



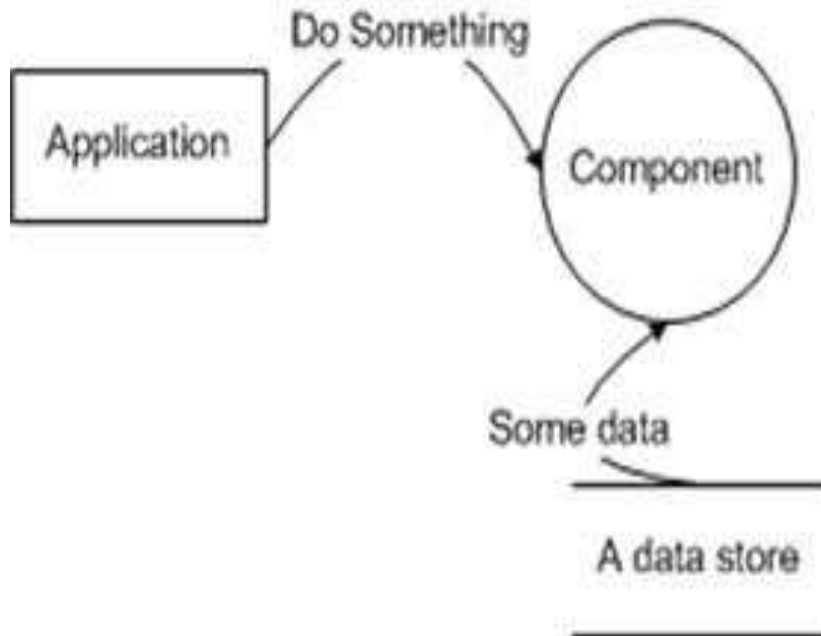
Amidst a sea of threats, how ready is your enterprise to navigate beyond risk?

Real World Threat Modeling Using the PASTA Methodology

Tony UcedaVelez
Managing Partner, VerSprite
OWASP AppSec EU 2012

Why Threat Modeling?

Threat Dissection



Targeted Analysis

- Focused on understanding targeted attacks
- You can't mitigate all of your threats
- So, what are your most likely threats to your published sites/ services?

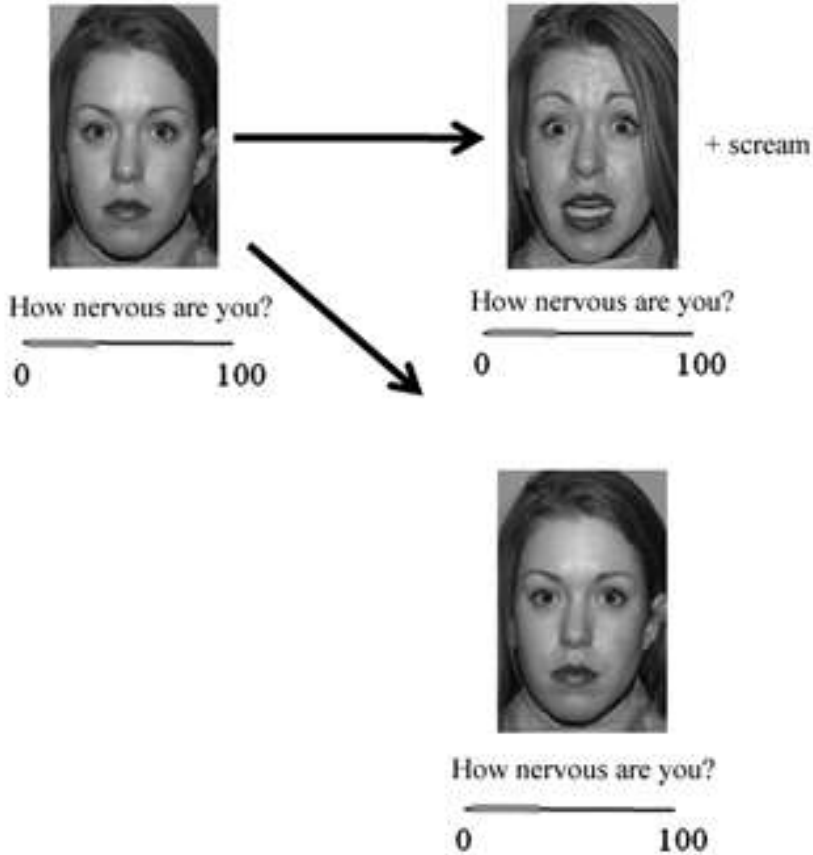
Taxonomy of Terms

Asset



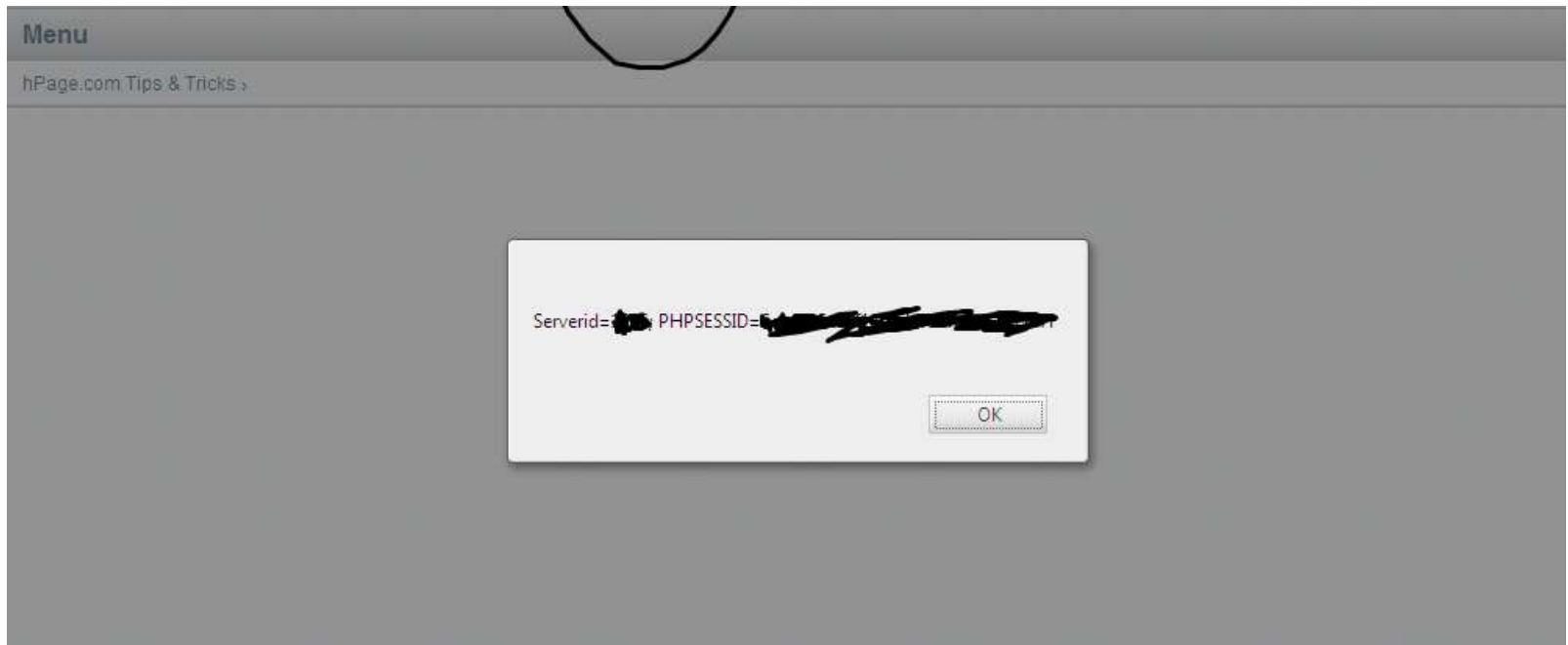
Asset. An asset is a resource of value. It varies by perspective. To your business, an asset might be the availability of information, or the information itself, such as customer data. It might be intangible, such as your company's reputation.

Threat



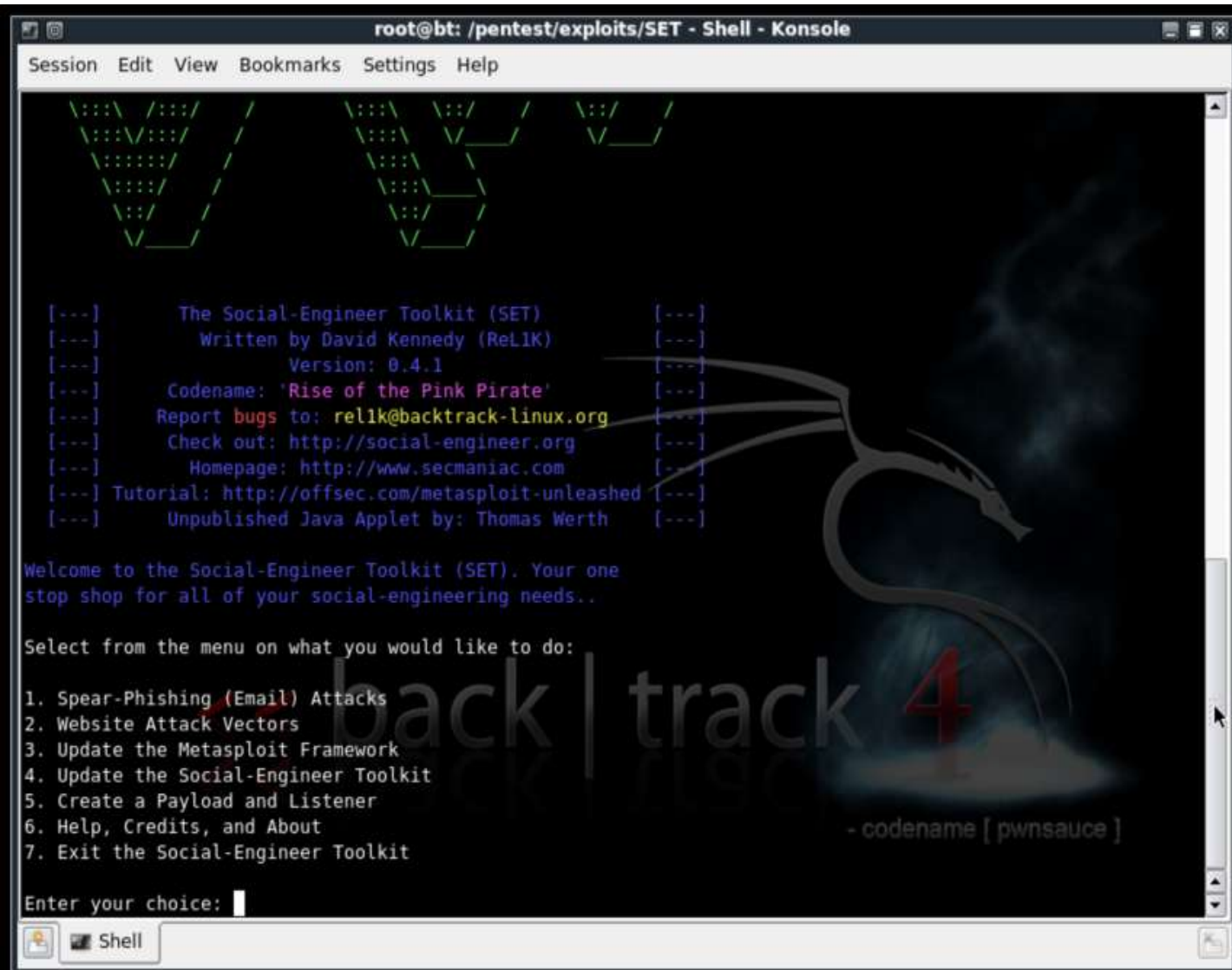
Threat. A threat is an undesired event. A potential occurrence, often best described as an effect that might damage or compromise an asset or objective. Relative to each site, industry, company; more difficult to uniformly define.

Vulnerability (Weakness)



Vulnerability. A vulnerability is a weakness in some aspect or feature of a system that makes an exploit possible. Vulnerabilities can exist at the network, host, or application levels and include operational practices.

