Nemesis: Automated Architecture for Threat Modeling and Risk Assessment for Cloud Computing

Patrick Kamongi[1], Mahadevan Gomathisankaran[2], Krishna Kavi[3]
Computer Science and Engineering
University of North Texas
Denton, TX 76203, USA
patrickkamongi@my.unt.edu[1], mgomathi@unt.edu[2], kavi@cse.unt.edu[3]

Abstract

It is critical to ask and address the following type of questions, both as a cloud computing architect who has designed and deployed a public, or private, or hybrid cloud; or a user who benefits from available cloud services: What are the types of threats facing the cloud's assets? Is there any scale to indicate the cloud's assets threat level? Is there any metric to characterize critical vulnerabilities facing the cloud's assets? In this paper, we present a novel automated architecture for threat modeling and risk assessment for cloud system called Nemesis, which address all the above and other related questions. With Nemesis, we use ontologies knowledge bases to model the threats and assess the risks of the given cloud system. To realize this feat, we built ontologies for vulnerabilities, defenses and attacks and automatically instantiate them to generate the Ontologies Knowledge Bases (OKBs). These OKBs capture the relationship between vulnerabilities, defenses mechanisms and attacks. We use the generated OKBs and Microsoft STRIDE model [1] to classify the threats and map them to relevant vulnerabilities. This is used together with the cloud configurations and the Bayesian threat probability model in assessing the risk. Apart from classifying the given cloud system's threats and assessing its risk, we deliver two useful metrics to rank the severity of classified threat types and to evaluate exploitable vulnerabilities. In addition, we recommend an alternative cloud system's configuration with a lower perceived risk, and mitigations techniques to counter classified threat types. For the proof of concept of our proposed architecture, we have designed an OpenStack's [2] based cloud and deployed various services. Then, we evaluated our Nemesis, and presented our findings. Our proposed architecture can help evaluate the security threat level of any cloud computing configurations, and any configurations of shared technologies found in computing systems.

1 Introduction

Cloud computing is defined as the delivery of on-demand computing resources; everything from applications to data centers over the Internet on a pay-for-use basis [3]. Its design principle revolve around a custom or an open source cloud operating system that is in charge to control and provision allocated resources throughout the datacenter(s). Cloud computing services are deployed mainly in models such as: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). For example, OpenStack [2] cloud operating system enables developers to design any cloud computing deployment model such as: Public, Private or Hybrid cloud to support any of the cloud computing services. It is critical to ensure that the cloud computing services can stand against any security threat.

AlertLogic presented a recent cloud security report on their honeypot research findings [4]. These honeypots were deployed in public cloud infrastructures around the world with the intent to observe the types and frequencies of attacks, and how the attacks vary geographically. This report shows the origin of the attacks to a specific region along with their frequencies; For example the US had the highest proportion of attacks on HTTP (attack vector) compared to other regions, which could be due to its higher web adoption. These results show how real cloud's assets are being targeted and that there are imminent threats.

One view to go about discovering any likely security threat type towards cloud's assets, is to perform a security assessment task by identifying known vulnerabilities and their exploits. This task becomes complex due to a variety of shared technologies that are part of the cloud design and deployment. According to Cloud Security Alliance (CSA) latest Cloud Computing Top Threats in 2013 report [5], "Shared Technology Vulnerabilities" is ranked among the top threats facing cloud computing ecosystem. It has been shown that cloud computing threats are influenced by different agents, where some are a result of inherent vulnerabilities found in shared technologies used, and others come to life due to the composition of services of shared technologies.

Main pillars of cloud's assets that needs to be protected from threat agents are: confidentiality, integrity, availability, consistency, control, and audit [6]. When assessing security threats, it is useful to ask questions like:

- How can an attacker change the authentication data?
- What is the impact if an attacker can read the user profile data?
- What happens if access is denied to the user profile database?

Using this analogy of thinking about threats, there exist threat models like STRIDE [1] that help to anticipate a class of threat types that any given cloud's assets could be facing and techniques to mitigate them.

To unravel the complexity of performing vulnerability assessment for cloud computing services, there have been some existing works that had laid out the ground in terms of collecting and publishing known Information Technology Product's vulnerabilities, exploits and mitigations such as: National Vulnerability Database (NVD) [7], and Exploit Database [8]. And vulnerability assessment frameworks geared for cloud computing like VULCAN [9].

There is a need for an optimal solution for cloud security threat modeling which incorporates the vulnerability assessment process, and then offers an actionable risk analysis indicator. To rise up to this challenge, we propose a novel Automated Architecture for Threat Modeling and Risk Assessment for Cloud Computing called NEMESIS. Our architecture offers a state of the art automated portal from start to finish of Cloud's assets threat modeling and risk assessment.

The rest of this paper is organized as follows: In Section 2 we present some background key features of our work, while in Section 3 we present our novel Nemesis architecture's design and implementation details; the NEMESIS experiments and evaluations are detailed in Section 4; in section 5 we discuss the related works and finally, conclusions and further research directions are given in Section 6.

