# Enhancing Semantic Rule Engine for Semantic based Web Application Firewall (SWAF)

**By**

**Rana Faisal Munir**

**2008-NUST-MS PhD-IT-25**

**Supervisor**

**Dr. Khalid Latif**

A thesis submitted in partial fulfillment of the requirements for the degree of

Masters of Science in Information Technology (MSIT)

In

**NUST School of Electrical Engineering and Computer Science (SEECS),**

**National University of Sciences and Technology (NUST), Islamabad, Pakistan**

NUST School of Electrical Engineering and Computer Sciences

*A center of excellence for quality education and research*

# CERTIFICATE

Certified that the Scrutinizing Committee has reviewed the final documentation of Mr. Rana Faisal Munir Reg. no. 2008-NUST-MS PhD-IT-25 student of MS-IT-9 thesis title Enhancing Semantic Rule Engine for Semantic based Web Application Firewall (SWAF) and found satisfactory as per NUST's standard format for Master Thesis.

President

Wg Cdr (R) Muhammad Ramzan

# APPROVAL

It is certified that the contents and form of thesis entitled **"Enhancing Semantic Rule Engine for Semantic based Web Application Firewall (SWAF)"** submitted by **Rana Faisal Munir** have been found satisfactory for the requirement of the degree.

Advisor: Dr. Khalid Latif

Signature: ______________________________

Date: ______________________________

Committee Member: Dr. Hafiz Farooq Ahmed

Signature: ______________________________

Date: ______________________________

Committee Member: Mr. Ammar Karim

Signature: ______________________________

Date: ______________________________

Committee Member: Mr. Muhammad Bilal

Signature: ______________________________

Date: ______________________________

**IN THE NAME OF ALMIGHTY ALLAH**

**THE MOST BENEFICENT AND THE MOST MERCIFUL**

**TO MY PARENTS AND SISTERS**

# CERTIFICATE OF ORIGINALITY

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author: Rana Faisal Munir

Signature: ___________________

# ACKNOWLEDGEMENTS

First and foremost, I would like to extend my humble gratitude to Almighty Allah who always bestowed His blessings on me and gave me courage to accomplish this task. Darood-o- Salaam to Prophet Muhammad (P.B.U.H) chosen by Almighty Allah to guide the mankind to divine path.

I am truly indebted to my supervisor Dr. Khalid Latif for his support, guidance and unending tolerance. He patiently spelled out all new concepts and completely guided me in all technical directions. I own that without the inspiring guidance of Dr. Khalid Latif, this research would not have materialized. I extend my appreciation to my co-supervisor Dr. Hafiz Farooq Ahmad. Whenever there was a problem he steered me through. I am thankful to all of my committee members, for their guidance throughout the research work of this thesis. I really have no words of thanks for my friends and research fellows for their co-operation. I would always cherish the moments spent with them.

I can never forget the contribution of my parents in providing their all assistance to me. I owe all my achievements to my parents whose assistance and prayers enabled me to surpass all the hurdles in my life.

**Rana Faisal Munir**

# ABSTRACT

Web applications after their revolutionary advent and popularity have become target range for variety of attacks. Magnitude and complexity of these attacks is continuously growing with every minute development in World Wide Web. There are plenty of web attack detection techniques but they cannot fully comprehend the required degree of security for complex web applications. The reasons include static nature of attack detection mechanism, lack of expressiveness in attack detection rules, and absence of reasoning capability to detect unanticipated ways through which an attack can appear. To cater these issues, a formal approach is required that has more expressiveness and equipped reasoning. We used ontology as a formal approach which provides expressiveness and reasoning as a package. We also studied the important attributes that are helpful to analyze and detect web attacks. These are root causes, HTTP portion used, messages needed for attack, impact and detection models used for detection. On the basis of our empirical study and pragmatic results, we developed web application attacks ontology. The developed ontology underwent three evolution criteria. Formal correctness and consistency is validated using OntoClean and Pellet reasoner. Domain coverage is second criteria and our ontology covers all web attacks listed by OWASP. Last but not least is the task orientation that how it will be used for detecting web attacks; we made a case study which shows how effective it is when we use it for detection.

# Table of Contents

# List of Figures

# List of Tables

CHAPTER # 01

# Introduction & Motivation

## 1.1 Introduction

Web Applications security has become gradually more significant these days. Colossal numbers of attacks are being deployed on the web application layer. Due to spectacular increase in Web applications, security gets exposed to variety of threats. Various attacks are embattled towards the web application layer; network firewall alone cannot foil these kinds of attacks. The fundamental reason behind success of these attacks is the unawareness of application developers while writing the web applications and the vulnerabilities in the existing technologies. Various technologies from various vendors for instigate same standards, e.g. Common Gateway Interface (CGI) [1] is the standard mechanism for specifying the work of dynamic web application. Different technologies like ASP [2] and ASP.NET [3], JSP [4] and PHP [5] to name a few exist for implementing the same technology in different ways and hence results in rising complexities entailing in supplementary security concerns. Figure 1 [6] shows various technologies with respect to vulnerabilities found in their implementations.

The rapid development in Web 2.0 and evolution of social networks became centric to the hackers. Starting from any scholarly artifacts to media files, articles, business advice, and consultancy; the Web has grown from static web pages to a haystack of billions of dynamic portals effecting lives of millions of users.

As the information on the internet is growing at a volcanic speed, concerns about and its security are also arising.

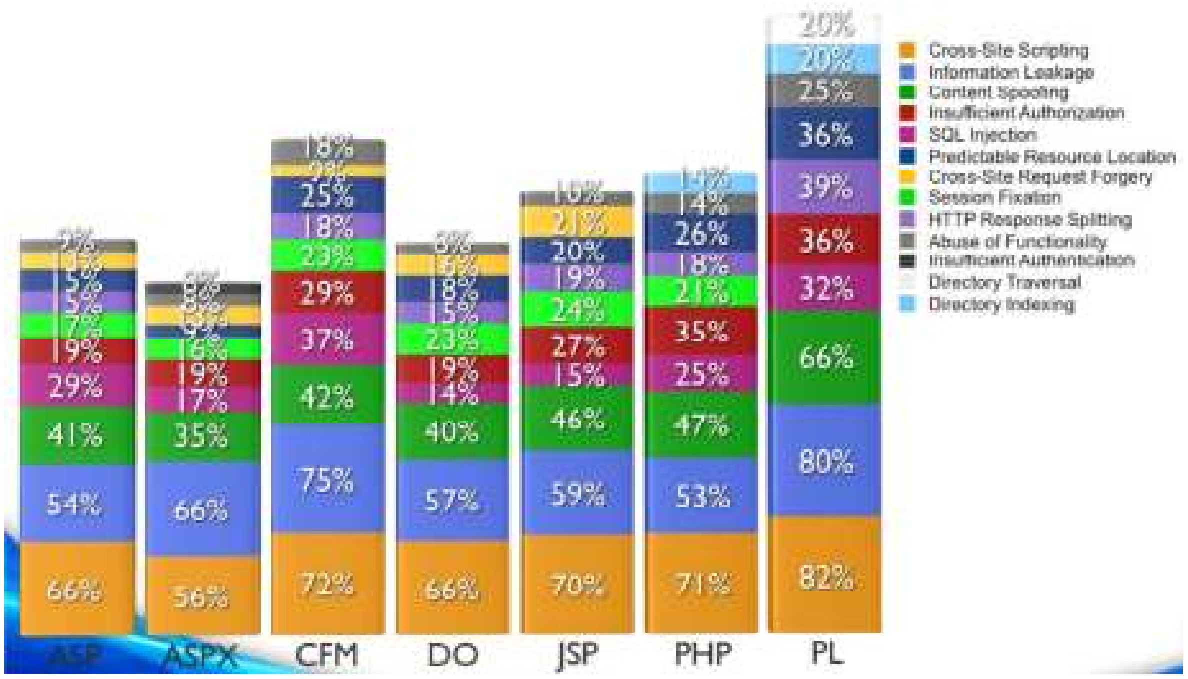

**Figure 1: Technology Breakdown with respect to attacks**

Considering above fact, web applications are the most vulnerable. 75% of attacks are being deployed on web application layer [7, 8, 9]. 81% of these attacks are targeted on payment card industry. The organizations which uses shared and default credentials give 51 % of the data to the hackers [10]. According to site security monitor that in every 90 breaches there are 285 million records exposed and that is greater the 230 million exposed records in previous 5 years [11]. 30% of each 57 attacks are carried out using SQL injection attack [11] that's why the web application security is the most important. Figure 2 [6] shows the vulnerabilities according to their impact in percentage, 80% website have critical vulnerabilities. To protect web applications from attacks, there should be clear understanding of each attacks and their main root causes. This information will help to find the way to detect and prevent these attacks. To present this knowledge, ontology [12] is the best solution for this purpose. Ontology has the features like expressiveness and reasoning capabilities. These features help to prevent the existing attacks and can help to understand and prevent the new attacks.

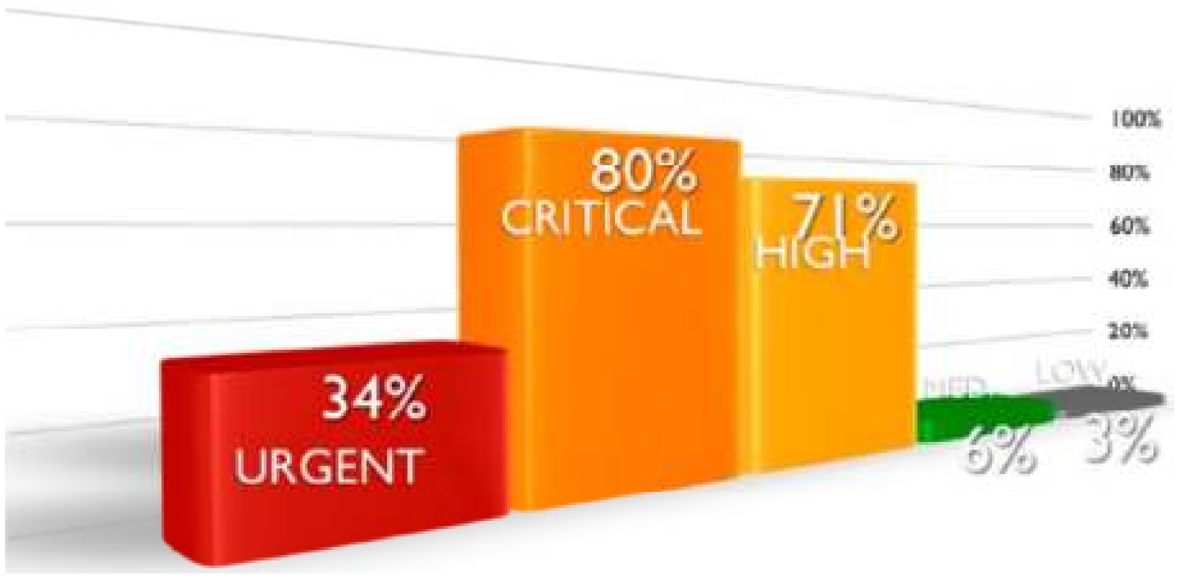

**Figure 2: Likelihood of websites having vulnerabilities by severity rating**

## 1.2. Introduction to Ontology

The idea of ontology is not new. It is a subject of discussion among many Philosophers for centuries [12]. However, ontology, today have turn out to be more formalized conceptual representation utilized in computer science, artificial intelligence, and database integration [12]. According to Gruber ontology is “the arrangement of conceptualizations, used to assist programs and humans distribute knowledge.” [12, 13] Ontology, thus, provides a basic and well defined vision of a specific area of domain. In the exacting application of knowledge base for artificial intelligence and data integration, knowledge enclosed within the ontology must be human and machine-readable in order to present greater semantic capacity of the World Wide Web for users within precise domains. Prescribed languages have been urbanized for the encoding of this ontology knowledge; Web Ontology Language (OWL) is the most well-known language now a day to describe knowledge of any domain. It’s based on concepts and relations that are working to involuntarily classify taxonomies [14]. OWL has been engaged here due to its expressiveness and durability. OWL contains classes, properties, individuals, and restrictions. Classes represent concepts in a domain. E.g. in the web application security domain, SQLi [15] would be presented as a class. Classes can be hierarchical structure where subclasses are defined.

## 1.3 Motivation

Web application security is a vast area which is expending day by day. Due to the increasing number of attacks on the web application, prevention and detection of these attacks are very difficult at the application layer. There is a need for common way of representing knowledge of web attacks which can help security community in detection and prevention of these attacks. There are various ontology designed for information security but none of them for web attacks which is in high demand. The proposed ontology meets the current demand and can be used for attack analysis, scanning web applications and most importantly detection of web attacks using web application firewall (WAF) [16]. World is becoming a cyber space and web attacks are the biggest threat for its utility and survival. Complete range of web assets are under potential threats especially the critical and secrete information. Sharing of information which is sensitive and crucial in nature is probable against potential threats. It needs utmost security infrastructure and techniques to withstand against a range of web attacks.

## 1.4 Objective

Main objective of thesis is to study web attacks to see how these attacks work on web application layer and what are their impact on an organization which have critical information on the internet. Second phase emphasizes to understand how ontology work and to study existing ontology's in the information security domain. This knowledge will be used to build web application attack ontology for detection and prevention of web attacks. Further, it can be used as a web application scanner which can identify loop holes in the web application or can be used in web application firewall to detect and prevent the attacks. This ontology covers all aspects of web application attacks to make it worthy for research community.

## 1.5 Thesis Organization

This thesis is ordered into six different chapters. **Chapter 2** provides extensive literature survey of the existing ontology in the information security domain. **Chapter 3** presents the study of all web application attacks that currently exists. **Chapter 4** presents the ontology that we developed for web application attacks. **Chapter 5** shows the evaluation of the proposed ontology using onto-clean. **Chapter 6** presents the conclusion and future work of this thesis.

CHAPTER # 02

# Existing Work & Literature Survey

This chapter provides the background knowledge for research. It gives a detailed overview of ontology that currently exists in the information security domain. Literature survey is divided into five different categories. Latest research papers are included in each category. These categories are intrusion detection system, network security, security privacy and policy, risk management, web services and malware. The all categories utilize the ontology for their domain and show the effectiveness of the ontology.

## 2.1 Intrusion Detection System

This categories show the work of ontology in the domain of intrusion detection system. These solutions have been presented in [17], [18], [19], [20], [21] and [22].

