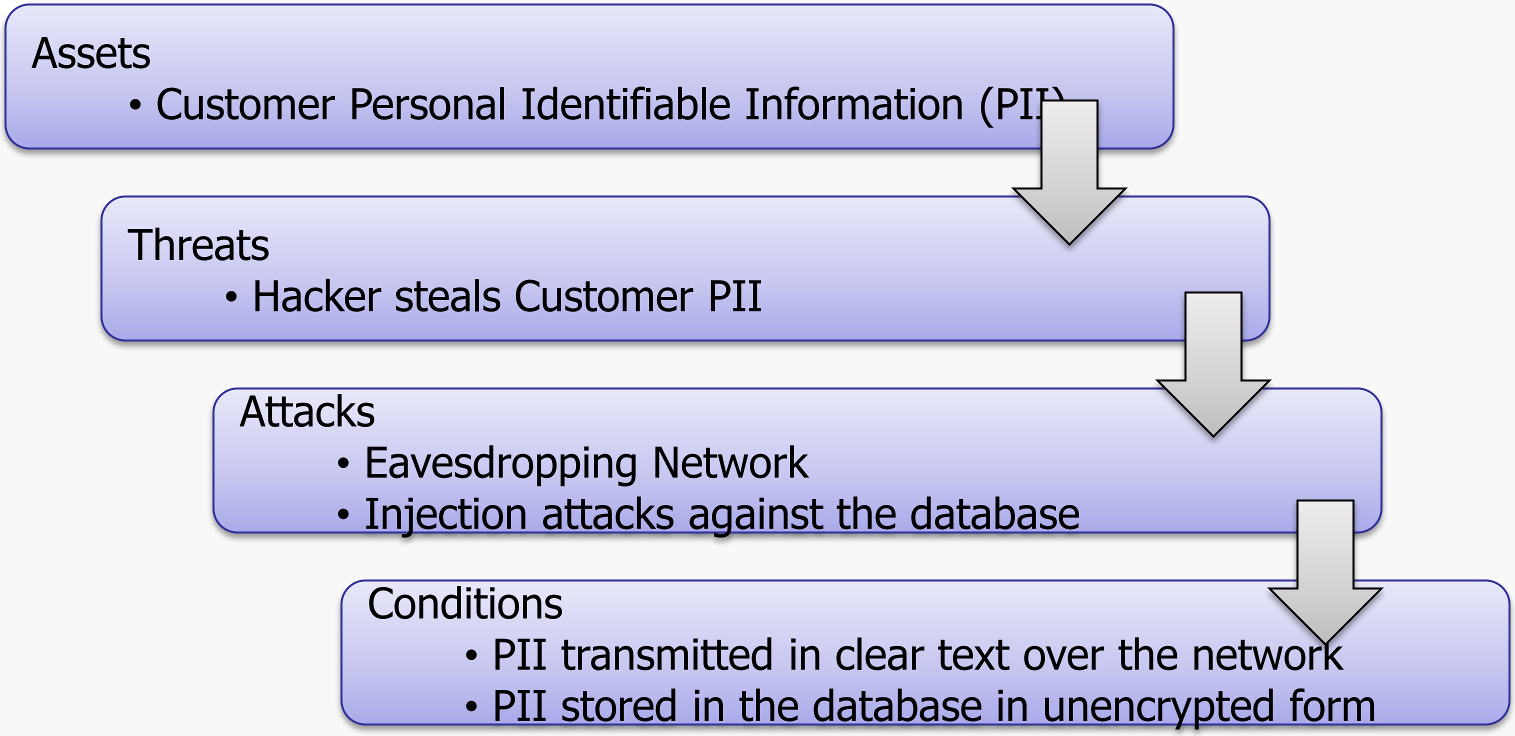
Threat Modeling Process

- DREAD Average Ranking
  - Damage Potential – If a threat exploit occurs, evaluate the damage caused
    - 0 = Nothing
    - 5 = Individual user data compromised
    - 10=Complete system or data destruction
  - Reproducibility – How easy is it to reproduce the threat exploit?
    - 0 = Very hard or impossible even for administrators/DBAs
    - 5 = One or two steps required, may need an authorized user
    - 10 = Just a web browser is enough
  - Exploitability – What is needed to exploit this threat?
    - 0 = Advanced programming & networking knowledge
    - 5 = Malware exists on the Net, or any tolls avaiable
    - 10 = Just a web browser is enough
  - Affected Users – How many users are affected?
    - 0 = None
    - 5 = Some users, but not all
    - 10 = All users
  - Discoverability – How easy is it to discover the threat
    - 0 = Very hard or impossible; needs source code or admin access
    - 5 = Can figure it out by guessing or monitoring network traces
    - 10 = Information is visible in the web browser or address bar or in the form or as a hidden variable
- Finally use the formula:
  - Average Threat Ranking = (D + R + E + A + D)/5

Test Focused Threat Modeling

- For those who don’t have a mature SDLC or Agile Methodologies
- For those who don’t have threat models done at design time but have deployed the applications
- A lightweight custom threat modeling methodology
- Is recommended to supplement threat modeling done at design time

Test Focused Threat Modeling



Assets
- Customer Personal Identifiable Information (PII)

Threats
- Hacker steals Customer PII

Attacks
- Eavesdropping Network
- Injection attacks against the database

Conditions
- PII transmitted in clear text over the network
- PII stored in the database in unencrypted form

Alternative Threat Models

- Trike
  - Trike is a threat modeling framework with similarities to the Microsoft threat modeling processes. However, Trike differs because it uses a risk based approach with distinct implementation, threat, and risk models, instead of using the STRIDE/DREAD aggregated threat model (attacks, threats, and weaknesses). From the Trike paper, Trike's goals are:
    - With assistance from the system stakeholders, to ensure that the risk this system entails to each asset is acceptable to all stakeholders.
    - Be able to tell whether we have done this.
    - Communicate what we've done and its effects to the stakeholders.
    - Empower stakeholders to understand and reduce the risks to them and other stakeholders implied by their actions within their domains.

Alternative Threat Models