Attack



```
root@bt: /pentest/exploits/SET - Shell - Konsole
Session Edit View Bookmarks Settings Help

[---] The Social-Engineer Toolkit (SET) [---]
[---]   Written by David Kennedy (ReLlK) [---]
[---]   Version: 0.4.1 [---]
[---]   Codename: 'Rise of the Pink Pirate' [---]
[---]   Report bugs to: rellk@backtrack-linux.org [---]
[---]   Check out: http://social-engineer.org [---]
[---]   Homepage: http://www.secmaniac.com [---]
[---]   Tutorial: http://offsec.com/metasploit-unleashed [---]
[---]   Unpublished Java Applet by: Thomas Werth [---]

Welcome to the Social-Engineer Toolkit (SET). Your one
stop shop for all of your social-engineering needs..

Select from the menu on what you would like to do:

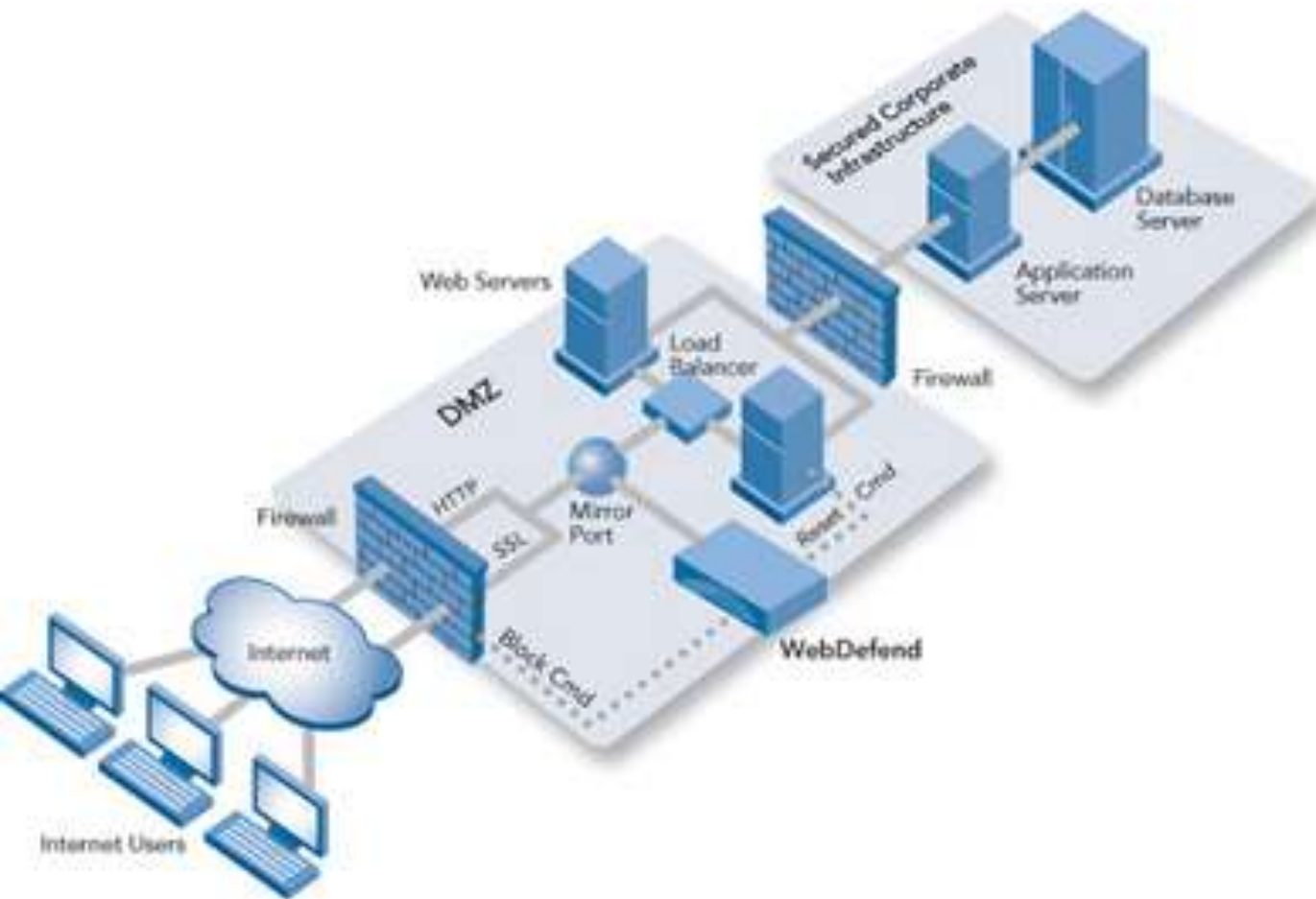
1. Spear-Phishing (Email) Attacks
2. Website Attack Vectors
3. Update the Metasploit Framework
4. Update the Social-Engineer Toolkit
5. Create a Payload and Listener
6. Help, Credits, and About
7. Exit the Social-Engineer Toolkit

Enter your choice: |

back | track 4
- codename [ pwnsauce ]
```

Attack (or exploit).
An attack is an action taken that utilizes one or more vulnerabilities to realize a threat.

Countermeasures



Countermeasure. Countermeasures address vulnerabilities to reduce the probability of attacks or the impacts of threats. They do not directly address threats; instead, they address the factors that define the threats.

Use Case

Login to NDS Tree

Available NDS Trees:

- 000MM
- ADG-FILEMAKER
- ALS_TREE
- API-500-CLIENT-6

User Name:

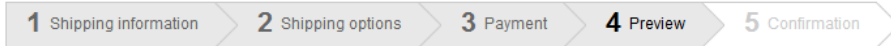
Password:

Use Case.

Functional, as designed function of an application.

Abuse Case

Checkout



Place this order

Shipping address:

Web Admin
Shipping_Company
Shipping_AddressLine1
Shipping_AddressLine2
Shipping_City, 1234 AB
Netherlands
+31(0) 123-456789
[Change](#)

Billing address:

Web Admin
Shipping_Company
Shipping_AddressLine1
Shipping_AddressLine2
Shipping_City, 1234 AB
Netherlands
+31(0) 123-456789
[Change](#)

Shipping method:

Ship_WW
[Change](#)

Payment method:

Offline [Change](#)

Product	Options	Price	Quantity	Subtotal
title		€ 33,00 (incl Tax)	1	€ 33,00
ProductB		€ 2,65 € 5,39 (incl Tax)		€ 2,65
SteveV and the magical labels		€ 1,06 (incl Tax)	1	€ 1,06
3 Products Subtotal (Excluding VAT Taxes):				€ 33,50
VAT Tax:				€ 3,21
Shipping:				€ 10,00
Total Before Tax:				€ 43,50
Sales Tax:				€ 3,21
Total: € 46,71				

Place this order

Search

Shopping cart (3)

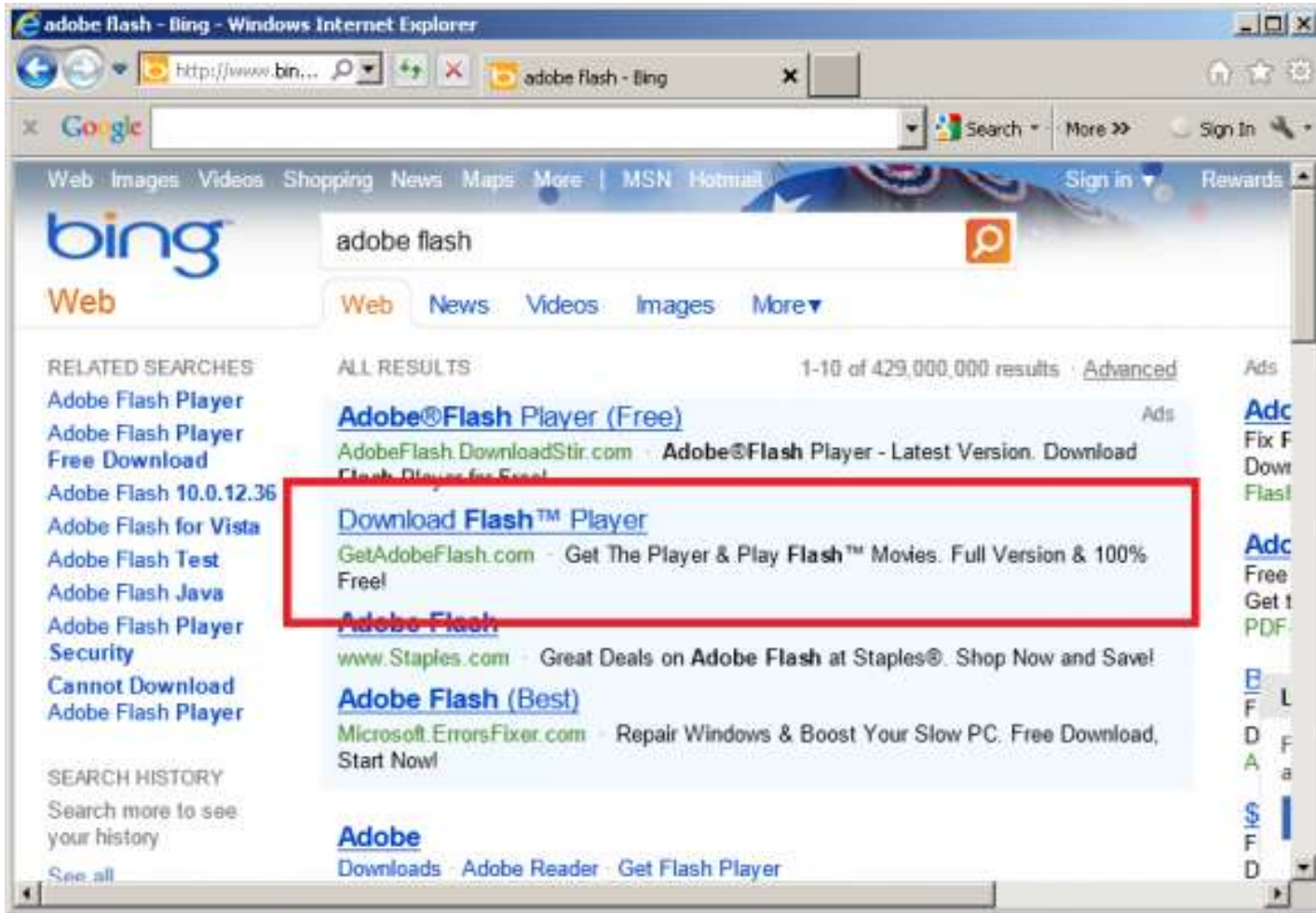
	title	1 x € 33,00
	ProductB	1 x € 2,65
	SteveV and the magical labels	1 x € 1,06

3 products Subtotal: € 36,71

[View shopping cart](#) [Checkout](#)

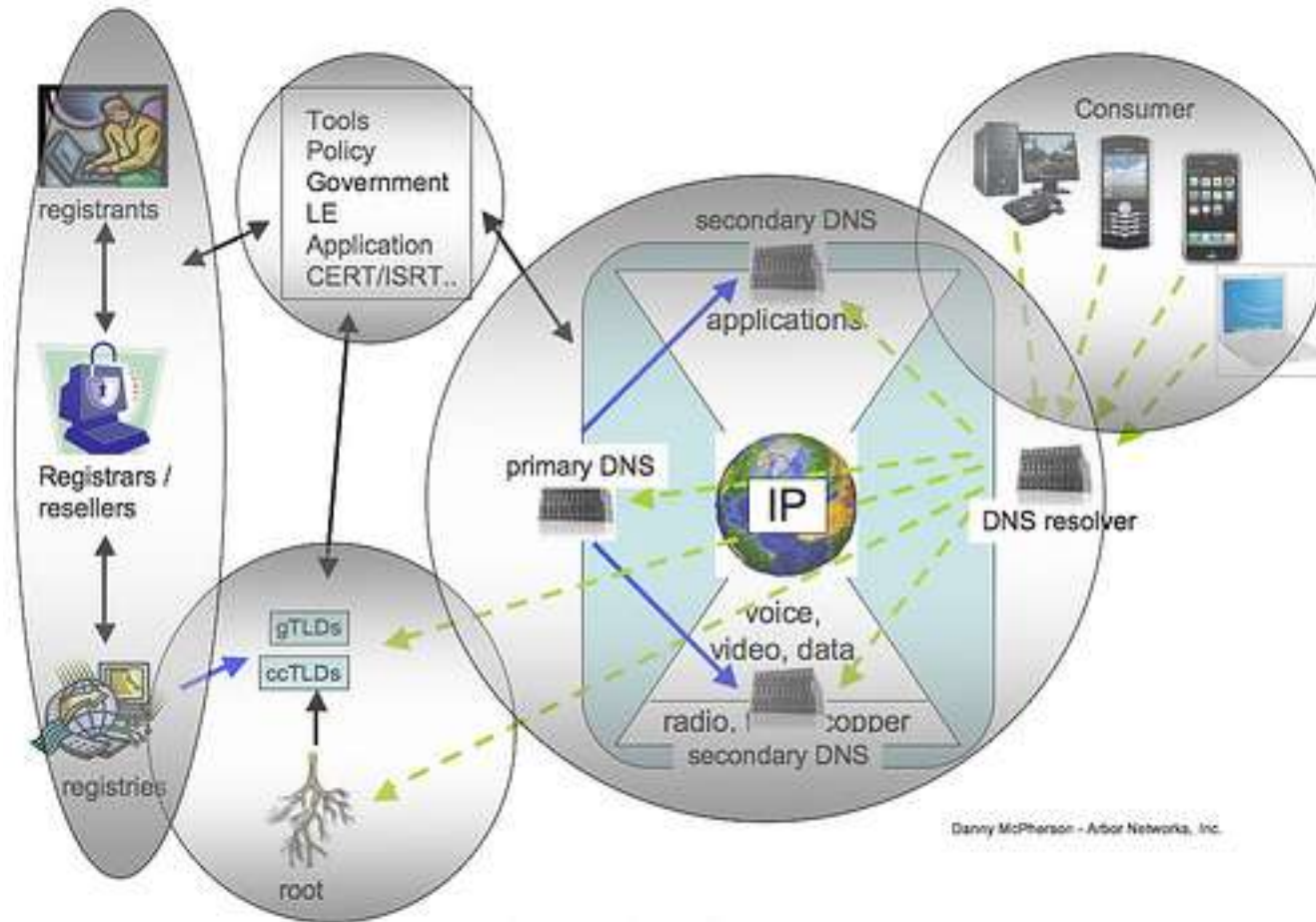
Abuse Case.
Deliberate
abuse of
functional use
cases in order to
yield
unintended
results

Attack Vector



Attack Vector.
Point & channel for which attacks travel over (card reader, form fields, network proxy, client browser, etc)

Attack Surface



Attack Surface.