The proposed solution in [17] presents the ontology for computer network attacks that they used for distributed IDS. The authors have analyzed 4,000 classes of computer network attacks, their attributes and relationships with each other. Their main focal point of ontology is attack target. The ontology Host class stores victim of any attack. Then Attack class is part of this ontology that represent different computer network attacks, this class has properties: Directed To, Resulting In and Effected By. Each attack is directed to a System Component. System component class has two sub-classes: System and Process. These classes represent the current state of the Host. Each attack has Consequences. Examples of Consequence subclass are Denial of Service. Each attack is launched by some means of Input, and Input can be caused by some Means, Means class has two subclasses: Input Validation Error and Logic Exploit. They represent the different means of an Attack. The ontology was evaluated in two phases: In first phase it detects the anomaly and then they prepare a dataset that is inserted in ontology as instances. In second phase, they use reasoner to find the possible anomaly behavior. The inserted sample data to show how efficiently ontology based IDS detects attack. The attacks they focused were: Sync Flood attack, Mitnick type attack and buffer flow attack.

The solution presented in [18] proposed the ontology for distributed Intrusion detection System. This ontology helps the IDS to share the messages to each other that can be easily interpreted using ontology interoperability. The attack signature is the root node and has three classes that cover the different features from heterogeneous sensors. Host feature contains all information that is related to the system, like memory usage, then we have to go from host feature to system status and then we can check the memory usage. This is very useful in cooperative detection process; we can locate the required information using ontology easily. Because ontology help us which sensor contains the required information, like we know that all the sensor who has Host feature, have the information related to System call, Application log, and system status. They assign the weight to each edge between value node and its parent node. The weight is ranges from 0 to 1, where1 mean two nodes have maximum similarity and 0 mean minimum. They weights are assigned by expert and can be adjusted according the result feedback. By finding the maximum similarity between any nodes possible attack can be identified that attacker tries launch by modifying some of attack parameter. In start different rules are loaded into the ontology as instance. A total score function is used to calculate the similarity score for each audit data that different sensors observe. The paper gives example of backdoor, that their IDS detected successfully using ontology and matching algorithm. Using their methodology they are successful to detect the all the possible ways of backdoor communication.  Ontology also helps to find the sensor that have the right information that is required by another sensor. Like if we need to know the memory usage, then we should contact with the system status sensor.

The proposed solution in [19] used ontology to detect the automated scripts or tools that normal user downloaded from the internet unintentionally. These scripts then use victim PC as source of launching DDOS attacks or other attacks. The ontology presented is attacker centric that is they consider that attacker is logged in to remote system and then try to compromise any other system using that remote PC. Author introduces different concepts in ontology, he divides the network system into agents that will monitor the different activities, agent has sensor that will use to monitor the traffic and process running on the system. Reactor and reaction concept are used to perform an action after a malicious behavior is detected. Agent cells are environment in which agent can perform reliably. The agents that share common goals are placed in the same agent cell. Correlators are used to gather information and perform the analysis on it. As an input it will take two streams of data, one is outbound network traffic and second is process execution data. The network and process data stream is used by the agent cell to generate the signature locally and this is shared with others using the ontology. They generate the two types of signature, one is for network traffic and one is for process. Correlators cells continuously monitor the traffic and process streams, and identify the malicious program execution on the basis of analysis. The generation of signature is generated on the basis of some learning data that can be for example some k packet of TCP.

The proposed ontology of [20] is used in Distributed IDS. They proposed two types of agents that will work in the distributed environment. They will communicate with each other using the proposed ontology, and this ontology helps the IDS to get the relationships between different computer attacks and suspect situation in networks accurately. The ontology covers different type of computer attacks; they classify the attacks in Trojans, Network Attacks, Physical attacks, viruses, Denial of Service and password attacks. The focus of the paper is on Denial of Service attack only, that’s why they expand their ontology in the domain of DoS. They divide the DoS attack into Exhausting service and Stopping service. These attack classes further divide and then we can go into the depth of the DoS attack to capture the exact information related to the type of DoS attack. Their IDS has two agents, Master agent holds the ontology and IDSagent is a special host based or network based IDS. Whenever they find a suspicious status, they send the report to the master agent. Master agent saves the report to the ontology and then queries the ontology to find the information, is network under attack. Currently they show this technique for DoS attack. The ontology helps the Master agent to find the correct type of attack from the ontology. If master agent detects an attack, then it will send the suitable alarms to IDSagent, so that they take the correct reaction for that attack.

The ontology proposed in [21] used for detection of web application attacks. Their ontology covers the different web attacks like SQLi, XSS etc. They have the concept #Attack and from which port they received this attack. In this paper they only focused on HTTP attacks, that’s why they shows the ontology for HTTP Request only. Each HTTP request has three components, Request Line, Header and Payload. Then they further divide it according to the RFC 2616. Attacker used some attack vectors to launch the attack on the application. They call the attack vectors as malicious code in the ontology. This malicious code can be inserted into application by means of input like Query String. This paper primarily focuses on SQLi and XSS attacks. They used the concepts of web crawler to extract the information for a web application and then find entry points from which attacker can inject malicious code. They modify the Bayesian filter to detect the web application attacks. They assign different weight to different html attributes and then calculate the weight for incoming string, if weight exceeds than the threshold consider it as attack. They use the inference engine to find the possible ways from which a malicious code can be injected. This will help them to find the exact path from which attacker can insert the code and then they apply the detection techniques on that path to detect the attack.

The proposed ontology model [22] to define the events of Intrusion detection and prevention system. Primary focus of this paper is on the Distributed Intrusion detection system, Distributed IDS used the Intrusion Detection Messages Exchange Format (IDMEF). This format is based on XML, but XML doesn't provide any reasoning capabilities, it is just used to define structure. To add the reasoning capability authors used the ontology model to describe all the events their relationships and also use the Semantic Web Rule Language (SWRL) for reasoning. It covers different type of attacks, anomalous protocols, web code injection and Denial of Service attack using sync flood. They have used attack signature for detection. They have rules as subclass of Signatures; they used it to define the different prevention rules to prevent defined attacks. This ontology focuses on the network node that is targeted by an attacker. All the properties and relationships are developed according to network node. They choose the Agent Software engineering process as methodology to build their Distributed IDS. Agents perform the following functionalities: Sensor agent capture the traffic from the network, Analyzer agent analyzed the capture data with predefined signature define in ontology. Correlation agent uses the inference engine and ontology to classify the attacks accurately by the help of inference. Then Reaction agent at the end, generate alarms and inform the other network nodes.

## 2.2 Network Security

This categories show the work of ontology in the domain of network security. These solutions have been presented in [23], [24], [25], [26] and [27].

This solution presents in [23] proposed ontology for network security attacks. They review the existing threat and vulnerabilities profiles. Then identifies the core concepts that can be used in network security ontology. Actor class represents the Black hat hacker, Cracker, Malevolent user, Malevolent Systems Administrator, Script kiddie etc, that makes attack on network. Attack class represents the different network attacks; these are control of system, DoS, modification of network message content, replay, spoofing or traffic interception. They classify the attack into two categories; Active attacks and passive attacks. Impact represents the effect of attack on the network. Threat that is used to launch attacks is bacteria, worm, virus, Trojan horse or logic bomb. Motive class represents the reason of an attack. This can be for fun, gain, revenge. The Information class represents what attacker try to steal. Outcome of the attack can be interruption, interception, modification or fabrication.

The ontology presents in [24] for the Mobile Ad-Hoc Network security threats. They used the ontology to describe all MANET security threats so that everyone can easily understand its threat and can find the solution to prevent these threats. System class is the victim of threat and it has the following subclasses: Network, System components and Processes. Actor launches the threat on the system by using some input. This input can be launched by using some attacks like wormhole, dynamic topology or selfish node etc. We can check the system is under threat by checking the following fault condition; that are input validation error, logic error or other errors. Consequences are also divided into three types; loss of assets, illegal access and other general outcomes like modification etc. Attacker can be any System problem, human or other. System problems are classify into Software Defect, System crash etc, Human can be inside or outside. For any attack, actors have some motivation that causes them to launch attack on target system. This motive can be for fun or revenge.

Voice of IP service based on Session Initiation Protocol has gained popularity in last few years. With the popularity there are my serious threats appeared in SIP protocol that should be addressed. To address these threats in-depth understanding is required. The proposed solution in [25] its ontology that helps to understand each threat. The ontology considers the attack related to the protocol that has different consequences from DoS to gain access. They have two core concepts in this ontology; that are SIP_message: It shows the structure of SIP message that attacker can use to launch attacks, and SIP_attack: It represents the all SIP attacks that can be malformed or flood. Root Malformed attack is the incorrect sip message sent, and flooding can be launched by sending many sip messages simultaneously. SIP message consists of first line and header. Every sip message should contain two things otherwise it will be considered as invalid. First_Line class used to differentiate between request and response. First_Line for request also contains different methods that SIP used to sending request; these are Register, Invite etc. They used URI to write the rule that is based on the SIP grammar that can help to validate the SIP request. Like, for SIP register request we have a register URI rule that is used to validity of incoming request. Header is consisting of two things; one is header name and second is rule that is used to check its validity. They focused on two attacks, one is malformed attack, this attack is caused by sending the malformed messages that is not according to the grammar or inconsistent, it can validate by using ontology. Second attack is flooding attack for this they proposed to use the threshold for single user to detect it. Target class represent the SIP component on which attack is launched, it has IP and port as properties. Consequence class show the impact of attack on target, it can be DoS or unauthorized access.

The strategy proposed in [26] focuses on the reaction that is takes after an attack detected in network. The important thing in Reaction after Detection is how effectively we apply the accurate policy to solve the attack related issues. They propose the ontology that helps to find the policies that can be applied in the network to solve the threats. They used ontological approach that is based on the OrBAC security model. OrBAC is stands for opens Role Based Access Control. This model helps to define the security policies for an Organization.  This model has a set of contextual security rules corresponding to permissions, prohibitions and obligations. The security rule applies when their associated context is activated. They also used attack ontology to provide the formal description of network attacks. The primary focus of this paper mapping from attack alerts to the threat contexts, and these threat contexts then helps to identify the security policies that are used for reaction strategy. Subject, object and action classes represent the concepts that are used in OrBAC model. These classes are mapped with Role, View and Activity. This ontology has the following classes, Organization, Hold, Context and Rule. Organization is the core class of the OrBAC model; Hold class will have Subject, object and action. Context is based on CVE for classification of alert. OrBAC model allows defining the rules to apply the security policies in any organization.  These rules will be used as reaction policies. The ontology has the relationship of Hold class with other classes. It contains the subject, object, action, organization and also a context. Ontology also presents Rule class and its properties that it used to attach with other concepts.

Ontology for computer and networks attack presented in [27], they primarily focused on network based Denial of Service attack. To develop this ontology they studied different number of log and connections that cause the network DoS. The ontology covers different type of computer attacks; they classify attacks in Trojans, Network Attacks, Physical attacks, viruses, Denial of Service and password attacks. The focused of the paper is on Denial of Service attack only, that’s why they expand their ontology in the domain of DoS. They divide DoS attack into Exhausting service and Stopping service. These attack classes further divide and then we can go into the depth of the DoS attack to capture the exact information related to the type of DoS attack. This ontology is focused on these types of DoS attacks; these are SYN flood, smurf, teardrop, ping of death etc.

## 2.3 Security Privacy and Policy

This categories show the work of ontology in the domain of security privacy and policy. These solutions have been presented in [28], [29], [30] and [31].

The paper [28] presents the security ontology that focuses on annotating the functional aspect of the resources. Annotation with security related metadata help to identify the resources that fulfill the security requirements. Security information includes the mechanisms, protocols, objectives algorithm and credentials in different levels of details and specificity. This paper claims that their security ontology helps to describe the security concepts at different level of detail. The paper also represents this ontology because previous ontology annotates web services rather than resources, but their ontology focuses on how to annotate resources in a web service. There ontology is an aggregation of seven other ontology's. These are Main Security Ontology, Credentials Ontology, Security Algorithms Ontology, Security Assurance ontology, Service Security ontology, Agent Security Ontology and Information Object Ontology. The main security ontology describes the security concepts, credentials ontology to specify the authentication credentials, security algorithm ontology to describe various security algorithm, security assurance ontology to specify different assurance standards, service security ontology to facilitate security annotation of semantic web services, agent security ontology to enable querying the security information and information object ontology to describe the security of input and output parameters of web services. The below picture shows the main security ontology,SecurityConcept is the top class and has three subclasses, these are Security Protocol, SecurityMechanism and SecurityPolicy. SecurityPolicy and SecurityMechanism are implementation of protocols to accomplish a task. And SecurityPolicy defines the set of rules to protect and secure information. This ontology also hasSecurityObjective that user specify for a web service. After annotation to all resources of web services, they apply the matching algorithm to verify that security policies should apply accurately. They have to match two things; one is service provider and other service requestor. It will check the provider's security requirements should be satisfied by the requestor's security capabilities. The ontology and matching algorithm help to a web service finder to find the services that has the desired functionalities and also the security capability and requirements that client want.

The paper [29] proposed an ontological solution for policy specification, administration and formalization. This solution helps to define policy for information sharing between two parties. These policies will enforce the security specification that administrator wants to apply when information is shared. The purpose of formal specification is to help in the context of trust negotiation that is the first phase when two parties that wants to share information. The information related to trust are credentials attributes that are needed. The root represents any ID, this is divided into classes, and these are Government and Enterprise. Enterprise ID class represents the credential that an organization issued; these can be IBM, Cisco etc. And Government ID can be issued by state etc. They define the policy specification in F-Logic, policy has the name, value and type attribute. Policy is divided into two classes that is express using type; these can be Default or Mandatory. They defined the metaClass that help to define the policies in any class. This class has two attributes, policySlot and overallPolicy. PolicySlot has the set of policies and overallPolicy whose value is the set of policy of all policies. Mandatory policy helps to enforce the higher level policies at lower levels. Default policies are those policies that every subclass should be fulfilled. The paper makes a PolicyTab as a protypefor protégé. This tab helps to define policy and can be attached to any class that is defined in ontology.

The paper [30] proposed an ontological solution that helps the organization to apply the IT security policy in easy way. This solution helps to implement the COBIT and ISO 17799 standards in any organization. The security ontology has five sub-ontology; these are Attribute, Threat, Infrastructure, Role and Person. The Attribute ontology helps to model the impact of threats i.e. which threats influence the certain security attributes and on the basis of this organization can prioritize the IT security strategy. These attributes can be Availability, Confidentiality, Integrity, Maintainability, Reliability and Safety. This ontology has the relation with other ontology, like Attribute, Infrastructure etc. The Infrastructure ontology has the information related to building, rooms, electronics devices, networks etc. The Role ontology is used to define the roles of each person in any organization, and person ontology help to define the persons that are responsible to apply the security policies.