- **AS/NZS 4360:2004 Risk Management**
  - The Australian/New Zealand Standard AS/NZS 4360, first issued in 1999, and revised in 2004, is the world's first formal standard for documenting and managing risk and is still one of the few formal standards for managing it. The standard's approach is simple (it's only 28 pages long), flexible, and iterative. Furthermore, it does not lock organizations into a particular risk management methodology, provided the methodology fulfils the AS/NZS 4360 five steps. It also provides several sets of risk tables as examples, and allows organizations to freely develop and adopt their own.
  - **The five steps of the AS/NZS 4360 process are:**
    - **Establish Context:** Establish the risk domain, i.e., which assets/systems are important?
    - **Identify the Risks:** Within the risk domain, what specific risks are apparent?
    - **Analyze the Risks:** Look at the risks and determine if there are any supporting controls in place.
    - **Evaluate the Risks:** Determine the residual risk.
    - **Treat the Risks:** Describe the method to treat the risks so that risks selected by the business will be mitigated.
  - AS/NZS 4360 assumes that risk will be managed by an ***operational risk group***, and that the organization has adequate skills and risk management resources in house to identify, analyze, and treat the risks.
- **The advantages of AS/NZS 4360:**
  - AS/NZS 4360 works well as a risk management methodology for organizations requiring Sarbanes-Oxley compliance.
  - AS/NZS 4360 works well for organizations that prefer to manage risks in a traditional way, such as just using likelihood and consequence to determine an overall risk.
  - AS/NZS 4360 is familiar to most risk managers worldwide, and your organization may already have implemented an AS/NZS 4360 compatible approach.
  - You are an Australian organization, and may be required to use it if you are audited on a regular basis, or to justify why you aren't using it. Luckily, the STRIDE/DREAD model discussed earlier is AS/NZS 4360 compatible.
- **The limitations of AS/NZS 4360:**
  - The AS/NZS 4360 approach works best for business or systemic risks than for technical risks.
  - AS/NZS 4360 does not define the methodology to perform a structured threat risk modeling exercise.
  - As AS/NZS 4360 is a generic framework for managing risk, it does not provide any structured method to enumerate web application security risks.
- Although AS/NZS 4360 may be used to rank risks for security reviews, the lack of structured methods of enumerating threats for web applications makes it less desirable than other methodologies described earlier.