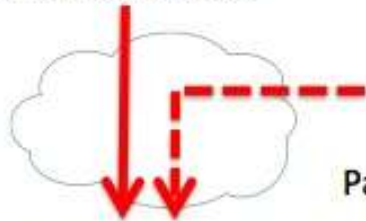
Logical area (browser stack, infrastructure components, etc) or physical area (hotel kiosk)

Actor (Threat Agent)

External Agent



External Attacker

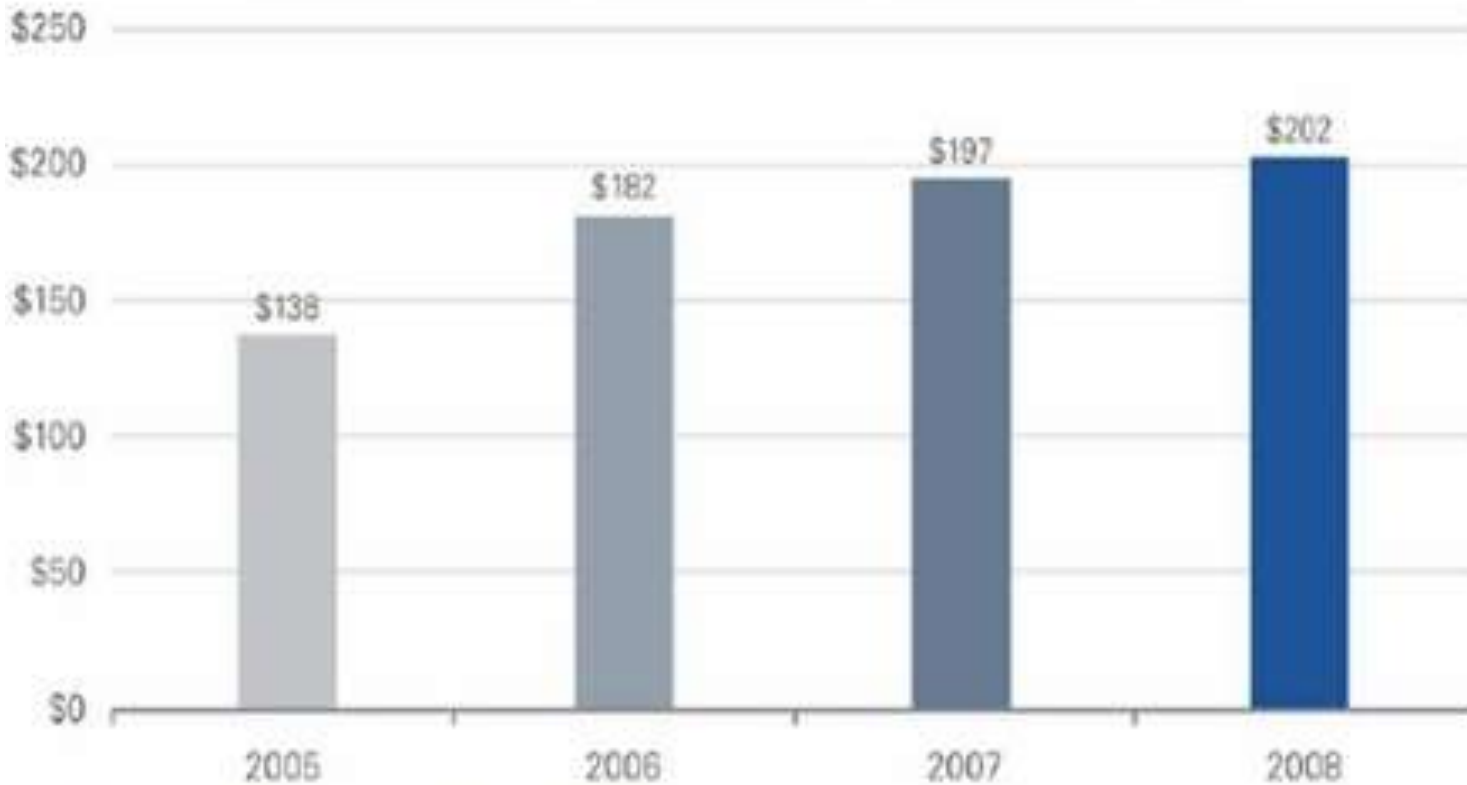


Victim Organization

Actor. Legit or adverse caller of use or abuse cases.

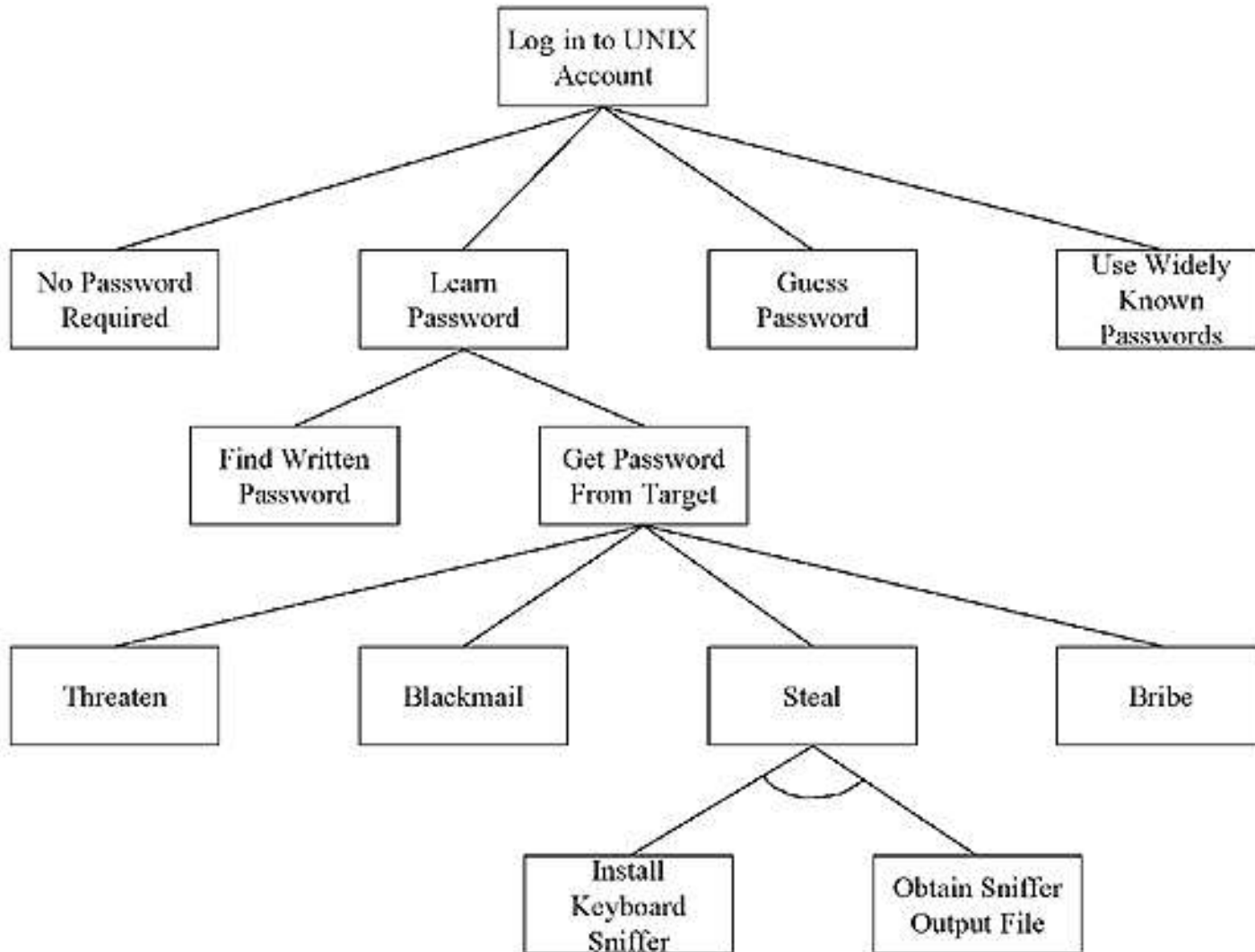
Impact

Figure 1: Average per-record cost of a data breach, 2005–2008



Impact. Value of [financial] damage possibly sustained via attack. Relative.

Attack Tress



Attack Tree.

Diagram of relationship amongst asset-actor-use case-abuse case-vuln-exploit-countermeasure

What is PASTA?

What is PASTA?

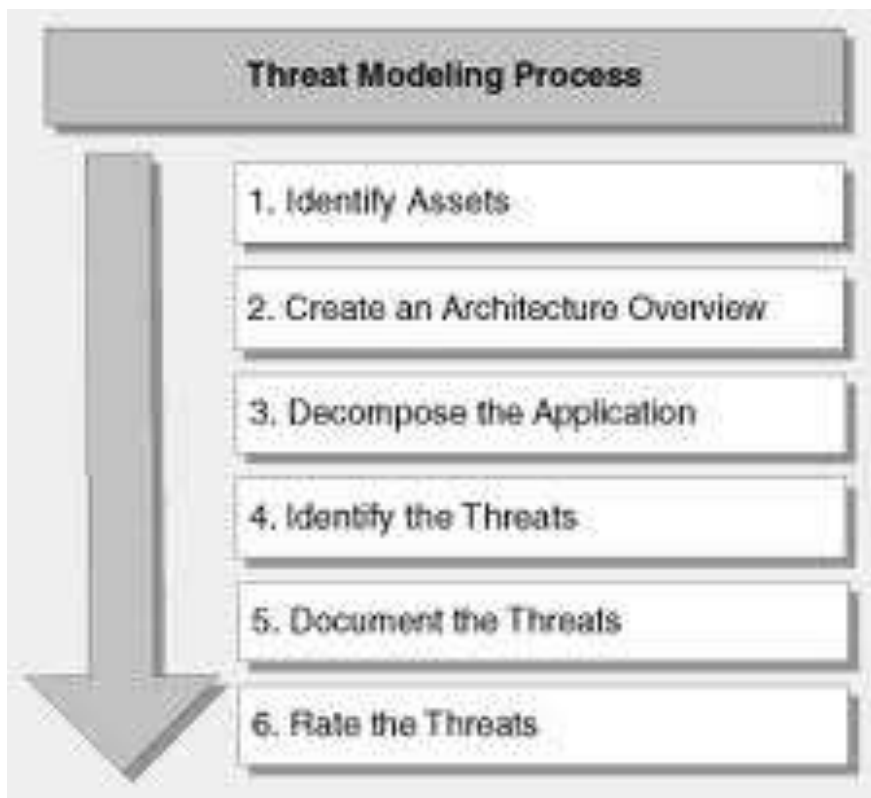
- **Process for Attack Simulation & Threat Analysis**
 - Integrated application threat analysis
 - Application threat modeling methodology
 - Risk or asset based approach; great for business integration
 - 7 stages building up to impact of threat to application & business.
- **Aimed at addressing most viable threats to a given application target**

Why should I eat this?

- **Current menu of application testing doesn't provide a full security meal**
 - Pen Tests: Exploit driven
 - Risk Assessments: Subjective; lacks meat
 - Static Analysis: Weakness, flaw driven; disregards threats, narrow focus
 - Vuln Scans: (C'mon! As if this could provide a decent meal!)
 - Security testing deliverables are adversarial
 - Integrated disciplines are needed via a unifying methodology
- **Better form of risk analysis w/ more substance**
- **Encapsulates other security efforts**

Threat Modeling Comparisons

MS Approach



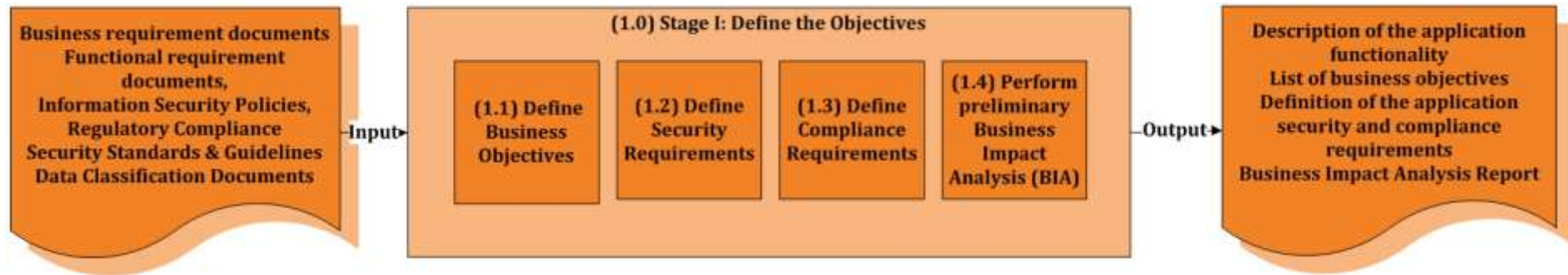
Process for Attack Simulation & Threat Analysis (PASTA)



STAGE I

Define The Business & Security Objectives: “Capture requirements for the analysis and management of web based risks”

Stage 1 Walkthru – Understand Biz Objectives



Business Objectives affect Web Apps

- Function req of supercookies (marketing)
 - Persistent storage of PII
- Easily accessible web services for internal APIs
 - ‘Internal’ lets security guard down w/ authentication
- Over-scoping of functional requirements
 - Orphaned features that lose maintenance
- Change Management System Web App Example
 - Biz Objective: Track & Manage Changes Across Groups; Easily accessible; Control Changes; Role based access
 - Discovered Threats/ Vulnerabilities: Internet accessible, elevation of privileges,