The paper [31] presents ontology for information security domain. This ontology covers all the concepts that are required for information security domain. This ontology has the following core concepts, assets, threats, vulnerabilities, countermeasures and their relationships. These concepts are borrowed from the risk analysis ontology. Asset is connected with vulnerability, and also with threat and threat is attach with the security goals that are target of the threat. A countermeasure is used to protect the asset from the threat. The core concept countermeasures is very well defined in this ontology, they cover all possible countermeasures that exist in the security domain. Some examples are encryption, secure network communication, access control etc. The Assets class is also defined in expressiveness manner to cover all possible asset exist in any organization. The general asset in any organization is Credential, Technology, Human and

Countermeasure. These assets should be protected from the threats. The Threats or attacks are also very well defined, it cover all possible threats that can threaten the asset of any organization. Threat is divided into two categories, these are active or passive. Active threat have direct attack with asset, these can be brute force, denial of service, disruption etc. And passive threats are statistical attack, eavesdropping etc. The vulnerability covers thirteen vulnerabilities; this can be buffer overflow, malformed input etc. The countermeasures are divided into memory protection or source code analysis. And these concepts then further divided to cover all tools in these domains and techniques.

## 2.4 Risk Management

This categories show the work of ontology in the domain of risk management. These solutions have been presented in [32], [33] and [34].

The paper [32] purposes a system named SemanticLife which is a personal Information management system which gather user interaction events and correlates them using ontology. This paper present a risk assessment method using SemanticLife tool which will help in security planning and decision making. SemanticLife stores, manages and retrieves the lifetime's information entities of individuals. SemanticLife ontology is devised into three parts (1) user environment ontology, (2) project ontology and (3) attack ontology. User environment captures the information of environment e.g. operating system on node, software installed on node etc. Project ontology describes the classification of project-related entities such as tasks, project plans, assignments & allocations, resources, and costs. Attack ontology provides classification of attacks like active and passive attacks with pre and post conditions of attack. SemanticLife has three main plug-ins, these are Message Bus, web service and pipeline. Message Bus Plug-in, this plug-in manage all information that is exchanged between different processes running inside

SemanticLife. Web Service Plug-in, this plug-in uniform all resources as services and expose them inter or extern user based on semantic policies. It has two types, External web services, Internal Web services. Pipeline plug-in, it plays central role, basically it perform the intermediate transformations between different web services calls. Semantic life handles the information using policies. Policies are stored in the form of RDF store. For risk assessment it captures the data form user and correlates it with other events to establish a user profile of single users. Combining this data with risk ontology, useful results can be generated. For example, from the risk ontology we know that a specific attack can happen only when specific preconditions are met. Some typical preconditions are, OS version, open ports, etc.

The paper [33] presents an ontological mapping of the ISO/IEC 27001 standard, IT security EBK and its control countermeasure. The above ontology has four concepts these are, Control, Category and Objective. Control class defines the security perimeters that are used to protect areas that contain information and information processing facilities. Category defines the types of physical and environmental security. Actual security goals are to protect the information from the unauthorized access. Objectives to prevent unauthorized physical access, damage and interference to the organization's premises and information. These concepts help how we will manage our assets according to security policies, how we design the procedure that ensure the security policy, and then in implement concept we want to implemented the designed procedures and evaluate concepts contains all information to evaluate the effectiveness of our implemented procedure in any organization for risk assessment. This paper also presents the security incident ontology, this ontology has the following classes, these are Access, Agent, Asset, it is divided into four Organization Functions, Information System, Information, and Environment assets, Attack class, Consequences, Security Incident, Time, Vulnerability, Threat, types of threats can be Natural disasters, Industrial Origin, Errors and Unintentional Failures, remaining classes are Origin and Countermeasure.

The paper [34] represents the ontology based information security risk assessment structure for Ontology-based Unified Problem-Solving Method Description Language (UPML) approach. The paper proposed three ontology, these are Domain ontology, Task ontology, and Resolution ontology. This paper seeks to construct a knowledge model that represents a framework which related goals to the control tasks of information security management by analyzing the current accepted information security management standards and practices BS7799. First phase of the risk assessment is the Establishment of "Domain" Ontology Knowledge Base as shown in below picture. In this phase we have to take out the actual value of the organizations asset. It contains all the tangible and intangible assets of the organization. Second step is Establishment of "Task" Ontology Knowledge Base which contains impact and analyses of the threats and determines the risk factor based on these factors. These factors are risk of the asset under a certain threat, possibility of the threat, possibility of the leak being used and potential threat impact. Third and the final phase is establishment of "Resolution Ontology" Knowledge Base. This phase has two steps, first step is to define the ontology, and second phase is by using the "Propose & Revise" Method to Improve Information Security Risks. This phase help us to analyze and prioritize planned risk reassessment/readjustment tasks based on importance, and assign resource requirements to each suggestion.

## 2.5 Web Services

This categories show the work of ontology in the domain of risk management. These solutions have been presented in [35] and [36].

The use of web services is increasing day by day. To satisfy the user need web services are becoming more complex. Sometime to satisfy the user request, one web services are not enough then we need web services composition. These web services are heterogeneous in nature that is the obstacle between compositions. To solve this heterogeneity and context reconciliation, according to this paper [35], context is the interaction between humans, applications and surrounding environment. They also proposed the OWL-C, a language to specify the web services context, this language is based on the OWL-S, language for web services. Context is the core concept in this ontology, a context type is associated with a specific type of service namely Web service, Web service instance, or composite service. Web service to Web service instance indicates that a Web service has one or more Web service instances. Web service instance to composite service indicates that a Web service instance belongs to one composite service. Web service to constraint and Web service instance to constraint illustrate respectively the constraints on a Web service (e.g. maximum number of Web service instances that can be created). Context type indicates that a context has exactly one type namely I, W and C. I-Context corresponds to the identifier of the service instance, W-Context corresponds to the identifier of the web service and C-Context corresponds to the identifier of the composite service. Each context type has a set of sensors, purpose, description, name, and a set of arguments. An argument has a name, data type, description, and synonyms. This paper also focuses on the security of web services, they categorize the web service threats into three category: impersonate attack, content-borne threats and operational threats. They proposed the security context for web services instance, web service and composite web service. Security context focuses on the strategy of securing the interactions of services during data-context exchange. At the Web-service instance level, the primary use of I-context is to track its execution status. If a service instance was subject to threats that attempted altering its context, this should be reported in its respective ISec-context so that corrective actions are planned for the forthcoming service instances. At the Web-service provider level, whenever a Web service receives a request of participation in a composition, it validates its current capabilities using C-context and checks its security requirements. If both are satisfactory, the Web service creates a new service instance. At the composite-service level, the C-context traces execution of the composite service and its respective component service instances, and tries to identify potential heterogeneities (in terms of resources, shared variables, shared log files) between these service instances. The CSec-context ensures that the essential security property of non-repudiation of messages sent and received by the composite service during its interactions with Web service providers and Web service instances.

The paper [36] focuses on web services security and their attacks. They proposed the use of distributed IDS that can be used to detect multi-phase web services attacks. They main problem in distributed IDS is communication with each other's. These IDS can be from different vendors and don't have same vocabulary for communication. They proposed the use of ontology that provide them common vocabulary and this will solve this issue. For this purpose they proposed the web services attack ontology. The discovery attacks have two sub classes, these are WS Probing Attacks and UDDI attacks. Probing attacks is further divided into WSDL scanning and Parameter Tampering. WSDL is use for web services specification; it has all the function and their parameters. This is the main target to collect the information related to a web service. Discovery attacks are classified in Probing attacks, this attack can be on WSDL scanning, Parameter tampering. UDDI attacks are used for probing attacks.

## 2.6 Malware

This categories show the work of ontology in the domain malware. These solutions have been presented in [37].

The paper [37] proposed a system that used to analyze the malware behavior based on the ontology. They proposed domain ontology for Malware. They divide the ontology in different layers, the first layer is the domain name of the ontology, second layer is category layer that defines the different category of malwares, third layer is concept layer define different concepts or classes in different categories, and the last layer is known as instance layer, this layer contains data for different classes of malware. The above ontology use Malware_Type that has the Worm, Backdoor, Trojan etc as sub classes. This will represent the different malware types. Malwar_Impact_Target class has the File, Registry and Network as its subclasses. Malware_Impact_Target class represents the targeted component of the system that will effect when this malware will executed on the victim system. Malware_Behaviorial, represents the behavior of malware, how it work and affect the system. They divide their system into three layers: that are knowledge layer, communication layer and application layer. Knowledge layer include the knowledge base, rules and the ontology. Communication layer is help for interaction between application layer and knowledge layer. To analyze the behavior of malware, they monitor the following things; network traffic and system file changes. These systems analyze the malware in the windows environment. This system populates ontology with the sample malware, and waits for some time and monitors the system to track what malware done. They have used SWRL rule for behavior analysis, this help in inference and infer the new knowledge that help in better malware analysis and can help to detect correct behavior.

## 2.7 Summary

The above literature survey presents all existing ontology solutions. They mostly present the ontology and solutions for network security or web services. All solutions mostly ignore the web application attacks. That is the most famous now a days. There should be ontology for web application attacks that can be used by research community. They above literature survey help us to find the dimension that is useful in the design of ontology, but these solutions are inadequate and commend the need to carry out this research and come up with a solution to overcome these shortcomings.

CHAPTER # 03

# Web Application Attacks

This chapter discusses all web application attacks that are listed by Open Web Application Security Group (OWASP) [8].

## 3.1 Attacks Attributes or Dimensions

We identify the different dimensions that are helpful for ontology design. These dimensions can help to analyze the attack behavior and help to find the way to detect and block it. The below list shows the dimensions that we finalized after our literature survey, these are

- Attack Root Cause
- HTTP Portion
- Attack Behavior
- Detection Model
- Attack Impact

### 3.1.1 Attack Root Cause

This dimension helps us to find the exact reason of any attacks. This is very helpful for analysis or can assist in detection of that attack. The root cause is the weakness in our system, if we fix our system weakness, its mean attacks on the system will be fixed. Figure 3 shows the hierarchy of root cause that can be reason for any attacks.

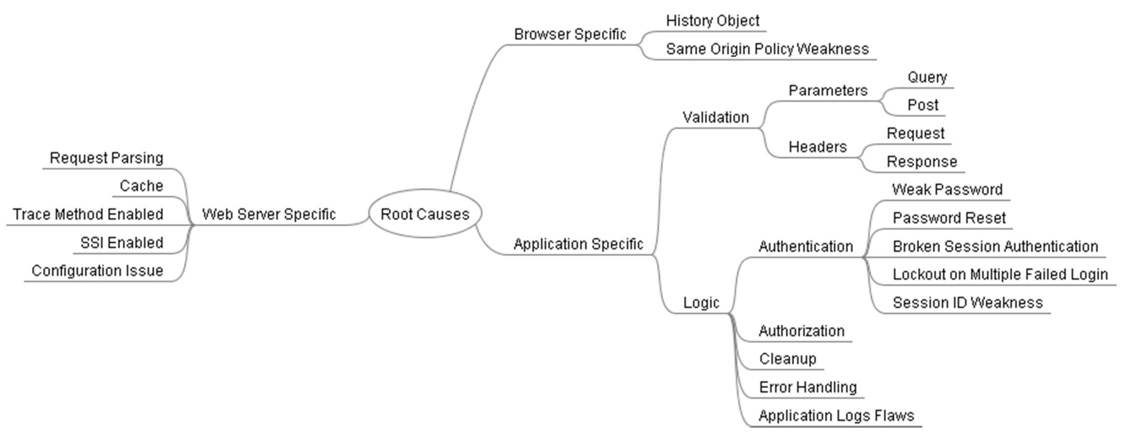

**Figure 3: Root Cause Hierarchy**

### 3.1.2 HTTP Portion

This dimension is useful to locate the exact location of the attack in HTTP message. HTTP is the main protocol used in web application for sending requests. Attackers used some portion of HTTP message that is used as carrier. Figure 4 shows the complete HTTP message.

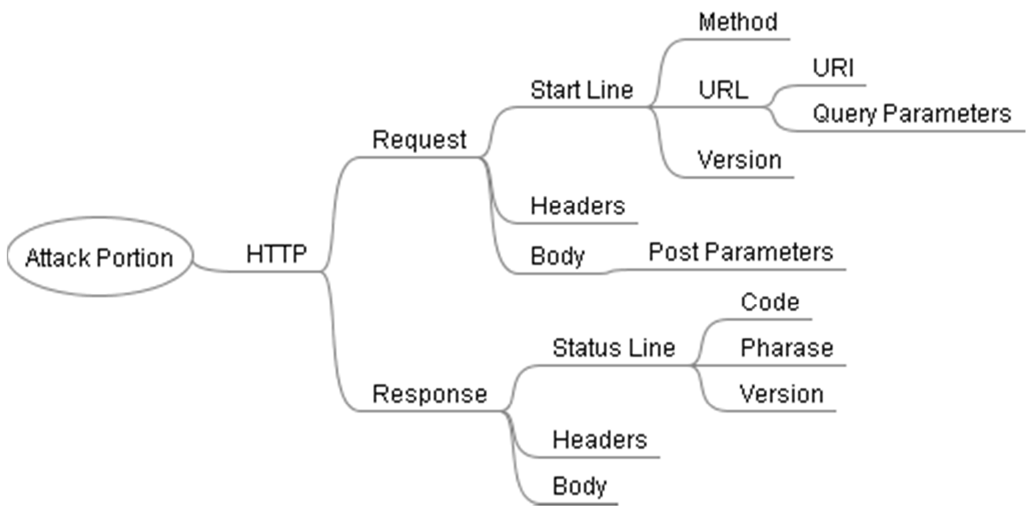

**Figure 4: HTTP Portion**

### 3.1.3 Attack Behavior

This dimension help to analyze that the attack can be send using one HTTP message or need multiple messages to launch attack. Figure 5 shows the attack behavior in graphical form

Multiple Request
Attack Behavior
Single Request

**Figure 5: Attack Behavior**

### 3.1.4 Detection Model

In firewall, we have three models to detect an attack. We also model this in our ontology, so that if ontology is used inside firewall then we will get the model that will be used for attack prevention. Figure 6 shows the detection model diagrams

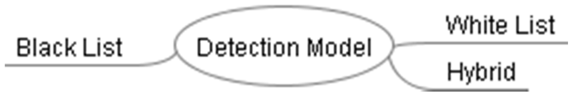

Figure 6: Detection Model

### 3.1.5 Attack Impact

This dimension help to analyze the damage that an attack can made on the system. We take help from the DREAD [38] model proposed by Microsoft for damage assessment. And also we study the consequences written in RFC 2828 [39]. Figure 7 shows the attack impact hierarchy

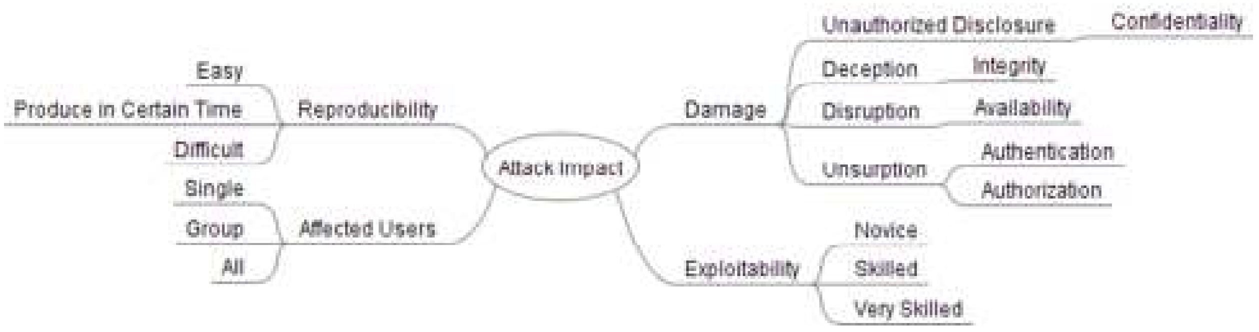

Figure 7: Attack Impact

## 3. 2 Web Application Attacks Details

This section contains the details of each web application attacks according to the above defined attributes. This will help us to design our ontology in next chapter.