Alternative Threat Models

- **CVSS**
  - The US Department of Homeland Security (DHS) established the NIAC Vulnerability Disclosure Working Group, which incorporates input from Cisco Systems, Symantec, ISS, Qualys, Microsoft, CERT/CC, and eBay. One of the group's outputs is the ***Common Vulnerability Scoring System (CVSS).***
- **The advantages of CVSS:**
  - You have just received notification from a security researcher or other source that your product has vulnerability, and you wish to ensure that it has an accurate and normalized severity rating, so as to alert your customers to the appropriate level of action required when you release the patch.
  - You are a security researcher, and have found several threat exploits within an application. You would like to use the CVSS ranking system to produce reliable risk rankings, to ensure that the ISV will take the exploits seriously as indicated by their rating.
  - CVSS has been recommended by the working group for use by US Government departments. However, it is unclear if it will become policy or be widely adopted at the time of this writing.
- **The limitations of CVSS:**
  - CVSS does not find or reduce the attack surface area (i.e. design flaws), or help enumerate risks within any arbitrary piece of code, as it is just a scoring system, not a modeling methodology.
  - CVSS is more complex than STRIDE/DREAD, as it aims to calculate the risk of announced vulnerabilities as applied to deployed software and environmental factors.
  - The CVSS risk ranking is complex – a spreadsheet is required to calculate the risk components as the assumption behind CVSS is that a specific vulnerability has been identified and announced, or a worm or Trojan has been released targeting a small number of attack vectors.
  - The overhead of calculating the CVSS risk ranking is quite high if applied to a thorough code review, which may have 250 or more threats to rank.

Alternative Threat Models

- **OCTAVE**
  - OCTAVE is a heavyweight risk methodology approach originating from Carnegie Mellon University's Software Engineering Institute (SEI) in collaboration with CERT. OCTAVE focuses on organizational risk, not technical risk. OCTAVE comes in two versions: Full OCTAVE, for large organizations, and OCTAVE-S for small organizations, both of which have specific catalogs of practices, profiles, and worksheets to document the modeling outcomes.
- **OCTAVE is popular with many sites and is useful when:**
  - Implementing an organizational culture of risk management and controls becomes necessary.
  - Documenting and measuring business risk becomes timely.
  - Documenting and measuring the overall IT security risk, particularly as it relates to the corporate IT risk management, becomes necessary.
  - When documenting risks surrounding complete systems becomes necessary.
  - To accommodate a fundamental reorganization, such as when an organization does not have a working risk methodology in place, and requires a robust risk management framework to be put in place.
- **The limitations of OCTAVE are:**
  - OCTAVE is incompatible with AS/NZS 4360, as it mandates Likelihood = 1 (i.e., It assumes a threat will always occur) and this is inappropriate for many organizations. OCTAVE-S makes the inclusion of this probability optional, but this is not part of the more comprehensive OCTAVE standard.
  - Consisting of 18 volumes, OCTAVE is large and complex, with many worksheets and practices to implement.
  - It does not provide a list of "out of the box" practices for assessing and mitigating web application security risks.

Estimating Threat Modeling for Applications

- I feel that it is a standard measure – 6 to 8 weeks
- Just came up with some random thoughts to brainstorm
  - Count of Use case based
    - Define a standard based on the number of use cases (All numbers below are just place holders (examples)…Need to define ranges and the values appropriately)
      - 0 to 50 – 4 weeks
      - 51 – 100 – 5 weeks
      - 101 – 150 – 6 weeks
      - 151 – 200 – 7 weeks
      - 201 – 250 - 8 weeks and so on…
  - Project Type & Project Size based
    - Application Development
      - Small Project – 4 weeks
      - Medium Project – 5 weeks
      - Large Project – 6 weeks
      - Enterprise Projects (Extra Large or XXL) – 8 weeks
    - Application Maintenance
      - Small Project – 4 weeks
      - Medium Project – 5 weeks
      - Large Project – 6 weeks
      - Enterprise Projects (Extra Large or XXL) – 8 weeks
  - Dependent Factors Based
    - No external dependency – 4 weeks
    - Complex Internal services – 8 weeks
    - External domain services – 8 weeks

QUESTIONS?

THANK YOU