Threat Modeling Stage 1 Artifact

Application Profile: Online Banking Application

General Description	The online banking application allows customers to perform banking activities such as financial transactions over the internet. The type of transactions supported by the application includes bill payments, wires, funds transfers between customer's own accounts and other bank institutions, account balance-inquires, transaction inquires, bank statements, new bank accounts loan and credit card applications. New online customers can register an online account using existing debit card, PIN and account information. Customers authenticate to the application using username and password and different types of Multi Factor Authentication (MFA) and Risk Based Authentication (RBA)
Application Type	Internet Facing
Data Classification	Public, Non Confidential, Sensitive and Confidential PII
Inherent Risk	HIGH (Infrastructure , Limited Trust Boundary, Platform Risks, Accessibility)
High Risk Transactions	YES
User roles	Visitor, customer, administrator, customer support representative
Number of users	3 million registered customers

Merging Business & Security Requirements

Project Business Objective	Security and Compliance Requirement
Perform an application risk assessment to analyze malware banking attacks	Risk assessment need to assess risk from attacker perspective and identify on-line banking transactions targeted by the attacks
Identify application controls and processes in place to mitigate the threat	Conduct architecture risk analysis to identify the application security controls in place and the effectiveness of these controls. Review current scope for vulnerability and risk assessments.
Comply with FACT Act of 2003 and FFIEC guidelines for authentication in the banking environment	Develop a written program that identifies and detects the relevant warning signs – or “red flags” – of identity theft. Perform a risk assessment of online banking high risk transactions such as transfer of money and access of Sensitive Customer Information
Analyze attacks and the targets that include data and high risk transactions	Analyze attack vectors used for acquisition of customers’ PII, logging credentials and other sensitive information. Analyze attacks against user account modifications, financial transactions (e.g. wires, bill-pay), new account linkages
Identify a Risk Mitigation Strategy That Includes Detective and Preventive Controls/Processes	Include stakeholders from Intelligence, IS, Fraud/Risk, Legal, Business, Engineering/Architecture. Identify application countermeasures that include preventive, detective (e.g. monitoring) and compensating controls against malware-based banking Trojan attacks

Baking in GRC

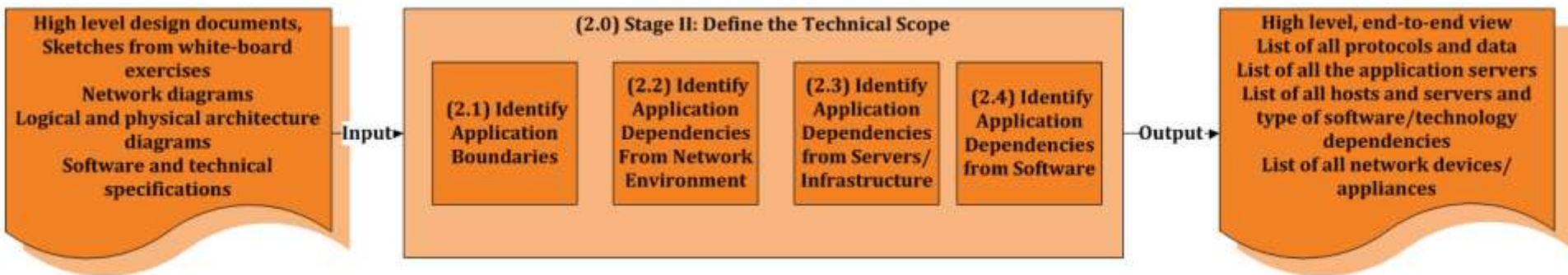
- Serve as inherent countermeasures in the form of people, process, technology
 - Policies (for people)
 - Standards (for technology)
- Prior risk assessments help build app risk profile
 - Historical RAs provide prior risk profile of app
- Regulatory landscape taken into consideration, but not the driver
 - Key here is to not retrofit compliance; more costly
- Web Related Example:
 - Tech: Using Nessus OWASP template to audit for PHP & ColdFusion hardening guidelines
 - OWASP Input Validation Cheat Sheets
 - CIS Web Technology Benchmarks



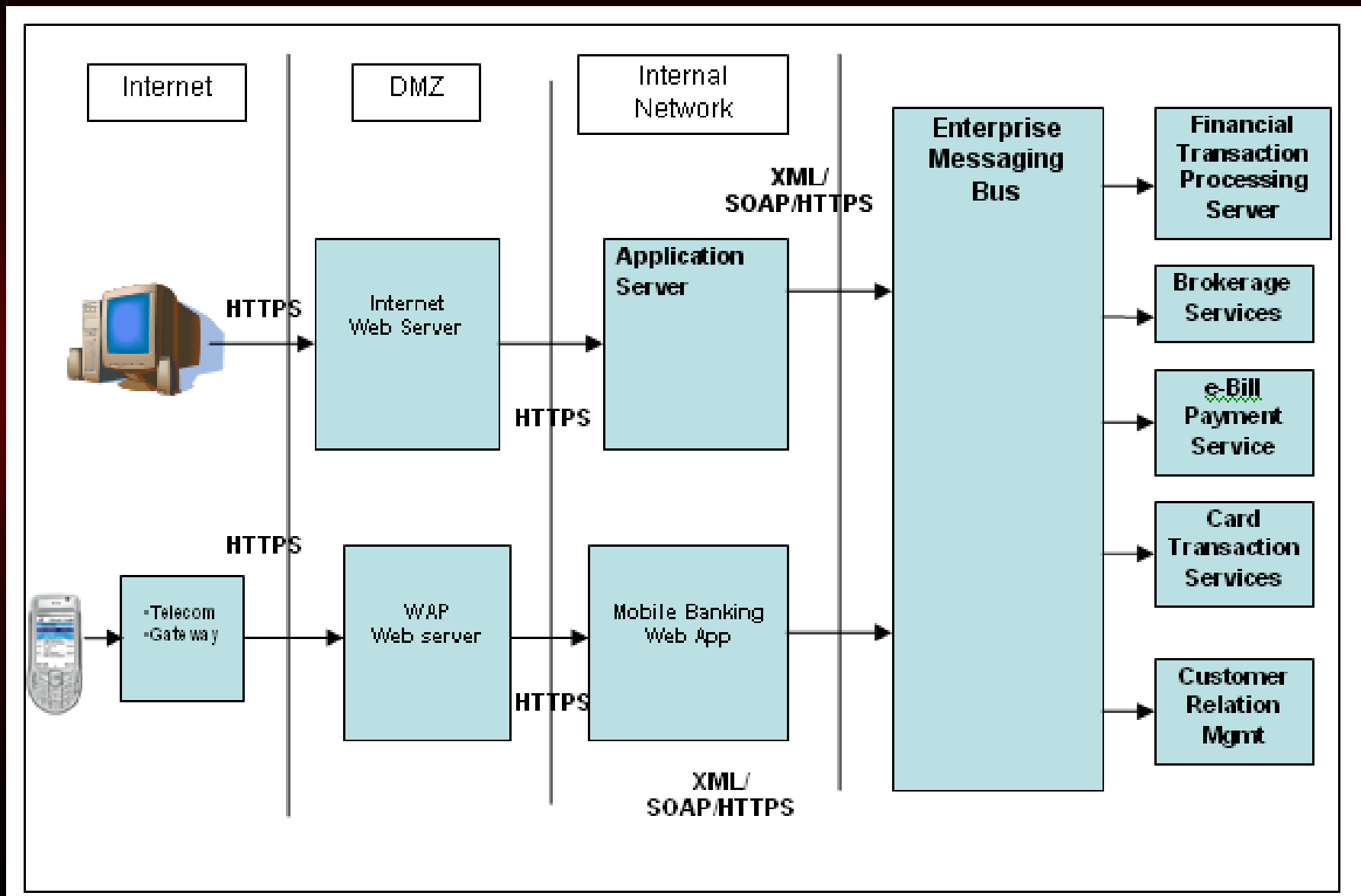
STAGE II

Define The Technical Scope: "Defining the scope of technical assets/ components for which threat enumeration will ensue"

Stage 2 Walkthru – Define Tech Scope



The Application Architecture Scope



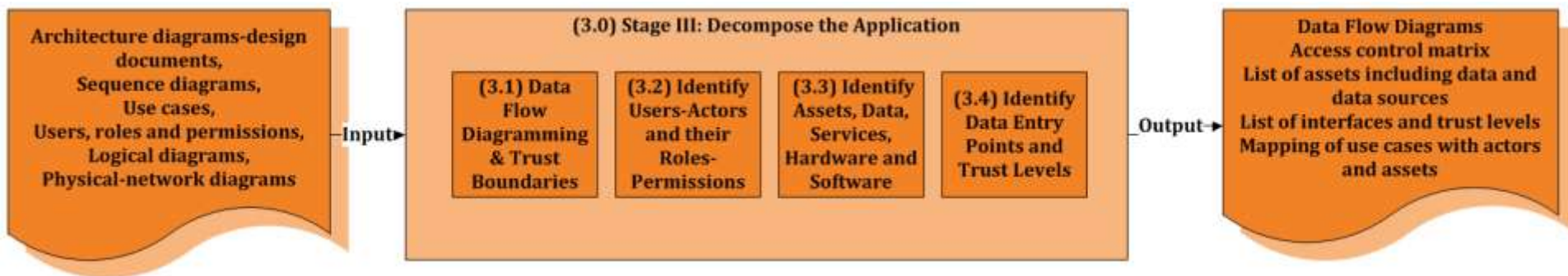
Technical Scope Definition

- Define the scope from design artifacts:
 - **Application components** with respect to the application tiers (presentation, application, data)
 - **Network topology**
 - **Protocol/services** being used/exposed from/to the user to/from the back end (e.g. data flow diagrams)
 - **Use case scenarios** (e.g. sequence diagrams)
- Model the application in support of security architecture risk analysis
 - **The application assets** (e.g. data/services at each tier)
 - **The security controls of the application** (e.g. authentication, authorization, encryption, session management, input validation, auditing and logging)
 - **Data interactions** between the user of the application and between servers for the main use case scenarios (e.g. login, registration, query etc)
- End of this stage results in inherent countermeasures (people, process, technology)

STAGE III

Decompose the Application :”Identify the application controls that protect high risk web transactions sought by adversaries”

Stage 3 Walkthru – App Decomposition

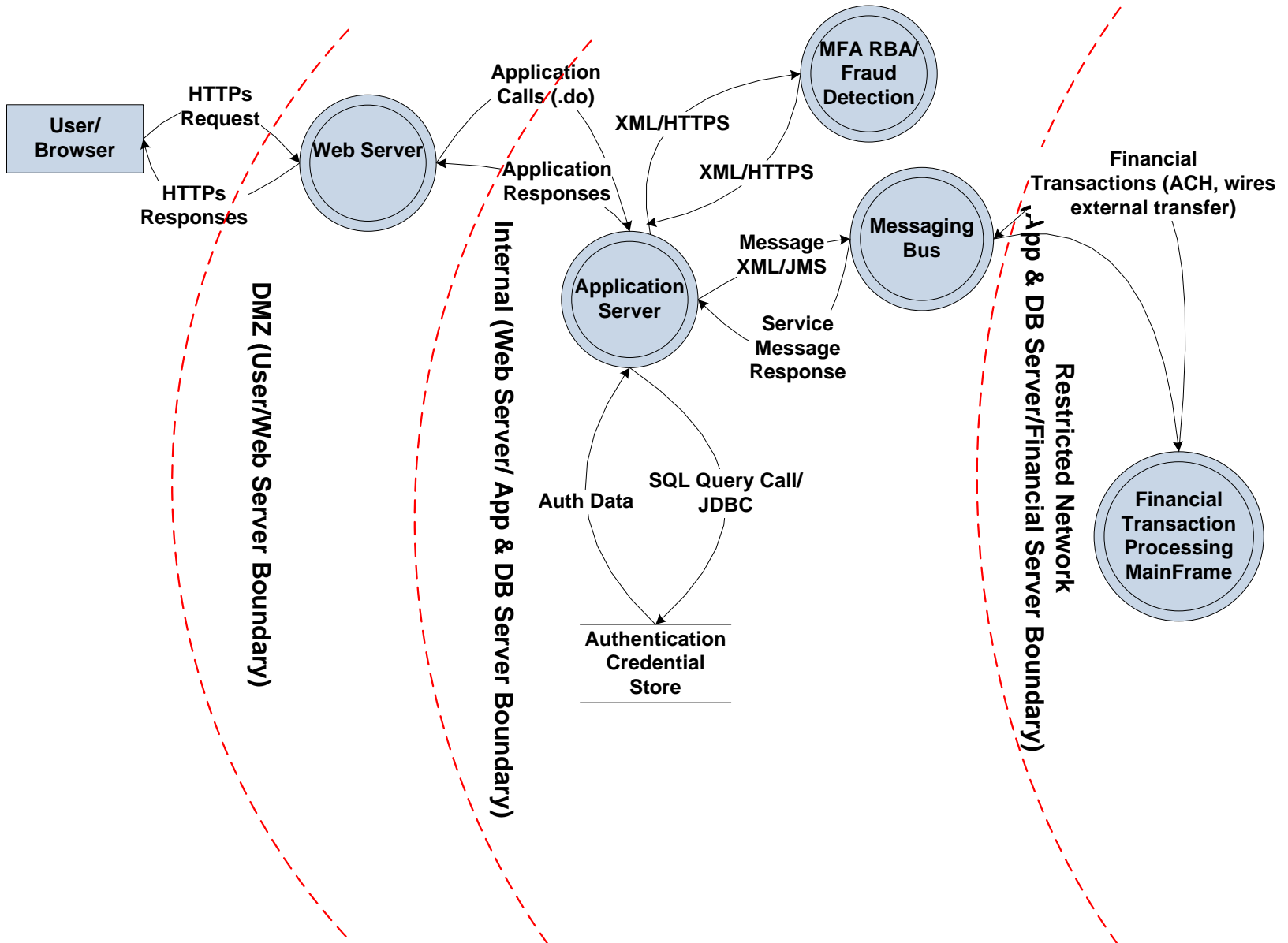


What's your web app cooking with?