Table 1: Account Lockout Attack

| Providing wrong password, till account block message is triggered by application [40][41] | |
|---|---|
| Root Cause | Lockout on Multiple Fail Login |
| Attack Behavior | Multiple Requests |
| Attack Portion | HTTP Query or Post Parameters |

| | |
|---|---|
| Detection Model | White List |
| Attack Impact | Denial of Service attack |

**Table 2: Argument injection**

| Injecting new parameters to HTTP request to trigger non – accessible logic directly [42][43] | | |
|---|---|---|
| Root Cause | NIL | |
| Attack Behavior | Multiple Requests | |
| Attack Portion | HTTP Query or Post Parameters | |
| Detection Model | White List | |
| Attack Impact | Damage | Authentication or Authorization |
| | Exploitability | Novice |
| | Reproducibility | Easy |
| | Affected Users | Single User |

**Table 3: Asymmetric resource consumption (amplification)**

| In this attack, attacker send requests to a resource that failed to clean the consumed resources, this make the system out of resource. [44][45] | |
|---|---|
| Root Cause | Cleanup |
| Attack Behavior | Multiple HTTP Requests |
| Attack Portion | NIL |
| Detection Model | Hybrid |
| Attack Impact | Denial of Service attack |

**Table 4: Blind SQL Injection**

| Injection of SQL queries without having knowledge of the application database. [46][47] |
|---|

| | | |
|---|---|---|
| Root Cause | Parameters and Request Headers Validation | |
| Attack Behavior | Single HTTP Request | |
| Attack Portion | HTTP Headers, Query and Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Group of Users |

**Table 5: Blind XPath Injection**

| Injecting XPATH query without having knowledge of the XML data-store schema. [48][49] | | |
|---|---|---|
| Root Cause | Parameters and Request Headers Validation | |
| Attack Behavior | Single HTTP Request | |
| Attack Portion | HTTP Headers, Query and Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Group of Users |

**Table 6: Brute force attack**

| Common ways of guessing value of a particular field, i.e. password, forget password answer or anything that is application specific. [50][51] | |
|---|---|
| Root Cause | NIL |

| | | |
|---|---|---|
| Attack Behavior | Multiple HTTP Requests | |
| Attack Portion | HTTP Query and Post Parameters | |
| Detection Model | Hybrid | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 7: Buffer overflow attack**

| | | |
|---|---|---|
| Sending more data to application / program data buffers that it can handle to cause stack overflow [52][53] | | |
| Root Cause | Parameters and Request Headers Validation | |
| Attack Behavior | Single HTTP Request | |
| Attack Portion | HTTP Headers, Query and Post Parameters | |
| Detection Model | White List | |
| Attack Impact | Damage | Integrity, Availability |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | All |

**Table 8: Cache Poisoning**

| | |
|---|---|
| Injecting malicious content in Web Server or browser cache. [54][55][56] | |
| Root Cause | Parameters, Request Headers Validation, Cache and Request Parsing |

| | | |
|---|---|---|
| Attack Behavior | Single HTTP Request | |
| Attack Portion | HTTP Headers and Query Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Integrity |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Group of Users |

**Table 9: Code Injection [Local or Remote File Inclusion]**

| Injecting code that is interpreted by application. [57][58] | | |
|---|---|---|
| Root Cause | Parameters Validation | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query and Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Easy |
| | Affected Users | All |

**Table 10: Command injection**

| Injection & execution of commands in vulnerable applications. [59][60] | |
|---|---|
| Root Cause | Parameters Validation |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | HTTP Query and Post Parameters |

| | | |
|---|---|---|
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality, Availability |
| | Exploitability | Very Skilled |
| | Reproducibility | Easy |
| | Affected Users | All |

**Table 11: Comment Injection**

| | | |
|---|---|---|
| Comments injected into an application through input can be used to compromise a system. These comments can be of SQLi, XPATH etc. [61] | | |
| Root Cause | Parameters and Request Headers | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Headers, Query and Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Authentication |
| | Exploitability | Novice |
| | Reproducibility | Easy |
| | Affected Users | Single |

**Table 12: Cookie Tampering**

| | |
|---|---|
| Cookie header can be modified to gain the privileges or by passed the authentication system [62] | |
| Root Cause | Request Headers |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | HTTP Cookie Header |
| Detection Model | White List |

| | | |
|---|---|---|
| Attack Impact | Damage | Authentication, Authorization |
| | Exploitability | Skilled |
| | Reproducibility | Easy |
| | Affected Users | Single |

**Table 13: Cross Frame Scripting**

| It is used to describe an XSS attack which uses an HTML frame in the attack. For example, an attacker might exploit a Cross Site Scripting Flaw to inject a frame into a third-party web page; the third party web page then steals the data. [63][64] | | |
|---|---|---|
| Root Cause | Parameters Validation and Same Origin Policy Weakness | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query or Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Skilled |
| | Reproducibility | Easy |
| | Affected Users | Group of Users |

**Table 14: Cross Site History Manipulation**

| Cross-Site History Manipulation breach is based on the fact that client-side browser history object is not properly partitioned on a per-site basis. [65] | |
|---|---|
| Root Cause | Parameters Validation and Browser History Object weakness |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | HTTP Query or Post Parameters |
| Detection Model | Black List |

|  |  |  |
|---|---|---|
| Attack Impact | Damage | Confidentiality |
|  | Exploitability | Skilled |
|  | Reproducibility | Easy |
|  | Affected Users | Group of Users |

**Table 15: Cross Site Tracing**

| An XST (Cross-Site Tracing) attack involves the use of XSS and the HTTP TRACE function. The client sends an HTTP TRACE with all header information including cookies, and the server simply responds with that same data. [66] |  |  |
|---|---|---|
| Root Cause | Parameters Validation and Trace Method Enabled |  |
| Attack Behavior | Single HTTP Requests |  |
| Attack Portion | HTTP Query or Post Parameters |  |
| Detection Model | Hybrid |  |
| Attack Impact | Damage | Confidentiality |
|  | Exploitability | Skilled |
|  | Reproducibility | Difficult |
|  | Affected Users | Group of Users |

**Table 16: Cross-Site Request Forgery (CSRF)**

| CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. [67][68][69] |  |
|---|---|
| Root Cause | Parameters Validation |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | Query or Post Parameters |

| | | |
|---|---|---|
| Detection Model | Hybrid | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Group of Users |

**Table 17: Cross-User Defacement**

| | | |
|---|---|---|
| An attacker can make a single request to a vulnerable server that will cause the sever to create two responses, the second of which may be misinterpreted as a response to a different request, possibly one made by another user sharing the same TCP connection with the sever. [70][71] | | |
| Root Cause | Request Parsing | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query and Request Headers | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 18: Cross-site Scripting (XSS)**

| | |
|---|---|
| Cross-Site Scripting attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted web sites. [72][73] | |
| Root Cause | Parameters and Request Headers Validation |
| Attack Behavior | Single HTTP Requests |

| | |
|---|---|
| Attack Portion | HTTP Query, Post and Request Headers |
| Detection Model | Black List. |
| Attack Impact | Cross Site Frame Scripting<br><br>Cross Site History Manipulation<br><br>Cross Site Tracing<br><br>Cross Site Request Forgery |

**Table 19: Custom Special Character Injection**

| | | |
|---|---|---|
| The software does not properly filter or quote special characters or reserved words that are used in a custom or proprietary language or representation that is used by the product. [74][75] | | |
| Root Cause | Parameters and Request Headers Validation | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query, Post and Request Headers | |
| Detection Model | Black List. | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Novice |
| | Reproducibility | Easy |
| | Affected Users | Group of Users |

**Table 20: Denial of Service**

| | |
|---|---|
| The Denial of Service (DoS) attack is focused on making unavailable a resource (site, application, server) for the purpose it was designed. [76][27] | |
| Root Cause | NIL |
| Attack Behavior | Multiple HTTP Requests |

| | | |
|---|---|---|
| Attack Portion | NIL | |
| Detection Model | White List | |
| Attack Impact | Damage | Availability |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | All |

**Table 21: Direct Dynamic Code Evaluation**

| | | |
|---|---|---|
| This attack consists of a script that does not properly validate user inputs in the page parameter. A remote user can supply a specially crafted URL to pass arbitrary code to an eval() statement, which results in code execution. [77][78] | | |
| Root Cause | Parameters Validation | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query or Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality, Integrity |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 22: Direct static Code Evaluation**

| | |
|---|---|
| A Direct Static Code Injection attack consists of injecting code directly onto the resource used by application while processing a user request. [79][80] | |
| Root Cause | Parameters Validation |
| Attack Behavior | Single HTTP Requests |

| | | |
|---|---|---|
| Attack Portion | HTTP Query or Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality, Integrity |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 23: Double Encoding**

| Send the request by encoding the values twice. By using double encoding it's possible to bypass security filters that only decode user input once. [81][82] | | |
|---|---|---|
| Root Cause | Parameters and Request Headers Validation | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Headers, Query or Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | All security services |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 24: Forced browsing**

| Forced browsing is an attack where the aim is to enumerate and access resources that are not referenced by the application, but are still accessible. [83][84] | |
|---|---|
| Root Cause | NIL |
| Attack Behavior | Multiple HTTP Requests |

| | | |
|---|---|---|
| Attack Portion | NIL | |
| Detection Model | White List | |
| Attack Impact | Damage | Confidentiality, Authorization |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 25: Format string Attack**

| The Format String exploit occurs when the submitted data of an input string is evaluated as a command by the application. [printf] [85][86] | | |
|---|---|---|
| Root Cause | Parameters and Request Headers Validation | |
| Attack Behavior | Single HTTP Request | |
| Attack Portion | HTTP Query, Post and Request Headers | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality, Integrity |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 26: Full Path Disclosure**

| Full Path Disclosure (FPD) vulnerabilities enable the attacker to see the path to the webroot/file. e.g.: /home/omg/htdocs/file/. [87][88] | |
|---|---|
| Root Cause | Error Handling |
| Attack Behavior | Single HTTP Request |

| | | |
|---|---|---|
| Attack Portion | NIL | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Skilled |
| | Reproducibility | Easy |
| | Affected Users | Single User |

**Table 27: HTTP Request Smuggling**

| | |
|---|---|
| HTTP Request Smuggling consists of sending a specially formatted HTTP request that will be parsed in a different way by the proxy system and by the final system, so the attacker could smuggle a request to one system without the other being aware of it. [89][56] | |
| Root Cause | Request Parsing |
| Attack Behavior | Single HTTP Request |
| Attack Portion | HTTP Query and Request headers |
| Detection Model | Black List |
| Attack Impact | Cache Poisoning |

**Table 28: HTTP Response Splitting**

| | |
|---|---|
| An attacker can make a single request to a vulnerable server that will cause the sever to create two responses, the second of which may be misinterpreted as a response to a different request, possibly one made by another user sharing the same TCP connection with the sever. [90][91] | |
| Root Cause | Request Parsing |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | HTTP Query and Request Headers |
| Detection Model | Black List. |

| | | |
|---|---|---|
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 29: LDAP injection**

| LDAP Injection is an attack used to exploit web based applications that construct LDAP statements based on user input. [92][93] | | |
|---|---|---|
| Root Cause | Parameters Validation | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query or Post Parameters | |
| Detection Model | Black List. | |
| Attack Impact | Damage | Authentication |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 30: Page Hijacking**

| An attacker can make a single request to a vulnerable server that will cause the sever to create two responses, the second of which may be misinterpreted as a response to a different request, possibly one made by another user sharing the same TCP connection with the sever. [94] | |
|---|---|
| Root Cause | Request Parsing |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | HTTP Query and Request Headers |
| Detection Model | Black List |

| | | |
|---|---|---|
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 31: Parameter Delimiter**

| This attack is based on the manipulation of parameter delimiters used by web application input vectors in order to cause unexpected behaviors. [95] | | |
|---|---|---|
| Root Cause | Parameters Validation | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query or Post Parameters | |
| Detection Model | White List | |
| Attack Impact | Damage | Integrity, Authorization |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 32: Path Manipulation**

| Allowing user input to control paths used in file system operations may enable an attacker to access or modify protected system resources. [96] | |
|---|---|
| Root Cause | Parameters Validation |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | HTTP Query or Post Parameters |
| Detection Model | Black List |

| | | |
|---|---|---|
| Attack Impact | Damage | Integrity |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 33: Path Traversal**

| A Path Traversal attack aims to access files and directories that are stored outside the web root folder. [97] | | |
|---|---|---|
| Root Cause | Configuration Issue | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query or Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 34: Regular expression Denial of Service – ReDoS**

| It is a Denial of Service attack that exploits the fact that most Regular Expression implementations may reach extreme situations that cause them to work very slowly. [98] | |
|---|---|
| Root Cause | Parameters Validation |
| Attack Behavior | Single HTTP Requests |
| Attack Portion | HTTP Query or Post Parameters |

| | |
|---|---|
| Detection Model | White List |
| Attack Impact | Denial of Service attack |

**Table 35: Repudiation Attack**

| A repudiation attack happens when an application or system does not adopt controls to properly track and log users' actions, thus permitting malicious manipulation or forging the identification of new actions. [99] | | |
|---|---|---|
| Root Cause | Application Logs Flaws | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query, Post Parameters and Request Headers | |
| Detection Model | White List | |
| Attack Impact | Damage | Integrity |
| | Exploitability | Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 36: SQL Injection**

| A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. [100][101] | | |
|---|---|---|
| Root Cause | Parameters and Request Headers Validation | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query, Post Parameters and Request Headers | |
| Detection Model | Black List | |
| Attack Impact | Damage | Authentication, Confidentiality |

| | | |
|---|---|---|
| | Exploitability | Novice |
| | Reproducibility | Easy |
| | Affected Users | Single |

**Table 37: Server-Side Includes (SSI) Injection**

| SSIs are directives present on Web applications used to feed an HTML page with dynamic contents. They are similar to CGIs. [102] | | |
|---|---|---|
| Root Cause | • Parameters Validation<br>• SSI Enabled | |
| Attack Behavior | Single HTTP Requests | |
| Attack Portion | HTTP Query and Post Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Very Skilled |
| | Reproducibility | Difficult |
| | Affected Users | Single User |

**Table 38: Session Prediction**

| The session prediction attack focuses on predicting session ID values that permit an attacker to bypass the authentication schema of an application. [103] | | |
|---|---|---|
| Root Cause | Session ID Weakness | |
| Attack Behavior | Multiple HTTP Requests | |
| Attack Portion | HTTP Cookie Header | |
| Detection Model | Hybrid | |
| Attack Impact | Damage | Authentication |

| | | |
|---|---|---|
| | Exploitability | Very Skilled |
| | Reproducibility | Produce in certain time |
| | Affected Users | Single User |

**Table 39: Session Fixation**

| When authenticating a user, it doesn't assign a new session ID, making it possible to use an existent session ID. The existent session ID can be fixed by an attacker that can be used for unauthorized operations. [104][105] | | |
|---|---|---|
| Root Cause | • Session ID Weakness<br>• Parameters Validation | |
| Attack Behavior | Single HTTP Request | |
| Attack Portion | HTTP Query and Post Parameters | |
| Detection Model | Hybrid | |
| Attack Impact | Damage | Authorization |
| | Exploitability | Very Skilled |
| | Reproducibility | Produce in certain time |
| | Affected Users | Single User |

**Table 40: Session Hijacking**

| Session ID is just a string stored in Cookie Header, an attacker can steal this by XSS or by sniffing the traffic. [106][107] | |
|---|---|
| Root Cause | • Session ID Weakness<br>• Parameters Validation |
| Attack Behavior | Single HTTP Request |
| Attack Portion | HTTP Query and Post Parameters |

| | | |
|---|---|---|
| Detection Model | Hybrid | |
| Attack Impact | Damage | Confidentiality |
| | Exploitability | Skilled |
| | Reproducibility | Produce in certain time |
| | Affected Users | Group of Users |

**Table 41: Unicode Encoding**

| | | |
|---|---|---|
| The attack aims to explore flaws in the decoding mechanism implemented on applications when decoding Unicode data format. [108] | | |
| Root Cause | Query Parameters Validation | |
| Attack Behavior | Single HTTP Request | |
| Attack Portion | HTTP Query Parameters | |
| Detection Model | Black List | |
| Attack Impact | Damage | All security services |
| | Exploitability | Skilled |
| | Reproducibility | Easy |
| | Affected Users | All |

**Table 42: Web Parameter Tempering**

| | |
|---|---|
| The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data. [109] | |
| Root Cause | Header and Parameters Validation |
| Attack Behavior | Single HTTP Request |
| Attack Portion | HTTP Headers, Query and Post Parameters |

|  |  |  |
|---|---|---|
| Detection Model | Hybrid |  |
| Attack Impact | Damage | Confidentiality, Integrity, Authorization |
|  | Exploitability | Very Skilled |
|  | Reproducibility | Difficult |
|  | Affected Users | Single User |

**Table 43: XPath Injection**

|  |  |  |
|---|---|---|
| XPath Injection attacks occur when a web site uses user-supplied information to construct an XPath query for XML data. By sending intentionally malformed information into the web site, an attacker can find out how the XML data is structured, or access data that he may not normally have access to. [110] |  |  |
| Root Cause | Parameters and Request Headers Validation |  |
| Attack Behavior | Single HTTP Request |  |
| Attack Portion | HTTP Headers, Query and Post Parameters |  |
| Detection Model | Black List |  |
| Attack Impact | Damage | Confidentiality |
|  | Exploitability | Skilled |
|  | Reproducibility | Easy |
|  | Affected Users | Group of Users |

## 3.3 Summary

The above study gathers all information regarding the web application attacks. Now our domain knowledge is complete. We will use this information in developing the web application ontology.

CHAPTER # 04

# Web Attacks Ontology

These chapters discuss web attacks ontology in detail. We have covered all web attacks that are discussed in the last chapter. This chapter describes high level diagram of our designed ontology that is shown in Figure8.

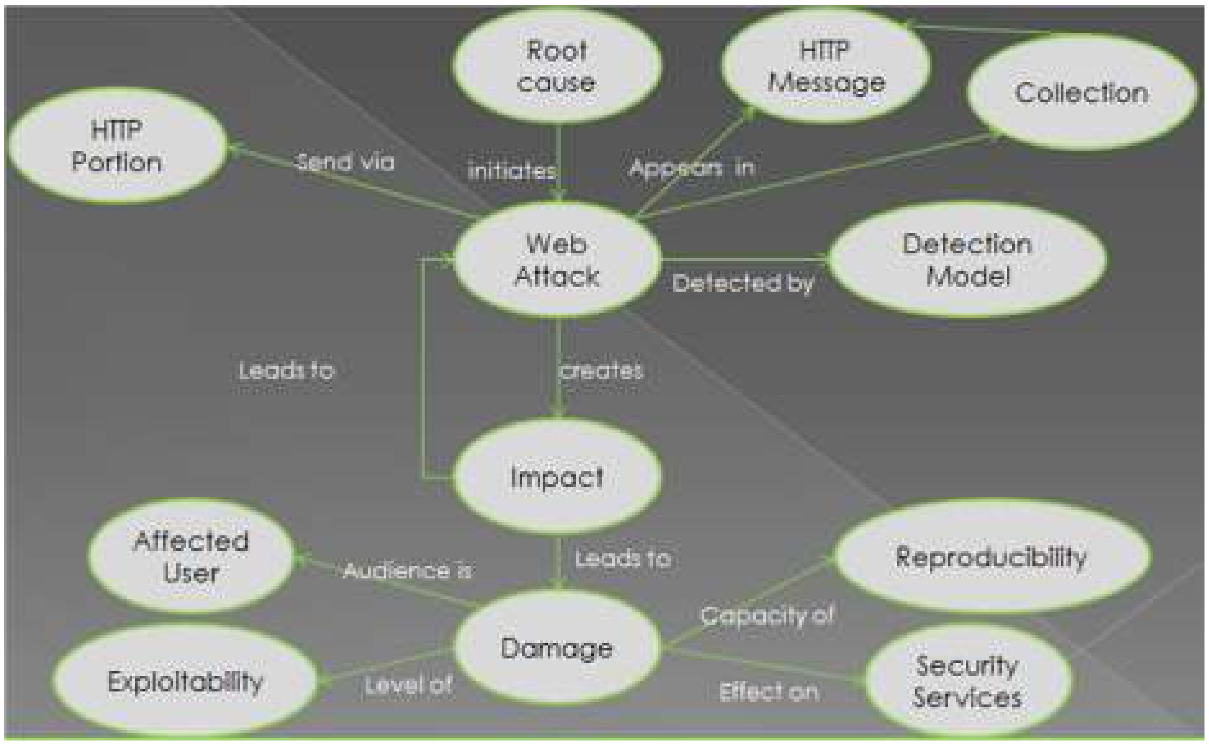

**Figure 8: Web Attacks Ontology-High Level Diagram**

The root cause class describes reason behind the generation of web application attacks. Each attack is sent over HTTP portion, this portion contain the values that cause the attack. Each attack uses one or more than one HTTP messages to damage successfully. We can detect these attacks using detection model as used in web application firewall now a day. Each web attack has an impact; this impact can lead to another web attack or can cause some damage. That we measured using four classes. These are affected users, exploitability, and reproducibility and security services.

## 4.1 Root Cause

Root cause will cover all possible causes of web application attacks that are the main reason behind each web application attacks. We studied all root causes and make them part of our ontology. This will help to detect attacks on the basis of their cause. The root causes are shown in Figure 3. The root cause is divided into three types; these are application, web server and browser specific.

### 4.1.1 Application Specific

This type of root cause cover all the problem that will help the attackers to launch attacks related to web application on which attack is launched. The application specific root cause can be

- Validation
- Logic

**Validation:** Validation issue in application will cause to launch the attack because of the malicious values. The input to the web application sent via HTTP parameters or HTTP headers. The validation problem cause many attacks to be launched.

**Logic:** This root cause helps the attackers to exploit weakness in the application logic that can be very helpful to launch attack. Logic root cause is further divided into the following

- Authentication
- Authorization
- Cleanup
- Error Handling

- Log Flaws

**<u>Authentication</u>**: This covers all root cause that is associated with the authentication section of the web application. Any web application that allowed different users to log in to it can have this issue. These issues can be related to weak password, broken session management or lockout on multiple failed login. These issues can cause authentication problem that has high severity level in web applications.

**<u>Authorization</u>**: This will cover all the root cause that is associated with the authorization part of the system. These root cause related to the Access Control List or the rights that a logged in user should have. This will help to identify all possible root causes that are exploited now days by the attacker.

**<u>Cleanup</u>**: This will cover all the root cause that is related to the cleaning the used resources. Every application used some of the web server resources while serving requests to the client. If these resources are not properly released this can be exploited by attacker to launch attack. This will cover all the issues related to the cleanup.

**<u>Error Handling</u>**: This class covers all root cause that is associated with the error handling of the web application. The information that's revealed to the attacker through improper error handling can help to launch target attack very easily. This root cause covers all error handling issues of the web application.

**<u>Log Flaws</u>**: Every application logs user activity which can further be used for analysis purposes. If attacker exploits that feature to impersonate his identity or can destroy the log. This can be very serious problem. These log flaws root causes cover all attacks that exploit the weakness related to application logs.

### 4.1.2 Web Server Specific

This type of root causes cover problems that is associated with the web server. The weakness of the web server helps us to detect all attacks that exploit them. These weaknesses are

- Request Parsing
- Cache
- Trace Method Enabled
- SSI Enabled
- Configuration Issue

**Request Parsing**: This root cause of web server helps the attacker to pass the malicious request if the web server does not parse the request according to the RFC 2616.

**Cache**: This root cause of web server helps the attacker to exploit the cache feature of the web server. This exploitation can help attacker to store malicious content in the cache. This will cover all root causes related to cache of the web server.

**Trace Method Enabled**: Trace method enabled on web server can become a weakness that helps the attacker to launch a special type of attack to gather information related to web server or can hit the users with information stealing.

**SSI Enabled**: This root cause helps in many attacks. It can be used to inject code in web application that can be executed on the user machines and can harm the system or users.

**Configuration Issue**: All attacks that help the attacker to exploit the incorrect web server configuration will cover in this category.

### 4.1.3 Browser Specific

This type of root cause covers entire problems that are associated with the web browsers. These are

- DOM Objects
- Same Origin Policy Weakness

**DOM Objects**: This root cause cover the problems associated with the DOM objects that are accessed using JavaScript. The exploitation of these can help the attacker to launch attacks that can help to steal information from the user system.

**Same Origin Policy Weakness**: This weakness helps in many attacks that exploit it. This is also related to browser and all browsers contain this policy but sometime this policy can become the root cause of many attacks.

## 4.2 HTTP Portion

This class helps to find portions that are used by attackers to send attacks on the web applications. These portions reside inside the HTTP packets, to show them the ontology is integrated and developed for Hyper Text Transfer Protocol [HTTP]. We developed this ontology in such a way so that it can be used for parsing of the incoming request and outgoing response. High level picture of HTTP ontology is shown in Figure 9.

The attack portions are the HTTP parameters that can be Query or Post. The headers are also used as attack portion for many attacks.

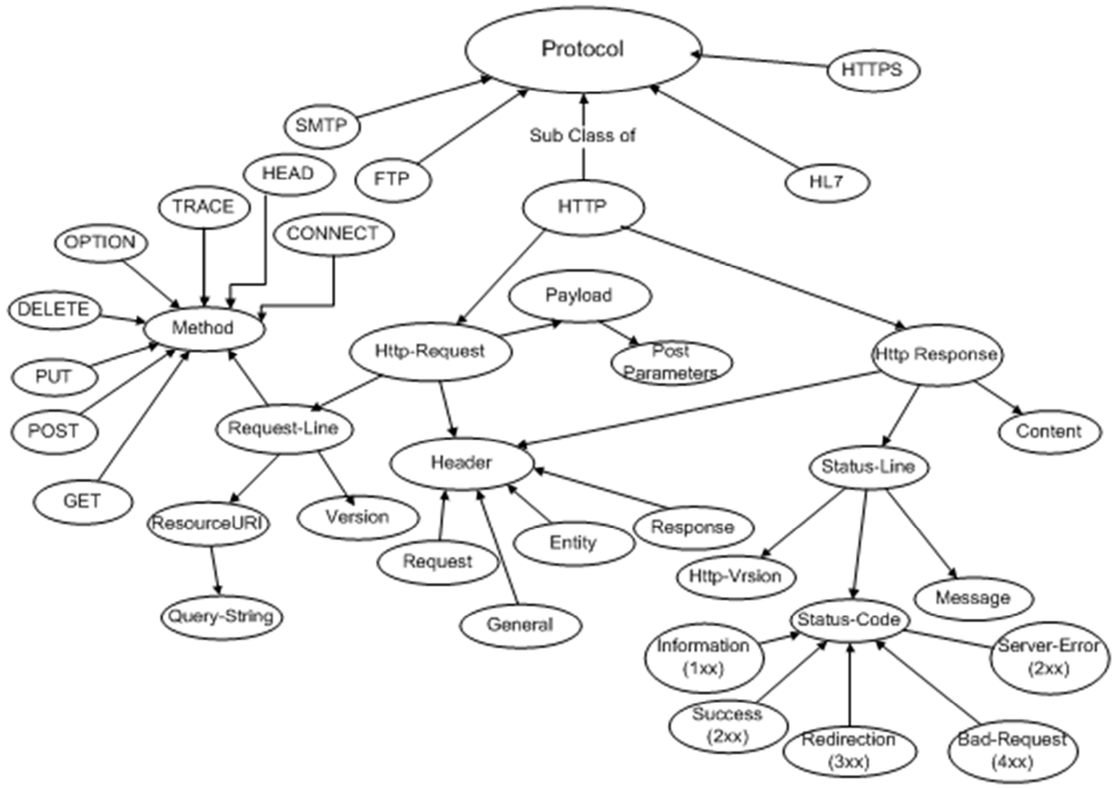

**Figure 9: HTTP Ontology-High Level Diagram**

## 4.3 Attack Behavior

This will help to analyze the behavior of attacks on basis of the message requires to launch the attack. Web application attacks could be launched using one or many http messages. This information helps us to find the attacks for which we need to maintain state to detect them. All attacks that sent via multiple HTTP messages need a state to detect them.