- What/ who are the actors?
- What calls do the actors make?
 - Key aspect of this phase
- Enumerate all use cases
- Define trust boundaries (implicit vs explicit trust)
 - Domains, networks, hosts, services, etc
- Further identify data sources and their relevant data flows



On-line Banking Application Data Flow Diagram (DFD) Example



Transactional Security Control Analysis

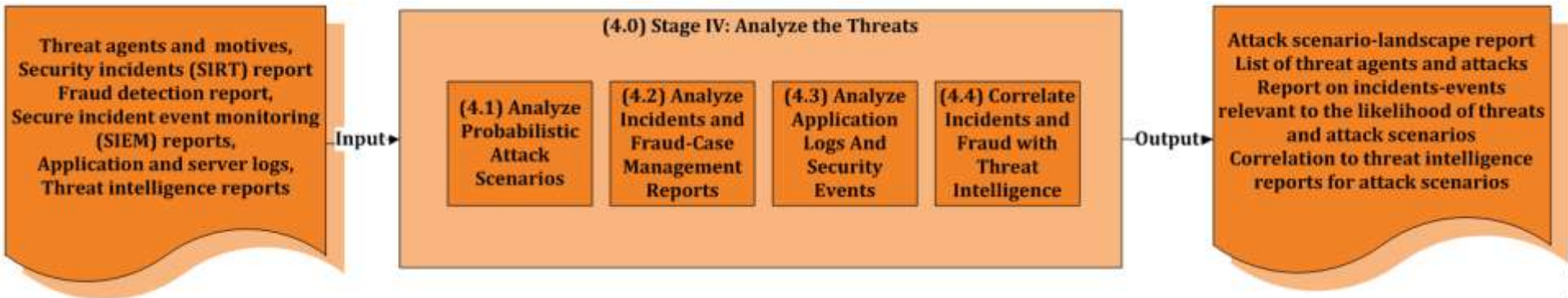
Online Banking Application Transaction Analysis			Data Input Validation (Initiation)	Authentication/ Identification	Authorization	Session Management	Cryptography (data in rest and transit)	Error Handling	Logging/Auditing /Monitoring
Transaction	Risk	Data Classification	Security Functions Invoked						
Password Reset	HIGH	Sensitive	Debit Card, PIN, Account#	Challenge/ Questions Risk Interdicted	Pre-Auth/Bank Customer	Pre-auth SessionID/ Cookie	HTTPS	Custom Errors & Messages	Application, Fraud Detection
Username Recovery	HIGH	Sensitive	Debit Card, PIN, Account#	Challenge/ Questions Risk Interdicted	Pre-Auth/Bank Customer	Pre-auth SessionID/ Cookie	HTTPS	Custom Errors & Messages	Application, Fraud Detection
Registration	MEDIUM	Confidential PII & Sensitive	Debit Card, PIN, Account#, PII (e.g. SSN), Demographics	OOB/ Confirmation	Visitor	Pre-auth SessionID/ Cookie	HTTPS	Custom Errors & Messages	Application
Logon	HIGH	Confidential PII & Sensitive	Username /Password	Single Auth + Challenge/ Questions Risk Interdicted	Post-Auth/Bank Customer	Post-auth SessionID Mgmt	HTTPS/ 3DES Token	Custom Errors & Messages	Application, Fraud Detection
Wires	HIGH	Confidential PII & Sensitive	Amount, Account#, IBAN/BIC	Single Auth + C/Q Risk Interdicted + OTP	Post-Auth/Bank Customer	Post-auth SessionID Mgmt	HTTPS	Custom Errors & Messages	Application, Fraud Detection
Bill Pay	HIGH	Confidential PII & Sensitive	Amount, Payee Account#	Single Auth + C/Q Risk Interdicted + OTP	Post-Auth/Bank Customer	Post-auth SessionID Mgmt	HTTPS	Custom Errors & Messages	Application, Fraud Detection

STAGE IV

Threat Analysis:

“Identifying and extracting threat information from sources of intelligence to learn about threat-attack scenarios used by web focused attack agents“

Stage 4 Walkthru – Threat Intelligence/ Analysis



Threat Intelligence is Golden

- **Threat Enumeration Based upon Good Intel**
 - Threats based upon known intel
 - Prior assessment info (where applicable & useful)
 - Other application assessments from 3rd parties
 - SIEM feeds/ Syslog data/ Application Logs/ WAF logs
 - Denote attacks but will reveal overarching threats
 - Threat Intel/ Feeds
 - Security Operations/ Incident Reports
 - Personnel/ Infrastructure
- **Threat examples:**
 - IP Theft
 - Data Theft
 - Sabotage
 - Infrastructure compromise
 - Ransom

Threat Analysis Prefaces Attack Enumeration

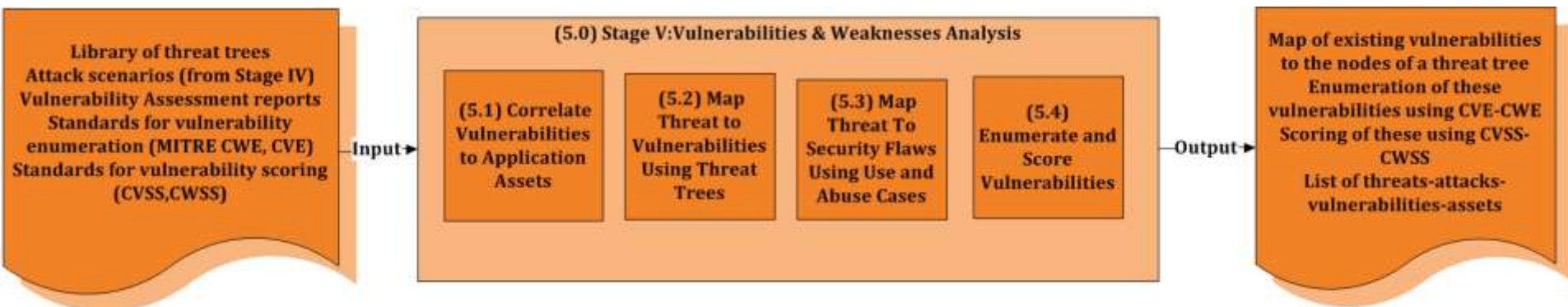
- Threat analysis will lead to attack enumeration
 - PII theft
 - XSS
 - SQL Injection
 - MITM
 - Sabotage driven threats
 - CMS exploits to web application (Zope, Joomla, Mambo, etc)
 - FTP Brute Force attacks
 - iFrame Injection attacks
 - Malware upload
- Identify most likely attack vectors
 - Address entire application footprint (email, client app, etc)
 - Web Forms/ Fields
 - WSDLs/ SWF Objects
 - Compiled Libraries/ Named Pipes

STAGE V

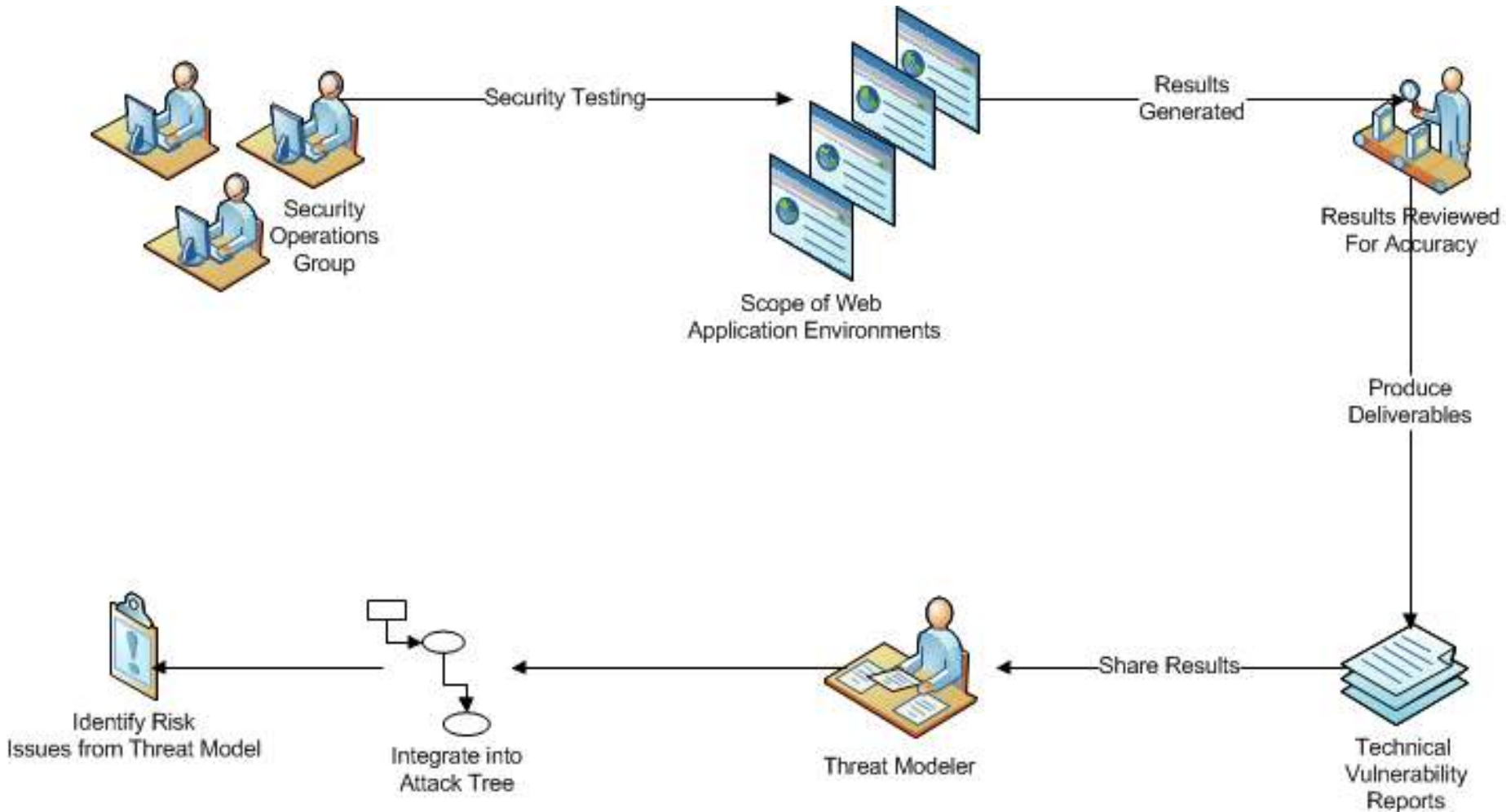
Weakness and Vulnerabilities Analysis:

Analyzing the weaknesses and vulnerabilities of web application security controls

Stage 5 Walkthru – Vuln Analysis



Leveraging Web App Testing



MITRE CWE Cross-Section: 20 of the Usual Suspects

- Absolute Path Traversal (CWE-36)
- Cross-site scripting (XSS) (CWE-79)
- Cross-Site Request Forgery (CSRF) (CWE-352)
- CRLF Injection (CWE-93)
- Error Message Information Leaks (CWE-209)
- Format string vulnerability (CWE-134)
- Hard-Coded Password (CWE-259)
- Insecure Default Permissions (CWE-276)
- Integer overflow (wrap or wraparound) (CWE-190)
- OS Command Injection (shell metacharacters) (CWE-78)
- PHP File Inclusion (CWE-98)
- Plaintext password Storage (CWE-256)
- Race condition (CWE-362)
- Relative Path Traversal (CWE-23)
- SQL injection (CWE-89)
- Unbounded Transfer ('classic buffer overflow') (CWE-120)
- UNIX symbolic link (symlink) following (CWE-61)
- Untrusted Search Path (CWE-426)
- Weak Encryption (CWE-326)
- Web Parameter Tampering (CWE-472)



Vulnerability/ Weakness Classification

WASC Threat Classification v2	OWASP Top Ten 2010 RC1
WASC-19 SQL Injection	A1 - Injection
WASC-23 XML Injection	
WASC-28 Null Byte Injection	
WASC-29 LDAP Injection	
WASC-30 Mail Command Injection	
WASC-31 OS Commanding	
WASC-39 XPath Injection	
WASC-46 XQuery Injection	
WASC-08 Cross-Site Scripting	A2 -Cross Site Scripting (XSS)
WASC-01 Insufficient Authentication	A3 - Broken Authentication and Session Management
WASC-48 Credential/Session Prediction	
WASC-37 Session Fixation	
WASC-47 Insufficient Session Expiration	
WASC-01 Insufficient Authentication	A4 - Insecure Direct Object References
WASC-02 Insufficient Authorization	
WASC-33 Path Traversal	
WASC-09 Cross-site Request Forgery	A5 - Cross-Site Request Forgery
WASC-14 Server Misconfiguration	A6 - Security Misconfiguration
WASC-15 Application Misconfiguration	
WASC-02 Insufficient Authorization	A7 - Failure to Restrict URL Access
WASC-16 Denial of Service	
WASC-11 Brute Force	
WASC-21 Insufficient Anti-automation	
WASC-54 Predictable Resource Location	
WASC-38 URL Redirector Abuse	A8 - Unvalidated Redirects and Forwards
WASC-50 Insufficient Data Protection	A9 - Insecure Cryptographic Storage
WASC-04 Insufficient Transport Layer Protection	A10 -Insufficient Transport Layer Protection