## 4.4 Detection Model

This class shows the detection model that is used in web application firewall for detection. These are shown in Figure 10. These are

- White Box
- Black Box

- Hybrid

**White Box**: This detection model needs some application information to detect a particular attack. Currently it is used by many web application firewalls that learn application profile then use that information to detect attacks.

**Black Box**: This detection model based on the signature that firewalls used to detect attacks. These signature match with all incoming data, if matched found then attack is detected otherwise request is normal.

**Hybrid**: This detection model used the features of the white list and black list. We can say that it is the combination of both. In it we also need application information and signature to detect attack.

## 4.5 Web Attacks

This class covers all web application attacks that we discussed in the Chapter 3. All the attributes that each attack has, will be applied using the property restriction.

## 4.6 Impact

Each attack has some impact on the system after success. This class covers the all possible consequences that attack can have afterwards. Our study shows that it can be divided into two portions, one is that attack impact can lead to some another attack and second attack can lead to some damage that caused because of that attack.

## 4.7 Damage

Each attack can lead to some damage as an impact. This damage can help to find out the attacks that are more dangerous for the system. Damage is also divided into four subclasses that are shown in Figure 10. Each subclass affects some of the security services on the basis of that information we measures the damage on the web application. We used four dimensions to measure the damage for each attack, these are

### 4.7.1 Affected Users

We measures the number of user affect with this attack. This will help us to measure the damage correctly. The values of this class are All, Group or Single.

### 4.7.2 Exploitability

To measure the damage we also note the level of attackers that required launching this attack. The values of this class are Novice, Skilled or Very Skilled.

### 4.7.3 Reproducibility

This measure helps to find that how much effort needed to launch this attack. The values of this class are Easy, Possible in Certain Time or Difficult.

### 4.7.4 Security Services

There are seven security services defined for the secure systems that should be up and running all the time otherwise we can say that our system is not secured if any of them is down or effected. These are shown in Figure 10.

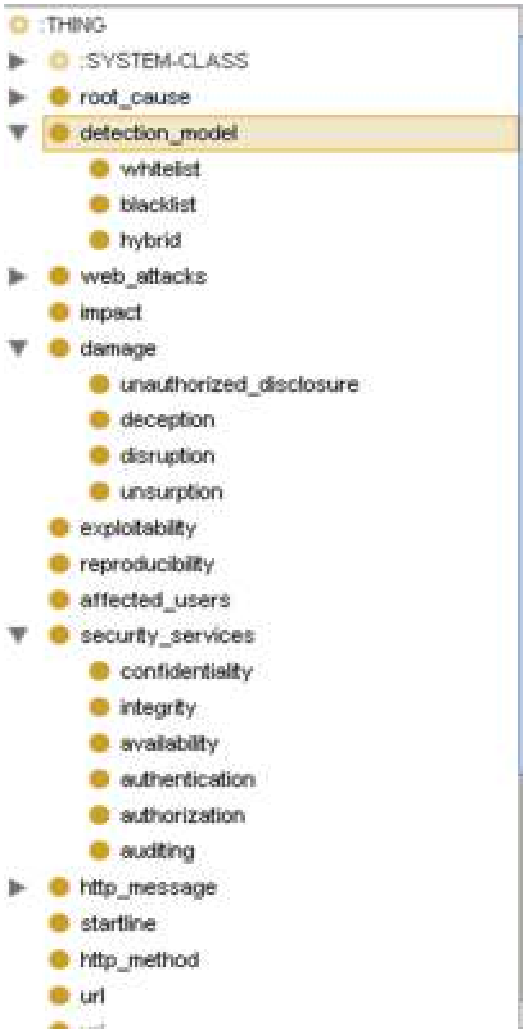

**Figure 10: Security Services**

## 4.8 Cause and Effect Design Pattern

We develop our ontology on the basis of Cause and Effect design pattern. This design pattern is taken from the medical domain. In medical domain, cause and effect pattern is mostly used to diagnose the disease. This pattern divides the problem into two sections, one is cause and other is the effect of that cause. This will help us to identify the actually cause of an effect to properly stop it in future. Figure 11 shows our ontology with this design pattern.

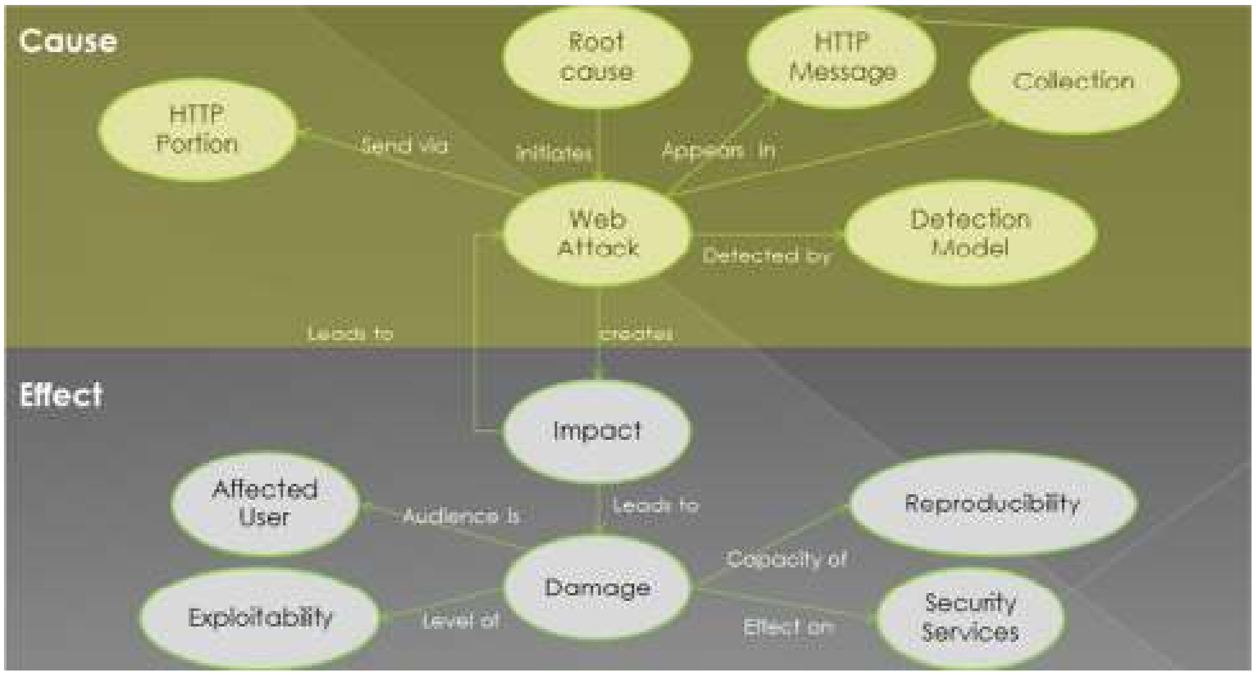

**Figure 11: Ontology with Design Pattern**

The section in green color shows the cause and the whole section generated some effect that we cover in the effect section using the impact and damage.

## 4.9 Web Attacks Example

In this section, we show two web attacks as an example, how we map them in our ontology. These attacks show the both impact type, attack as impact and damage as impact. These attacks are

- Account Lockout
- SQL Injection

### 4.9.1 Account Lockout

This attack launched, when user provides wrong password until web application lockout that username to stop the further login requests for that username. This made it possible for attacker to lockout all users. The ontology picture of this attack is shown in Figure 12; this figure shows the root causes and all other attributes.

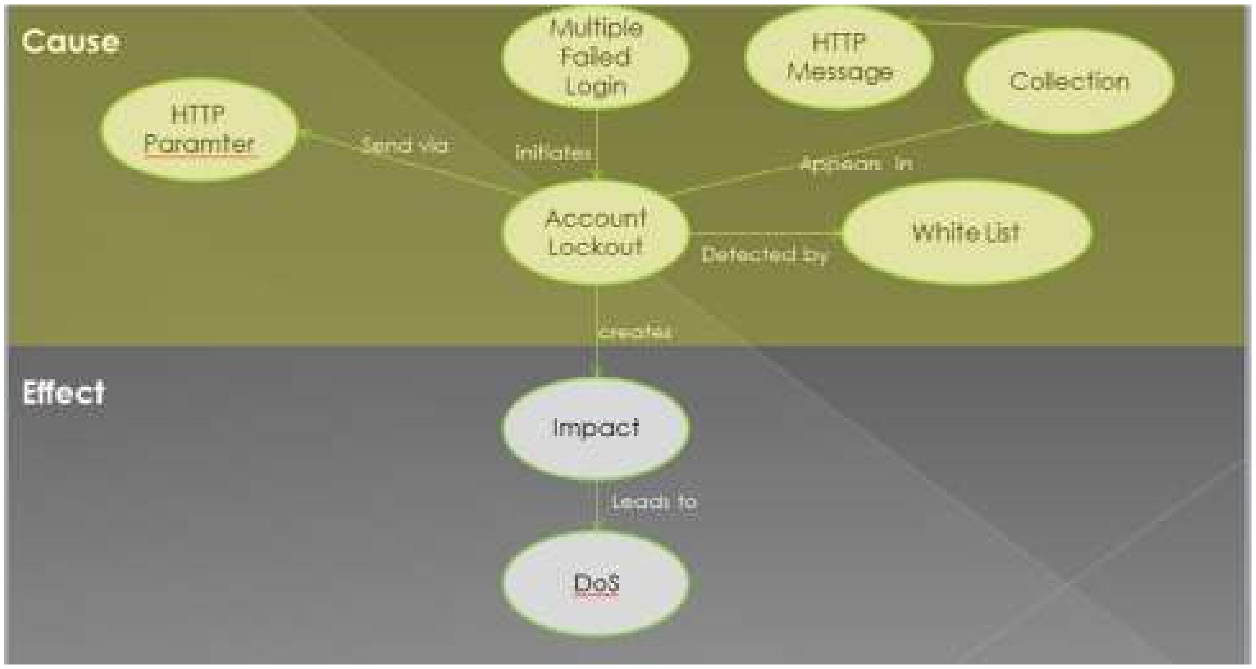

**Figure 12: Account Lockout**

### 4.9.2 SQL Injection

This cause of this attack is, when user provides malicious value in the parameters or headers to run the malicious SQL queries. This attack is possible on web application, if application have database. The diagram of this attack is shown on Figure 13.

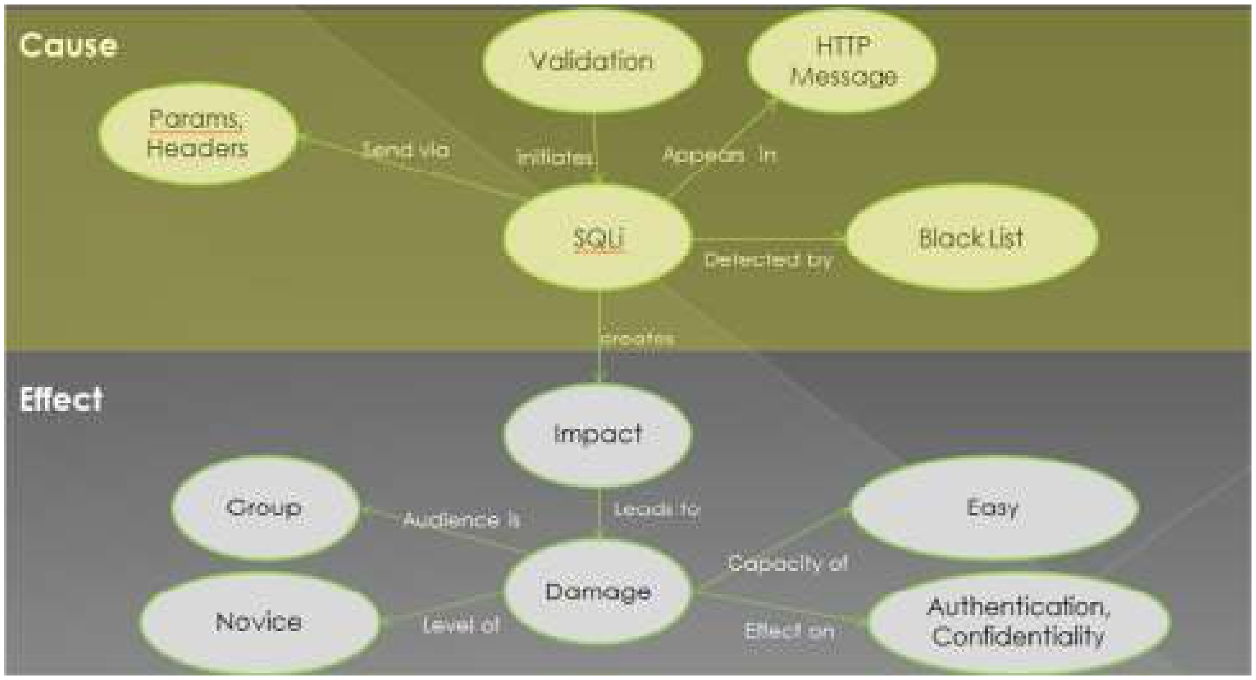

**Figure 13: SQL Injection**

## 4.10 Summary

This chapter shows proposed ontology that we developed. This ontology covers all web application attacks with their root causes and impact. This ontology can be used for analysis and detection purposes.

CHAPTER # 05

# Evaluation

This thesis proposed an ontology which is developed for web application attacks and this will help in detection of attacks in any web application firewall. To confirm the formal correctness of the ontology we have to evaluate it through some methodology that is recommended by community. Ontoclean [111] is the one of the best ontology validation methodology. We used it to validate our developed ontology.

## 5.1 Ontoclean

This section introduces ontoclean methodology that we used for evaluation. Ontoclean is very famous among community on the basis of some facts. These factors are mentioned blew

- provides the logic based argument for cleaning
- validate the ontological taxonomy relationships

There are some important terms that is referred in this chapter while we used this methodology, first of all we will discuss these terms before evaluation

**Rigid**: This property is essential for all possible instances of a class. For example, having brain is essential for all human beings.

Symbol of rigid that we used in evaluation is +R.

**Anti-Rigid**: This property depict that a particular property is not essential for all possible instances of a class. For example, being a student is not essential for all human beings.

Symbol of anti-rigid that we used in evaluation is ~R.

**Semi-Rigid**: This property is essential for some instances and not essential for remaining instances. For example, having brain is essential for living things.

Symbol of semi-rigid that we used in evaluation is -R.

**Identity**: This criterion works on the basis of recognition of entities to be same or different in the domain e.g. different identifiers of a person.

Symbol to identity if some concepts carry the identity criteria is +I and if not then –I.

**Unity**: It is the criteria on which basis we recognition of all parts / property that belongs to an entity and form an individual entity.

Symbol of unity if some concepts carry the unity criteria is +U and if not then –U. If not classify into these two then it will be ~U for anti-unity.

## 5.2 Ontoclean Evaluation Criteria

According to ontoclean we have to check the following condition while we evaluating ontology. These conditions are applied on all subclasses and sub properties to ensure that we made the correct relationships between them. Given two properties, p and q, when q subsumes p the following constraints hold:

- If q is anti-rigid, then p must be anti-rigid
- If q carries an identity criterion, then p must carry the same criterion
- If q carries a unity criterion, then p must carry the same criterion
- If q has anti-unity, then p must also have anti-unity

## 5.3 Ontology Statistics

This section shows the number of classes, properties and property restriction that we used in our ontology, this is shown in below table

Table 44: Ontology Statistics

| Serial # | Title | Total |
|---|---|---|
| 1 | Classes | 70 |
| 2 | Slots | 30 |
| 3 | Property Restrictions | 61 |

## 5.4 Ontology Evaluation

We follow three criteria during ontology evaluation. These evaluation criteria are very effective to proof the effectiveness and usefulness of the ontology. These are formal correctness, consistency, domain coverage and task orientation [112].

### 5.4.1 Formal Correctness

Formal correctness is ensured by using the Ontoclean methodology. We apply ontoclean rules on all our classes and properties to ensure its correctness. Our ontology prove that it is formally correct that is discussed in the below sections.

#### 5.4.1.1 Security Services and Its subclasses

We assign the symbols according to the properties that are discussed in the first section of this chapter. And then by applying the validation criteria we validate them. The evaluation picture is shown in Figure 14.

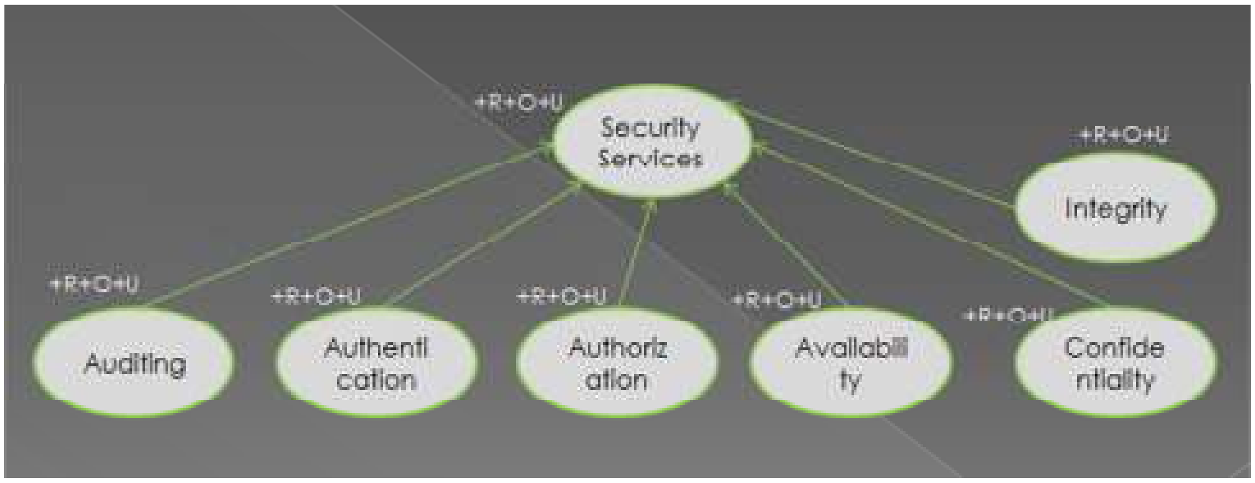

**Figure 14: Security Services Evaluation**

We assign rigid to security services, because all instance in our domain that belongs to security services can only be security services and cannot belong to any other class. And it has its own identity criteria that are shown with O. U are used for unity criteria. The same criteria are displayed in the subclasses. According to the conditions, it is correct.

#### 5.4.1.2 Damage and Its subclasses

The second evaluation picture is shown in Figure 15. In this figure we evaluate the damage and its subclasses. We assign +R, +O and ~U to these classes, because these are rigid, have their own criteria of identity and it is anti-unity, because there is no mechanism that help us to measure the damage or it is not representing something as a whole. It needs help of other classes to for measurement that’s why we cannot say it as unity. According to the evaluation criteria mentioned in ontoclean, these are also correct.

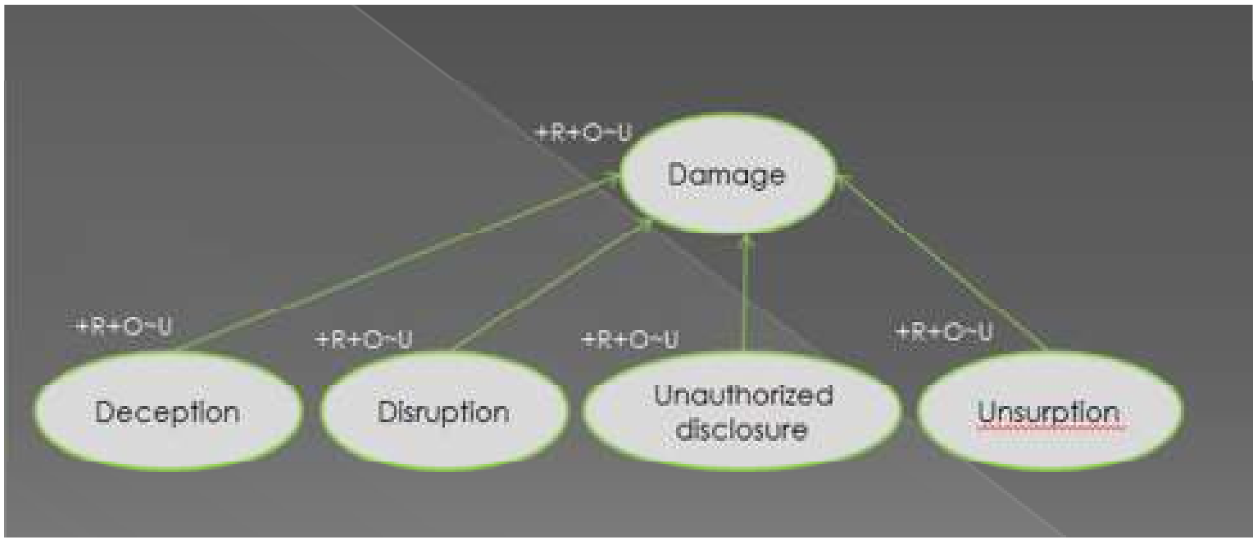

**Figure 15: Damage Evaluation**

#### 5.4.1.3 Web attack and its subclasses

We evaluate the web attack and its subclasses, they are Rigid, Identity and have unity criteria on the basis of these we evaluate it and confirmed their validation by ontoclean methodology.

#### 5.4.1.4 Root Cause and its subclasses

We evaluate the root cause and its subclasses, they are Rigid, Own Identity and have unity criteria on the basis of these we evaluate it and confirmed their validation by ontoclean methodology.

#### 5.4.1.5 Detection Model and its subclasses

We evaluate the detection model and its subclasses, they are Rigid, Own Identity and have unity criteria on the basis of these we evaluate it and confirmed their validation by ontoclean methodology.

### 5.4.2 Consistency

Ontology consistency is evaluated using Pellet[113] reasoner. This reasoner ensured that ontology is consistent and can be utilized for detection without creating any inconsistency.

### 5.4.3 Domain Coverage

To verify that our ontology covers all web application attacks. We used OWASP [8], OWASP is an open web application security project. They listed all web application attacks that exist now days. We follow OWASP attack list to develop this ontology. That's why our ontology covers all web application attacks and provides required information that is needed to detect them.

### 5.4.4 Task Orientation

In this criterion we have to show that our ontology fulfills the purpose for which we developed it. Our purpose is to use ontology to detect web application attacks. In this section we present an example that shows the effectiveness of our ontology in detection of attacks.

#### 5.4.4.1 Input Validation Attacks

This section shows the mechanism how effectively our ontology is utilized to detect input validation attacks. Each input validation attacks has set of rules associated with their model that will be applied on HTTP portion for detection. To get the required portion from the ontology is done using inference. Figure 16 shows ontology before without inference.

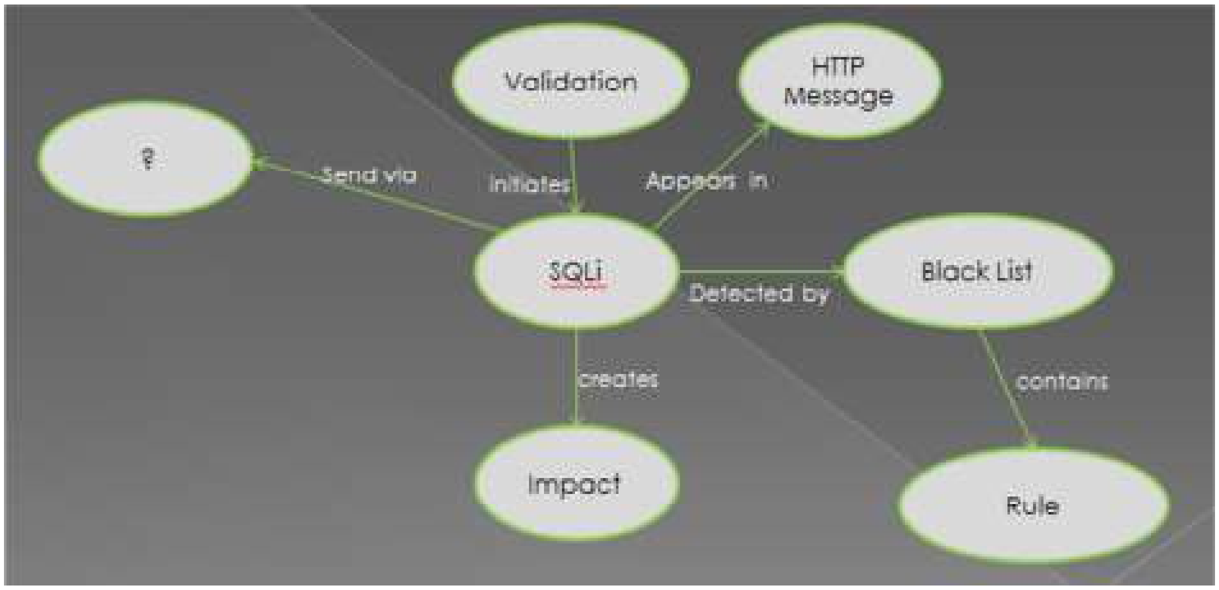

**Figure 16: Without Inference**

One sample inference rule is shown below that inferred the portion on which we apply rules.

```
[rule1:
        (?A rdf:type WebAttack),
        (?R rdf:type root_cause),
        (?I rdf:type Validation),
        (?H rdf:type http_header),
        (?P rdf:type http_parameters),
        (?R initiates ?A),
        (?R owl:sameas ?I)
        ➔  (?A send_via  (?H,?P))
]
```

After executing the inference engine, the portion is automatically derived that is shown in Figure 17.

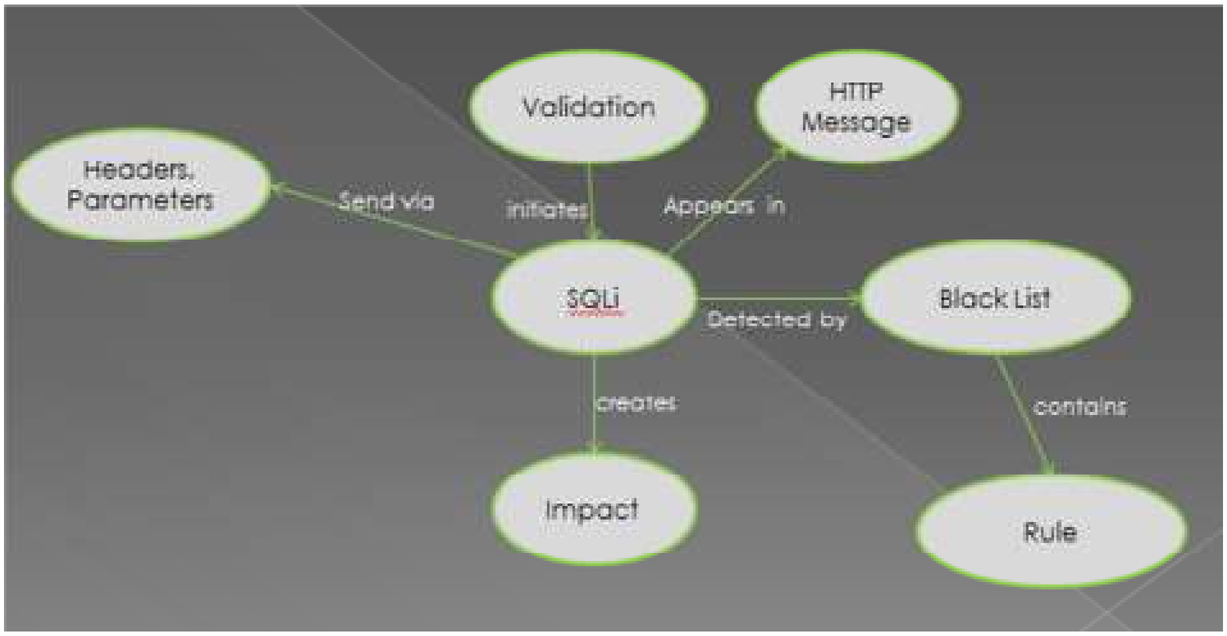

**Figure 17: After Inference**

## 5.5 Summary

We evaluate every class and relationships between the classes according to the methodology.

This will ensure that our ontology is formally correct and according to the ontology engineering.

We also ensure that it is complete and cover the whole domain.