OWASP Top Ten 2010 RC1	2010 Top 25
A1 - Injection	CWE-89 (SQL injection), CWE-78 (OS Command injection)
A2 - Cross Site Scripting (XSS)	CWE-79 (Cross-site scripting)
A3 - Broken Authentication and Session Management	CWE-306, CWE-307, CWE-796
A4 - Insecure Direct Object References	CWE-285
A5 - Cross Site Request Forgery (CSRF)	CWE-352
A6 - Security Misconfiguration	No direct mappings; CWE-209 is frequently the result of misconfiguration.
A7 - Failure to Restrict URL Access	CWE-285
A8 - Unvalidated Redirects and Forwards	CWE-601
A9 - Insecure Cryptographic Storage	CWE-327, CWE-311
A10 - Insufficient Transport Layer Protection	CWE-311



MITRE CWE Cross-Section: 22 More Suspects

•Design-Related

- High Algorithmic Complexity (CWE-407)
- Origin Validation Error (CWE-346)
- Small Space of Random Values (CWE-334)
- Timing Discrepancy Information Leak (CWE-208)
- Unprotected Windows Messaging Channel ('Shatter') (CWE-422)
- Inherently Dangerous Functions, e.g. gets (CWE-242)
- Logic/Time Bomb (CWE-511)

•Low-level coding

- Assigning instead of comparing (CWE-481)
- Double Free (CWE-415)
- Null Dereference (CWE-476)
- Unchecked array indexing (CWE-129)
- Unchecked Return Value (CWE-252)
- Path Equivalence - trailing dot - 'file.txt.' (CWE-42)

•Newer languages/frameworks

- Deserialization of untrusted data (CWE-502)
- Information leak through class cloning (CWE-498)
- .NET Misconfiguration: Impersonation (CWE-520)
- Passing mutable objects to an untrusted method (CWE-375)

•Security feature failures

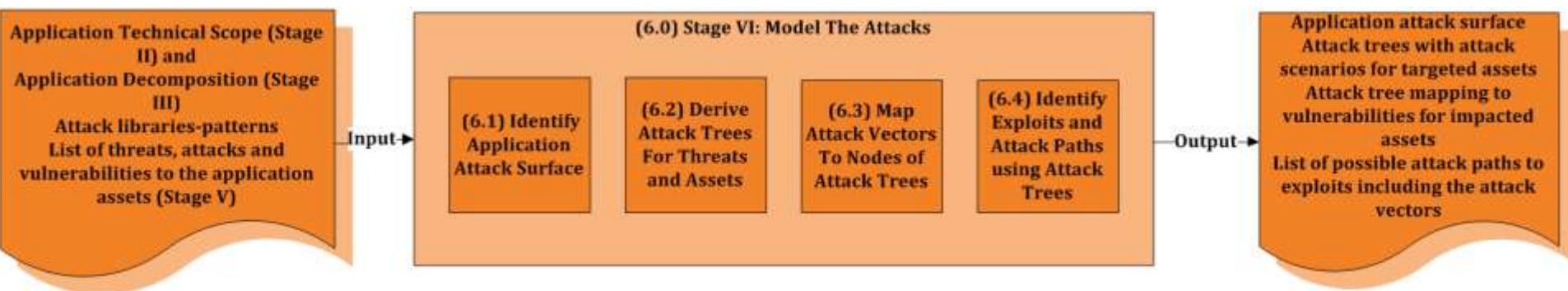
- Failure to check for certificate revocation (CWE-299)
- Improperly Implemented Security Check for Standard (CWE-358)
- Failure to check whether privileges were dropped successfully (CWE-273)
- Incomplete Blacklist (CWE-184)
- Use of hard-coded cryptographic key (CWE-321)