CHAPTER # 06

# Conclusion & Future Enhancements

This chapter concludes the overall work done in this thesis from the study of the domain to final development of ontology with evaluation. The developed ontology is formally verified in the evaluation phase that confirms its correctness.

## 6.1 Conclusion

After literature survey we found that existing solutions like signatures and rules for web application firewalls are not sufficient for web attack detection. These techniques can be easily bypasses using some evasion. They are also not much expressive and have no reasoning mechanism. To overcome these shortcomings, community proposes that semantic techniques should be used. Community used semantic techniques effectively for the network security but completely ignore the web application domain. To use semantic techniques in web application domain, we should have a common vocabulary for web application attacks in the shape of ontology. This ontology then can help us in reasoning. This reasoning can help us to analyze and detect the web application attacks efficiently. That's why we develop the web application attacks ontology that can be used by community in future for web application attack detection. This ontology is developed according to the ontology engineering and validated using the ontoclean methodology.

## 6.2 Future Work

The web application attacks ontology is developed and now we will utilize it in the web application firewall to detect the web application attacks. To detect the attacks we may need some inference rules to get the inferred knowledge, in future we will do research on it and utilize it efficiently. A plugin to parse the CVE web application vulnerabilities data and analyze it according to our ontology can be developed to find the most popular root causes, the relationships of the attacks with each other and the damage that an attack caused.

**Annexure - A**

# References

[1] Common Gateway Interface [http://oreilly.com/openbook/cgi/ch01_01.html]

[2] Active Server Pages [http://msdn.microsoft.com/en-us/library/aa286483.aspx]

[3] Asp.Net [http://www.asp.net/]

[4] Java Server Pages Technology [http://java.sun.com/products/jsp/]

[5] PHP: Hypertext Preprocessor [http://www.php.net/]

[6] WhiteHat Security Statistics Report [https://www.whitehatsec.com/resource/stats.html]

[7] MITREWeb application attacks statistics [http://www.mitre.org/]

[8] OWASP[http://www.owasp.org]

[9] Acunetix Web application attacks statistic [http://www.acunetix.com/]

[10] JeffOrloff, ApplicureWebhacking Facts and Figure [ http://www.applicure.com/blog/web-application-hacking-facts-figures]

[11] Robert Abela, Website DefenderGeneral facts and figure on hacking [http://www.sitesecuritymonitor.com/web-hacking-facts/]

[12] Christopher Welty. Ontology Research. AI Magazine. 2003

[13] Thomas R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Formal Ontology in Conceptual Analysis and Knowledge Representation. 1993

[14] Javier Gonzalez-Castillo, David Trastour, and Claudio Bartolini. Description Logics for Matchmaking of Services. Hewlett-Packard Company Technical Report. 2001

[15] OWASP SQLi [https://www.owasp.org/index.php/SQL_Injection]

[16] OWASP Web Application Firewall [https://www.owasp.org/index.php/Web_Application_Firewall]

[17] Jeffrey Undercoffer, Anupam Joshi, and John Pinkston,"Modeling Computer Attacks: An Ontology for Intrusion Detection", 6th International Symposium on Recent Advances in Intrusion Detection

[18] Yanxiang He, Wei Chen, Min Yang, and WenlingPeng,"Ontology Based Cooperative Intrusion Detection System", In Proceedings of NPC'2004

[19] Salvador Mandujano, Arturo Galvan and Juan A. Nolazco,"An Ontology-based Multiagent Architecture for Outbound Intrusion Detection", Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference

[20] F. Abdoli and M. Kahani,"Ontology-based Distributed Intrusion Detection System", Computer Conference, 2009. CSICC 2009. 14th International CSI

[21] Abdul Razzaq, Hafiz Farooq Ahmed, Ali Hur and NasirHaider,"Ontology based Application Level Intrusion Detection System by using Bayesian Filter", Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference

[22] Gustavo Isaza, Andrés Castillo, Manuel López and Luis Castillo,"Towards Ontology-Based Intelligent Model for Intrusion Detection and Prevention", Advances in Intelligent and Soft Computing, 2009

[23] Andrew Simmonds, Peter Sandilands, Louis van Ekert,"An Ontology for Network Security Attacks", In Proceedings of the 2nd Asian Applied Computing Conference (AACC'04)

[24] AniruddhaChandra,”Ontology for MANET Security Threats”, Proc. NCON, Krishnankoil, Tamil Nadu, Mar. 2005

[25] DimitrisGeneiatakis , Costas Lambrinoudakis,”An ontology description for SIP security flaws”, Journal of Computer Communications archiveVolume 30 Issue 6, April 2007

[26] Nora Cuppens-Boulahia, Fr´ ed´ ericCuppens, Jorge E. L´ opez de Vergara,Enrique V´ azquez, Javier Guerra, Herv´ e Debar,”An ontology-based approach to react to network attacks”, Risks and Security of Internet and Systems, 2008. CRiSIS '08. Third International Conference

[27] F.Abdoli, N.Meibody, R.Bazoubandi,”An Attacks Ontology for computer and networks attack”, Innovations and Advances in Computer Sciences and Engineering, 2010

[28] Anya Kim, Jim Luo, and MyongKang,”Security Ontology for Annotating Resources”, Research Lab, NRL Memorandum Report, 2005

[29] Wolfgang Nejdl, Daniel Olmedilla, Marianne Winslett, and Charles C. Zhang,”Ontology-Based Policy Specification and Management”, 2nd European Semantic Web Conference (ESWC)

[30] Stefan Fenz and Edgar Weippl,”Ontology based IT-security planning”, Dependable Computing, 2006. PRDC '06. 12th Pacific Rim International Symposium

[31] Almut Herzog, NahidShahmehri and ClaudiuDuma,”An Ontology of Information Security”, International Journal of Information Security (2007)

[32] Mansoor Ahmed, Amin Anjomshoaa, ThoManh Nguyen, and A Min Tjoa,”Towards an Ontology-based Organizational Risk Assessment in Collaborative Environments Using the Semantic LIFE”, The Second International Conference on Availability, Reliability and Security (ARES'07)

[33] Tung JuChiang and Jen ShiangKouh and Ray-I Chang,”Ontology-based Risk Control for the Incident Management”, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.11

[34] Fong-Hao Liu and Wei-TsongLee,”Constructing Enterprise Information Network Security Risk Management Mechanism by Ontology”, Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference

[35] Z. Maamar, N.C. Narendrab, S. Sattanathanc,”Towards an ontology-based approach for specifying and securing Web services”, Information and Software Technology (2006)

[36] ArtemVorobiev and Jun Han,”Security Attack Ontology for Web Services”, Semantics, Knowledge and Grid, 2006. SKG '06. Second International Conference

[37] Hsien-Der Huang, Tsung-Yen Chuang, Yi-Lang Tsai, and Chang-ShingLee,”Ontology-based Intelligent System for Malware Behavioral Analysis”, Fuzzy Systems (FUZZ), 2010 IEEE International Conference

[38] DREAD [http://msdn.microsoft.com/en-us/library/ff648644.aspx]

[39] RFC 2828 [http://www.ietf.org/rfc/rfc2828.txt]

[40] OWASP Account Lockout [https://www.owasp.org/index.php/Account_lockout_attack]

[41] Account Lockout Attack –Security Knowledge base [http://shalb.com/kb/entry/4/]

[42] OWASP Argument Injection [https://www.owasp.org/index.php/Argument_Injection_or_Modification]
[43] Argument Injection –Security Knowledge base [http://shalb.com/kb/entry/18/]
[44] OWASP Asymmetric Resource Consumption [https://www.owasp.org/index.php/Asymmetric_resource_consumption_%28amplification%29]
[45] Asymmetric Resource Consumption –Security Knowledge base [http://shalb.com/kb/entry/42/]
[46] OWASP Blind SQLi [https://www.owasp.org/index.php/Blind_SQL_Injection]
[47] Blind SQLi –Security Knowledge base [http://shalb.com/kb/entry/19/]
[48] OWASP Blind XPATH Injection [https://www.owasp.org/index.php/Blind_XPath_Injection]
[49] Blind XPATH Injection –Security Knowledge base [http://shalb.com/kb/entry/20/]
[50] OWASP Brute Force [https://www.owasp.org/index.php/Brute_force_attack]
[51] Brute Force –Security Knowledge base [http://shalb.com/kb/entry/37/]
[52] OWASP Buffer Overflow [https://www.owasp.org/index.php/Buffer_overflow_attack]
[53] Buffer Overflow –Security Knowledge base [http://shalb.com/kb/entry/175/]
[54] OWASP Cache Poisoning [https://www.owasp.org/index.php/Cache_Poisoning]
[55] Cache Poisoning –Security Knowledge base [http://shalb.com/kb/entry/5/]
[56] C. Linhart, A. Klein, R.Heled and S.Orrin,"White Paper: HTTP Request Smuggling", Watchfire (2005)
[57] OWASP Code Injection [https://www.owasp.org/index.php/Code_Injection]
[58] Code Injection –Security Knowledge base [http://shalb.com/kb/entry/21/]
[59] OWASP Command Injection [https://www.owasp.org/index.php/Command_Injection]
[60] Command Injection –Security Knowledge base [http://shalb.com/kb/entry/22/]
[61] OWASP Comment Injection [https://www.owasp.org/index.php/Comment_Injection_Attack]
[62] Joon S. Park and R.Sandhu,"Secure Cookies on the Web", IEEE INTERNET COMPUTING (2000)
[63] OWASP Cross Frame Scripting [https://www.owasp.org/index.php/Cross_Frame_Scripting]
[64] Cross Frame Scripting –Security Knowledge base [http://shalb.com/kb/entry/23/]
[65] OWASP Cross Site History Manipulation [https://www.owasp.org/index.php/Cross_Site_History_Manipulation_%28XSHM%29]
[66] OWASP Cross Site Tracing [https://www.owasp.org/index.php/Cross_Site_Tracing]
[67] Jesse Burns,"White Paper: Cross Site Request Forgery", Information Security Partners, LLC (2007)
[68] T. Alexenko, M. Jenne, S. Deb Roy and W.Zeng,"Cross-Site Request Forgery: Attack and Defense", IEEE CCNC (2010)

[69] OWASP Cross Site Request Forgery [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29]
[70] OWASP Cross User Defacement [https://www.owasp.org/index.php/Cross-User_Defacement]
[71] Cross User Defacement –Security Knowledge base [http://shalb.com/kb/entry/6/]
[72] F.Kerschbaum,"Simple Cross-Site Attack Prevention", SecureComm (2007)
[73] OWASP Cross Site Scripting [https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29]
[74] OWASP Custom Special Character Injection [https://www.owasp.org/index.php/Custom_Special_Character_Injection]
[75] Custom Special Character Injection –Security Knowledge base [http://shalb.com/kb/entry/44/]
[76] OWASP Denial of Service [https://www.owasp.org/index.php/Denial_of_Service]
[77] OWASP Direct Dynamic Code Evaluation [https://www.owasp.org/index.php/Direct_Dynamic_Code_Evaluation_%28%27Eval_Injection%27%29]
[78] Direct Dynamic Code Evaluation –Security Knowledge base [http://shalb.com/kb/entry/45/]
[79] OWASP Direct Static Code Injection [https://www.owasp.org/index.php/Direct_Static_Code_Injection]
[80] Direct Static Code Injection –Security Knowledge base [http://shalb.com/kb/entry/25/]
[81] OWASP Double Encoding [https://www.owasp.org/index.php/Double_Encoding]
[82] Double Encoding –Security Knowledge base [http://shalb.com/kb/entry/46/]
[83] OWASP Forced Browsing [https://www.owasp.org/index.php/Forced_browsing]
[84] Forced Browsing –Security Knowledge base [http://shalb.com/kb/entry/47/]
[85] OWASP Format String attack [https://www.owasp.org/index.php/Format_string_attack]
[86] Format String attack –Security Knowledge base [http://shalb.com/kb/entry/176/]
[87] OWASP Full Path Disclosure [https://www.owasp.org/index.php/Full_Path_Disclosure]
[88] Full Path Disclosure –Security Knowledge base [http://shalb.com/kb/entry/27/]
[89] OWASP HTTP Request Smuggling [https://www.owasp.org/index.php/HTTP_Request_Smuggling]
[90] OWASP HTTP Response Splitting[https://www.owasp.org/index.php/HTTP_Request_Smuggling]
[91] Diabolic Crab,"White Paper: HTTP ResponseSplitting", Digital Paradox (2005)
[92] OWASP LDAP Injection [https://www.owasp.org/index.php/LDAP_injection]
[93] LDAP Injection –Security Knowledge base [http://shalb.com/kb/entry/28/]
[94] OWASP Page Hijacking [https://www.owasp.org/index.php/Page_Hijacking]
[95] OWASP Parameter Delimiter [https://www.owasp.org/index.php/Parameter_Delimiter]
[96] OWASP Path Manipulation [https://www.owasp.org/index.php/Path_Manipulation]
[97] OWASP Path Traversal [https://www.owasp.org/index.php/Path_Traversal]

[98] OWASP ReDoS [https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS]
[99] OWASP Repudiation Attack [https://www.owasp.org/index.php/Repudiation_Attack]
[100] F. Valeur, D. Mutz, and G. Vigna,"A Learning-Based Approach to the Detectionof SQL Attacks", DIMVA (2005)
[101] OWASP SQLi [https://www.owasp.org/index.php/SQL_Injection]
[102] OWASP Service Side Injection [https://www.owasp.org/index.php/Server-Side_Includes_%28SSI%29_Injection]
[103] OWASP Session Prediction [https://www.owasp.org/index.php/Session_Prediction]
[104] OWASP Session Fixation [https://www.owasp.org/index.php/Session_fixation]
[105] Y. Takamatsu, Y.Kosuga, and K.Kono,"Automated Detection of Session Fixation Vulnerabilities", 19th international conference on World Wide Web (POSTER SESSION in WWW '10) (2010)
[106] OWASP Session Hijacking [https://www.owasp.org/index.php/Session_hijacking_attack]
[107] Ben Adida,"SessionLock:Securing Web Session against Eavesdropping", 17th international conference on World Wide Web(2008)
[108] OWASP Unicode Encoding [https://www.owasp.org/index.php/Unicode_Encoding]
[109] OWASP Parameter Tampering [https://www.owasp.org/index.php/Web_Parameter_Tampering]
[110] OWASP XPath Injection [https://www.owasp.org/index.php/XPATH_Injection]
[111] N.Guarino and C. A. Welty,"An Overview of OntoClean", The Handbook on Ontologies-Berlin:Springer-Verlag, 2004
[112] M. Brochhausen and A. Spear, "The ACGT Master Ontol-ogy and its applications towards an ontology-driven cancer research and management system", Journal Biomedical Informatics, 2010