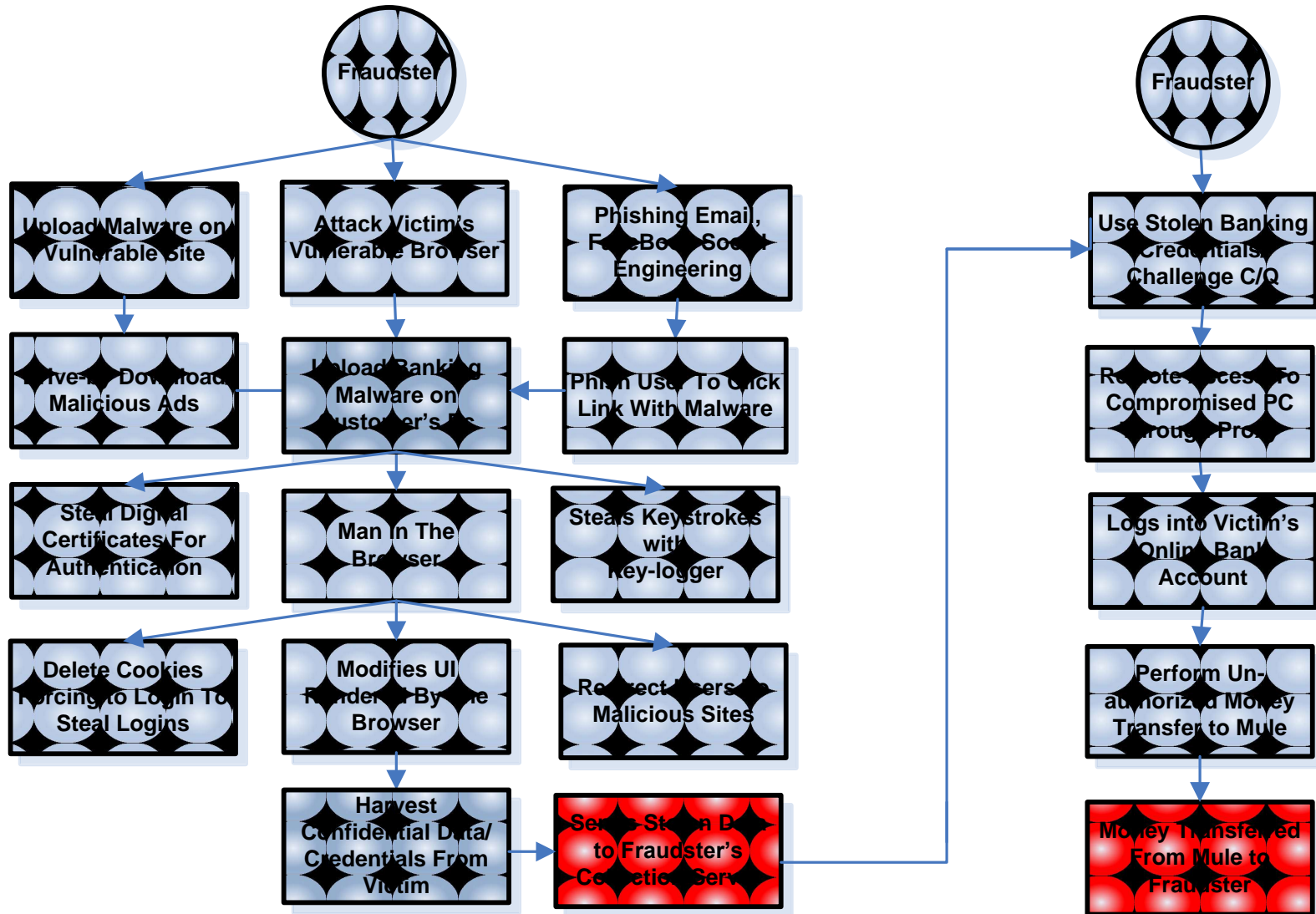
... and about 550 more

STAGE VI
Attacks/Exploits Enumeration & Modeling

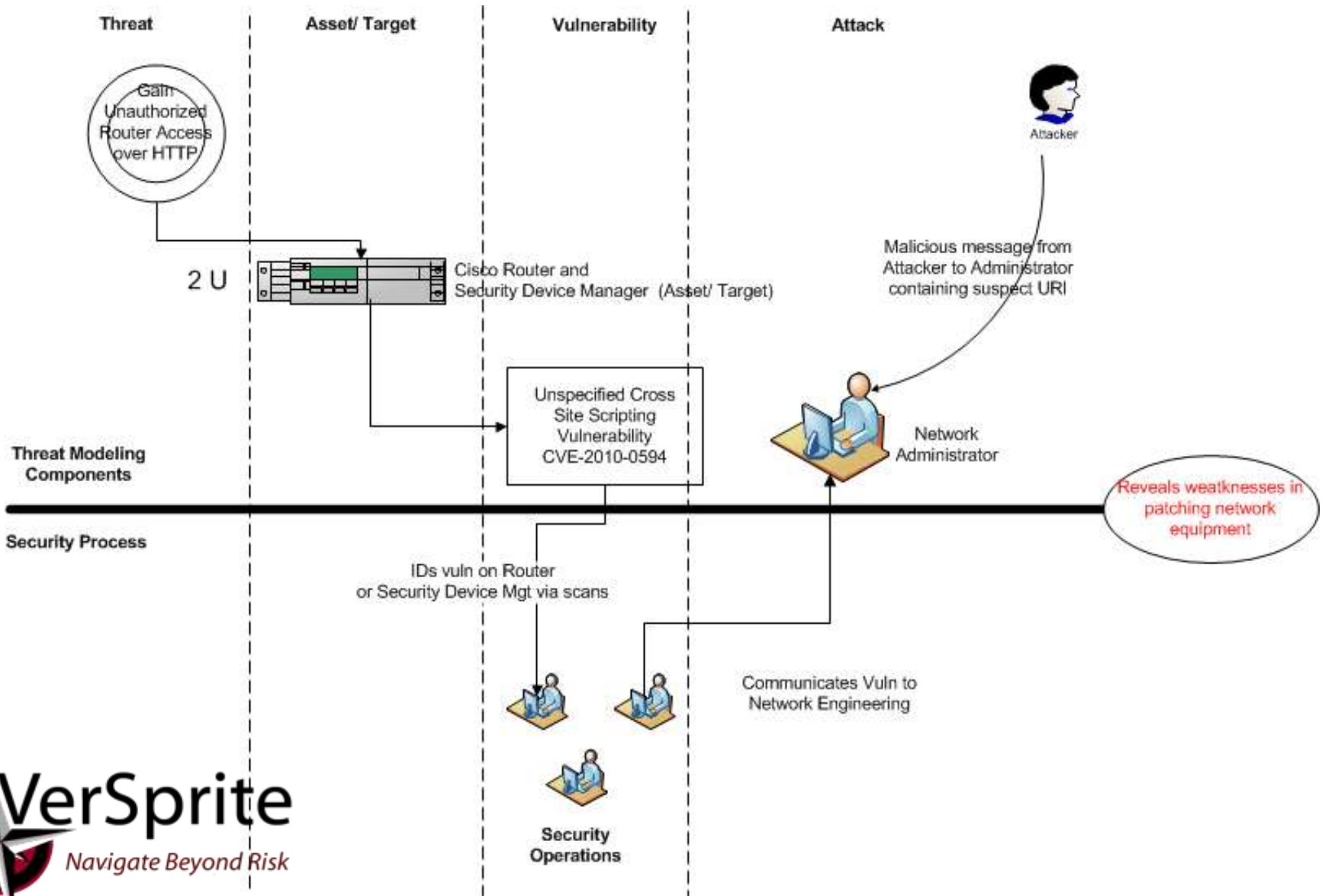
Stage 6 Walkthru – Attack Enumeration



Analysis Of Attacks Using Attack Trees



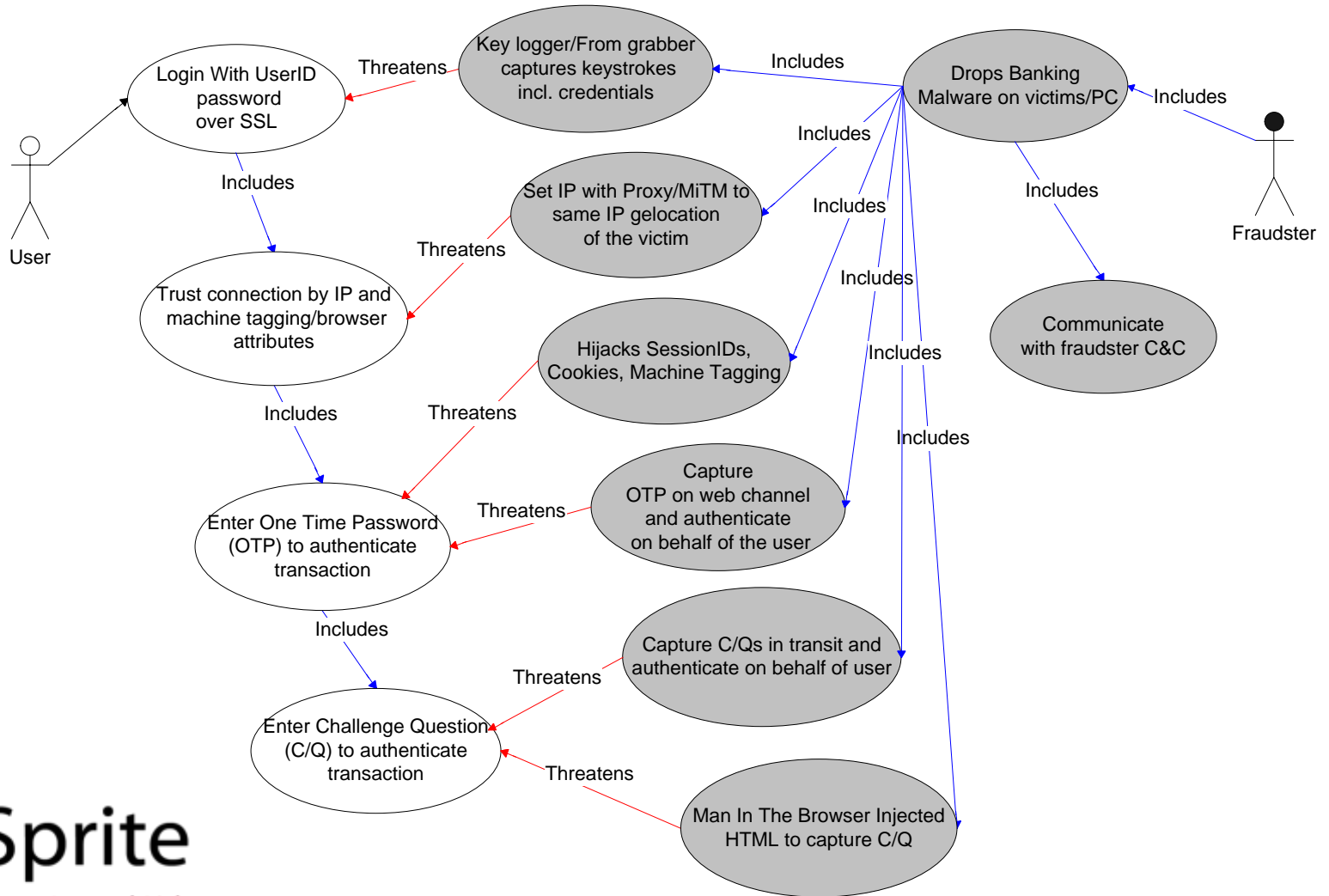
Attack Model



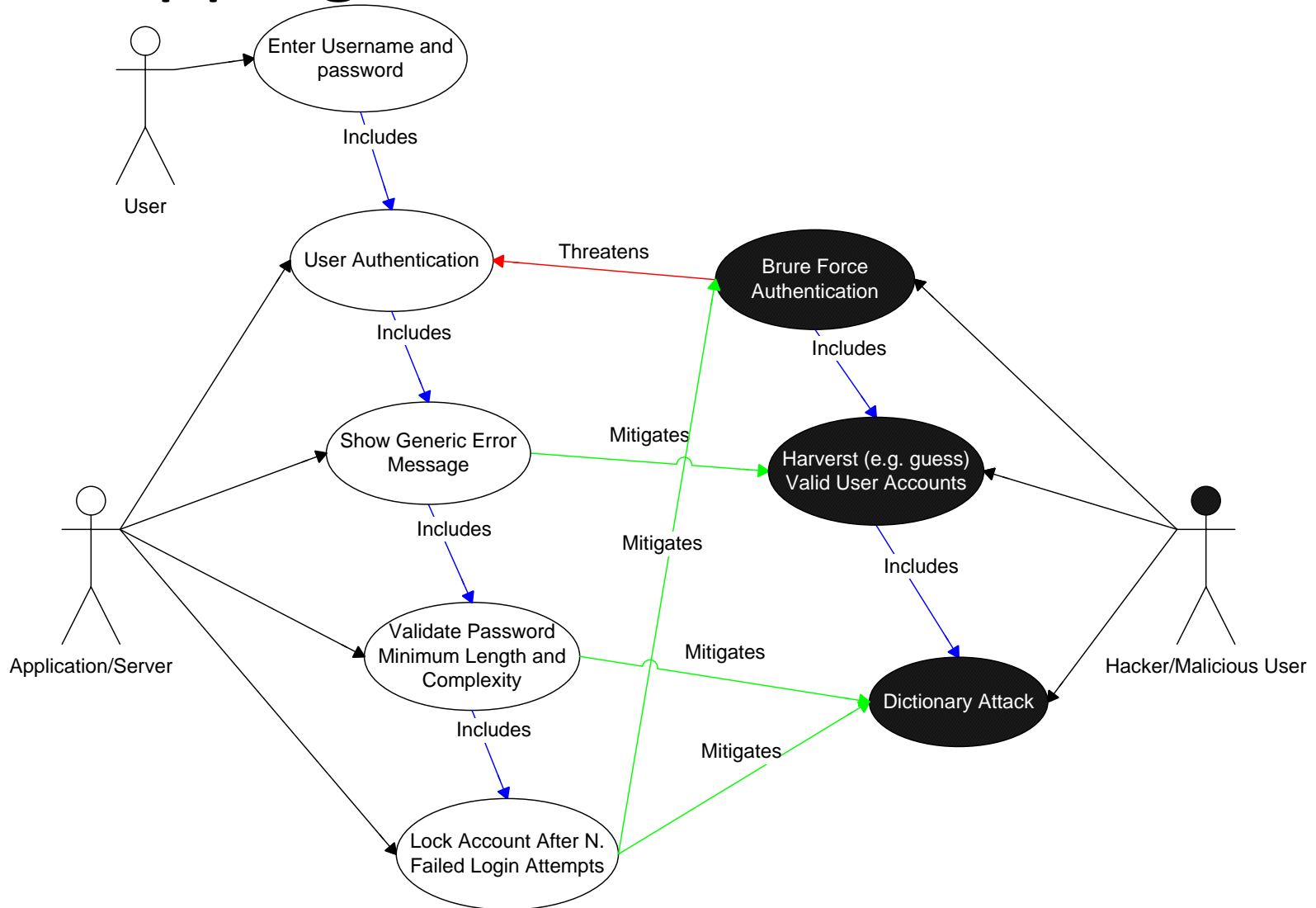
Attack Vectors Used By Different Types of Malware

Trojan	Infection Method					Attack Capabilities										Timing		Type	
	Phishing	Drive-by Download	Malicious Web Link	Malicious Ad	Virus Infection	HTTP Injection	Browser Redirect	Form Grabbing	Credential Theft	Keystroke Logging	By Pass MFA	Screen Capture/Video	Certificate Theft	Install Backdoor	Instant Message	Real-Time	Out of Band	Automated	Man-in-
MB- MitB MM-MitM B-Both O-Other						MB	MM	B	B	B	B	O	O	O	O				
Zeus	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
SpyEye	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	
InfoStealer	*	*	*	*	*	*		*	*	*	*	*	*			*	*		
Silent Banker	*	*	*	*	*	*	*	*		*	*	*	*			*	*		
URLZone	*	*	*	*	*	*		*		*	*	*	*			*	*		
Clampi/Bugat/Gozi	*	*	*	*	*	*				*						*	*		
Haxdoor	*	*	*	*	*	*		*		*			*			*	*		
Limbo	*	*	*	*	*	*		*		*	*		*			*	*		

Analysis of Web App Use and Abuse Cases



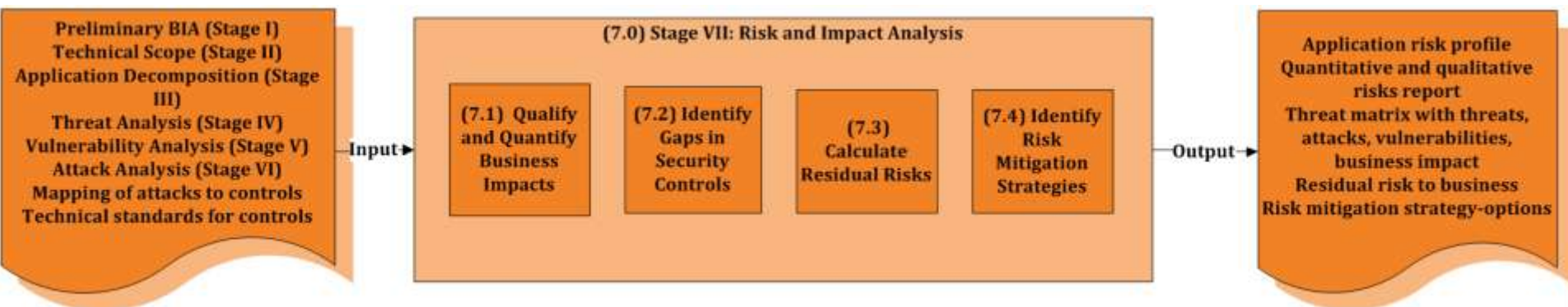
Mapping Use Cases to Misuse Cases



STAGE VII

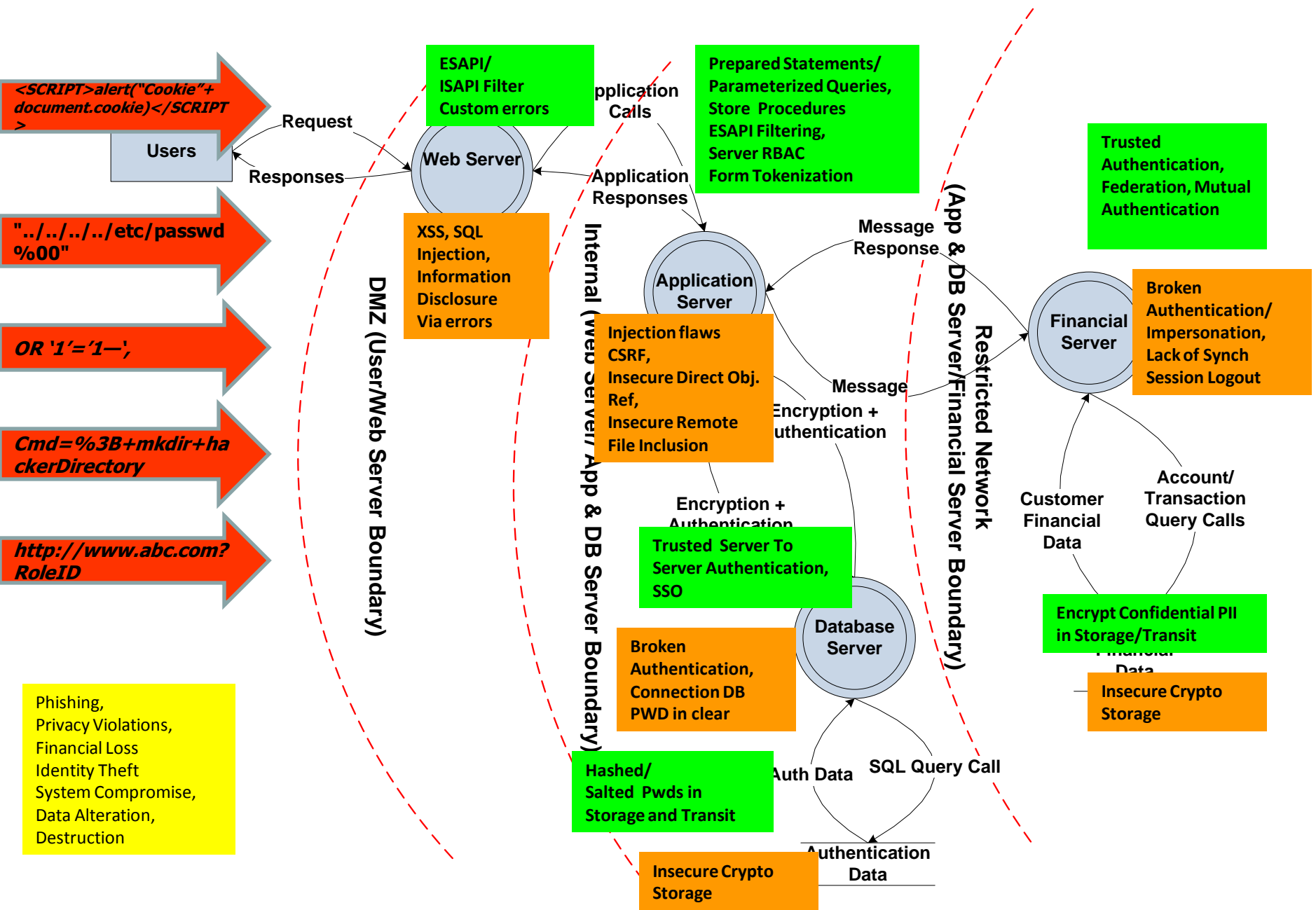
Risk And Impact Analysis: Impact Analysis, Residual Risk, and Countermeasure Development

Stage 7 Walkthru – Residual Risk Analysis



Exploits beget countermeasures

- Unacceptable risks give way to countermeasure development
- Develop countermeasures based upon the net risk of an application environment at multiple levels
 - Baseline configuration
 - Design and programmatic controls
 - 3rd party software/ COTS



Categories - into which the problem types are divided for diagnostic and resolution purposes.	Problem Types - (i.e., basic cases) involving security-related vulnerabilities.	Description	Consequences - of exploited vulnerabilities for basic security services. Do not fail in these basic security services: Authentication (resource access control), Confidentiality (of data or other resources), Authenticity (identity establishment & integrity), Availability (of services), Accountability , & Non-repudiation .	SDLC Phase - Exposure Period	Exposure Period - (i.e., SDLC phases) in which vulnerabilities can be inadvertently introduced into application source code.	SDLC Phase - Avoidance & Mitigation	Avoidance & Mitigation - (i.e., SDLC phases) in which preventative measures and countermeasures can be applied.	Platforms - which may be affected by a vulnerability.	Required Resources prerequisites for exploiting attack vulnerabilities in application's source code.
Range & Type	Buffer Overflow	A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers.	<ul style="list-style-type: none"> Availability: Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop. Access control (instruction processing): Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. Other: When the consequence is arbitrary code execution, this can often be used to subvert any other security service. 	Requirements	<ul style="list-style-type: none"> Requirements specification: The choice could be made to use a language that is not susceptible to these issues. 	Requirements	<ul style="list-style-type: none"> Pre-design: Use a language or compiler that performs automatic bounds checking. 	<ul style="list-style-type: none"> Languages: C, C++, Fortran, Assembly Operating platforms: All, although partial preventative measures may be deployed, depending on environment. 	Any
Range & Type	Buffer Overflow	A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers.	<ul style="list-style-type: none"> Availability: Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop. Access control (instruction processing): Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. Other: When the consequence is arbitrary code execution, this can often be used to subvert any other security service. 	Design	<ul style="list-style-type: none"> Design: Mitigating technologies such as safe-string libraries and container abstractions could be introduced. 	Design	<ul style="list-style-type: none"> Design: Use an abstraction library to abstract away risky APIs. Not a complete solution. 	<ul style="list-style-type: none"> Languages: C, C++, Fortran, Assembly Operating platforms: All, although partial preventative measures may be deployed, depending on environment. 	Any
Range & Type	Buffer Overflow	A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers.	<ul style="list-style-type: none"> Availability: Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop. Access control (instruction processing): Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. Other: When the consequence is arbitrary code execution, this can often be used to subvert any other security service. 	Implementation	<ul style="list-style-type: none"> Implementation: Many logic errors can lead to this condition. It can be exacerbated by lack of or misuse of mitigating technologies. 	Requirements	<ul style="list-style-type: none"> Pre-design through Build: Compile-based canary mechanisms such as StackGuard, ProPolice and the Microsoft Visual Studio /GS flag. Unless this provides automatic bounds checking, it is not a complete solution. 	<ul style="list-style-type: none"> Languages: C, C++, Fortran, Assembly Operating platforms: All, although partial preventative measures may be deployed, depending on environment. 	Any
Range & Type	Buffer Overflow	A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers.	<ul style="list-style-type: none"> Availability: Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop. Access control (instruction processing): Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. Other: When the consequence is arbitrary code execution, this can often be used to subvert any other security service. 	Implementation	<ul style="list-style-type: none"> Implementation: Many logic errors can lead to this condition. It can be exacerbated by lack of or misuse of mitigating technologies. 	Operational	<ul style="list-style-type: none"> Operational: Use OS-level preventative functionality. Not a complete solution. 	<ul style="list-style-type: none"> Languages: C, C++, Fortran, Assembly Operating platforms: All, although partial preventative measures may be deployed, depending on environment. 	Any
Range & Type	"Write-what-where" condition	Any condition where the attacker has the ability to write an arbitrary value to an arbitrary location, often as the result of a buffer overflow.	<ul style="list-style-type: none"> Access control (memory and instruction processing): Clearly, write-what-where conditions can be used to write data to areas of memory outside the scope of a policy. Also, they almost invariably can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. Availability: Many memory accesses can lead to program termination, such as when writing to addresses that are invalid for the current process. Other: When the consequence is arbitrary code execution, this can often be used to subvert any other security service. 	Requirements	<ul style="list-style-type: none"> Requirements: At this stage, one could specify an environment that abstracts memory access, instead of providing a single, flat address space. 	Requirements	<ul style="list-style-type: none"> Pre-design: Use a language that provides appropriate memory abstractions. 	<ul style="list-style-type: none"> Languages: C, C++, Fortran, Assembly Operating platforms: All, although partial preventative measures may be deployed depending on environment. 	Any
Range & Type	"Write-what-where" condition	Any condition where the attacker has the ability to write an arbitrary value to an arbitrary location, often as the result of a buffer overflow.	<ul style="list-style-type: none"> Access control (memory and instruction processing): Clearly, write-what-where conditions can be used to write data to areas of memory outside the scope of a policy. Also, they almost invariably can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. Availability: Many memory accesses can lead to program termination, such as when writing to addresses that are invalid for the current process. Other: When the consequence is arbitrary code execution, this can often be used to subvert any other security service. 	Design	<ul style="list-style-type: none"> Design: Many write-what-where problems are buffer overflows, and mitigating technologies for this subset of problems can be chosen at this time. 	Design	<ul style="list-style-type: none"> Design: Integrate technologies that try to prevent the consequences of this problem. 	<ul style="list-style-type: none"> Languages: C, C++, Fortran, Assembly Operating platforms: All, although partial preventative measures may be deployed depending on environment. 	Any

The PASTA™ Recipe For Risk Eval of Web Apps

- Focus on **the application** as business-asset target
- Risk $\neq t * v * i$
- Risk $= t * v * i * p$
- Attack simulation enhances (p) probability coefficients
- Considers both inherent countermeasures & those to be developed
- Focused on minimizing risks to applications and associated impacts to business

$$\blacksquare R_{\text{risk}} = [(t_p * v_p) / c] * i$$

Bonus Round

Tools Along the Way

File Add Edit View Scale Shift Tools Analysis Results Window Help

New Model 1 - SDL Threat Modeling Tool v3.1.4

File Edit Actions Help

Analyze Model

- All Elements
 - Commands (User to My Process)
 - Configuration (My Process to Data)
 - Responses (My Process to User)
 - Results (Data to My Process)
 - Data
 - User
 - My Process

ID	Element Name	Element Type	Element Diagram References	Threat Type	Bug ID	Completion
3	Commands (User to My Process)	DataFlow	Context	Tampering		
4	Commands (User to My Process)	DataFlow	Context	InformationDisclosure		
5	Commands (User to My Process)	DataFlow	Context	DenialOfService		
9	Configuration (My Process to Data)	DataFlow	Context	Tampering		
10	Configuration (My Process to Data)	DataFlow	Context	InformationDisclosure		
11	Configuration (My Process to Data)	DataFlow	Context	DenialOfService		
6	Responses (My Process to User)	DataFlow	Context	Tampering		
7	Responses (My Process to User)	DataFlow	Context	InformationDisclosure		
8	Responses (My Process to User)	DataFlow	Context	DenialOfService		
12	Results (Data to My Process)	DataFlow	Context	Tampering		
13	Results (Data to My Process)	DataFlow	Context	InformationDisclosure		
14	Results (Data to My Process)	DataFlow	Context	DenialOfService		
21	Data	DataStore	Context	Tampering		
22	Data	DataStore	Context	Repudiation		
23	Data	DataStore	Context	InformationDisclosure		
24	Data	DataStore	Context	DenialOfService		
1	User	Interactor	Context	Spoofing		
2	User	Interactor	Context	Repudiation		
15	My Process	Process	Context	Spoofing		
16	My Process	Process	Context	Tampering		
17	My Process	Process	Context	Repudiation		
18	My Process	Process	Context	InformationDisclosure		
19	My Process	Process	Context	DenialOfService		
20	My Process	Process	Context	ElevationOfPrivilege		

1 Draw Diagrams
 2 Analyze Model
 3 Describe Environment
 4 Generate Reports

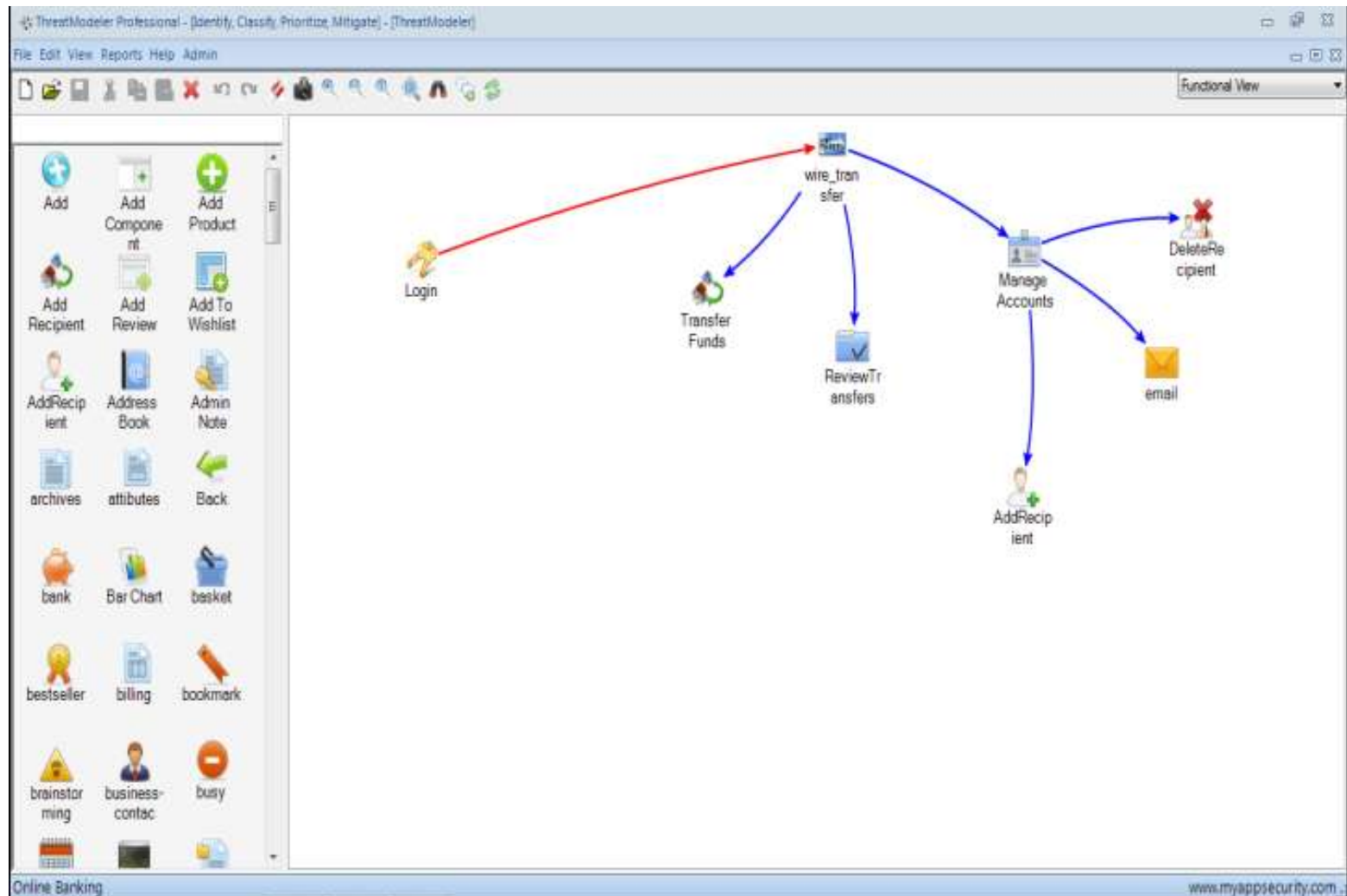
Quick Help

$P=0.1$
 $COST=2e+3$
 $EQUIPMENT=1$

A Use Case is an ordered sequence of actions used to fulfill a subset of the allowable permissions that are defined in your data access control matrix.

Unregistered Users → login → Website → Verifies the cred. → Web Service (Non Admin) → verify credentials → Database

Threat Analysis Using Threat Modeling Tool



Threat Modeling Web Apps via SDLC

- Asset based threat model is able to address inherent and new risks that should be mitigated based upon baseline of info.
 - Pen Tests, Risk Assessments, Compliance Audits, etc
 - Business Risk Mitigation Key
- Software based threat models will build upon understood threats to software environ
 - Comparable web apps, prior static/ dynamic analysis, and other web app assessments
 - Safeguarding software integrity is key and fosters building security in
- Security centric threat model focused on security of web application environment
 - More focused on attack identification and applying countermeasures
- PMs, business analysts, business owners devise functional requirements (Definition Phase)
- Architects and IT Leaders speak to architectural design and platform solutions (Design Phase)
- Governance leaders inject compliance & standards requirements for during the design phase; BIA
- Threat Model* (SOC/ NOC fed), DFDs Introduced, Trust Boundaries defined, Countermeasures proposed

^
T
h
r
e
a
t

M
o
d
e
l

v

Define

- Biz Objectives
- The C Word

Design

- Security Arch
- Security Frameworks
- AntiSamy (Java, .NET)
- OWASP ModSecurity

Develop

- OWASP Top 10
- OWASP Development Guide
- ESAPI
- OWASP Dev Guide/ OWASP .NET Project

Test (QA)

- ASVS (3rd Party Dev)
- OWASP Testing Guide (Internal)

The Beneficiaries of PASTA™

- **Business managers** can incorporate which security requirements that impact business
- **Architects** understand security/design flaws and how countermeasure protect data assets
- **Developers** understand how software is vulnerable and exposed
- **Testers** can use abuse cases to security tests of the application
- **Project managers** can manage security defects more efficiently
- **CISOs** can make informed risk management decisions





QUESTIONS

ANSWERS

[https://www.surveymonkey.com/s/
Research12_TonyUVRS](https://www.surveymonkey.com/s/Research12_TonyUVRS)