2 Background

2.1 Cloud Computing

At the highest level, the cloud Operating System (OS) does what a traditional operating system does like managing applications and hardware but at the scope and scale of cloud computing. Nowadays, there exist different type of Cloud OSes; for example OpenStack [2] which is an open source project and currently available in many flavors where its most popular flavor is tailored in Ubuntu [10]. For an organization to relies on using any Cloud OS solution, it means that they can shift to more efficiently managing datacenter resources as a whole, including networking, storage and compute, in addition to new possible add-on customization tailored to their services and computing resources needs. In general, Cloud OSes are designed around a wide range of technologies, and it has been shown that any software/hardware technology at some point after being produced shows one or more weaknesses/vulnerabilities that could later be exploited. This trend of inherent flaws could lead to various threats to the make of cloud's assets that utilizes these shared technologies. This alarming fact calls for a continuing vulnerability assessment culture of Cloud's components and configurations.

2.2 Ontology Knowledge Bases (OKBs)

There exist some open source communities that are trying to publish reported known vulnerabilities along with their exploits, and their fixes if they exist. However, these communities publish reported vulnerabilities/exploits/fixes per one individual technology, this information does not help identify issues with complex system that utilizes shared technologies, therefore a need to model this information and allow its interoperability into complex system comes in the picture. Some existing works have attempted to model vulnerabilities, exploits/attacks, and fixes/defenses into security ontology knowledge bases due to their formalism and semantic inference high capabilities to unravel the complexity found in vulnerability assessment for cloud computing systems.

In our previous work [9] on VULCAN framework, we designed and implemented security ontology knowledge bases (Vulnerabilities, Attacks and Defenses) to facilitate in the automatic vulnerability assessment task for the cloud. As illustrated in Figure 1, it enables us:

- To generate an ever growing domain of knowledge that encompasses all areas of vulnerability assessments from both the offensive to defensive side
- To be able to link and search recursively the association of any given attack to which vulnerabilities can be exploited during the attack and vice versa
- To provide effective Defense mechanisms that factor in the source of vulnerability and attacks and how they could be mitigated
- To study vulnerabilities lifecycle

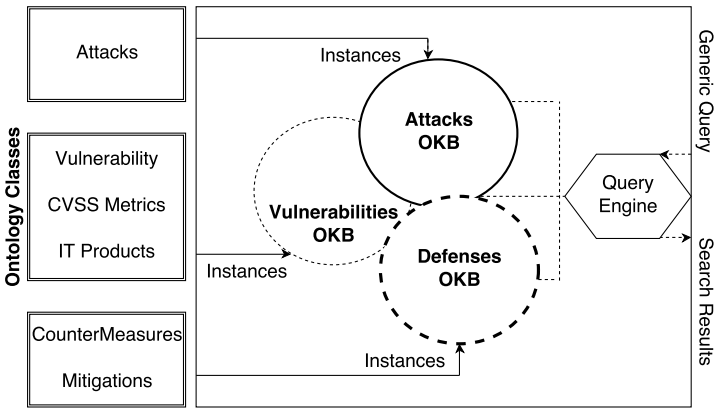

Figure 1: VULCAN - Security Ontology Knowledge Bases

Using this rich information about what are the vulnerabilities and exploits of a given technology or its association in the making of cloud's assets; we have enough supporting facts to go about performing threat modeling and risk assessment of any cloud's setting.

2.3 Threat Modeling

Given an adversary threat model which assumes that any attacker is highly capable, and well motivated; calls for a thorough investigation to identify the areas where any given cloud system is most vulnerable during its design or deployment phase. This way, one can choose the appropriate tools and implement the best design to protect the cloud's assets. To address this challenge, we adopted Microsoft STRIDE threat model [1] that enables us to classify and rank found threat types for each of the cloud's building block which is made of various shared technologies. STRIDE is an acronym for six threat categories described as:

- *Spoofing Identity.* An example of identity spoofing is illegally accessing and then using another user's authen-

tication information, such as username and password.

- *Tampering with data.* Data tampering involves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet.
- *Repudiation.* Repudiation threats are associated with users who deny performing an action without other parties having any way to prove otherwise for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations. Nonrepudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package.
- *Information Disclosure.* Information disclosure threats involve the exposure of information to individuals who are not supposed to have access to it for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers.
- *Denial of Service.* Denial of service (DoS) attacks deny service to valid users for example, by making a Web server temporarily unavailable or unusable. It is critical to protect against certain types of DoS threats simply to improve system availability and reliability.
- *Elevation of Privilege.* In this type of threat, an unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and become part of the trusted system itself, a dangerous situation indeed.

In section 3, we describe one of our automated procedure to classify any discovered vulnerability into STRIDE model. And if there's not any known countermeasure, we use STRIDE mitigation techniques [11] as shown in Table 1 to offer relevant mitigation recommendations towards evading enumerated threat types to the given cloud's assets.

Threat Type	**Mitigation Technique**
Spoofing Identity	Authentication Protect secrets Do not store secrets
Tampering with Data	Authorization Hashes Message Authentication Codes Digital signatures Tamper-resistant protocol
Repudiation	Digital signatures Timestamps Audit trails
Information disclosure	Authorization Privacy-enhanced protocols Encryption Protect secrets Do not store secrets
Denial of service	Authentication Authorization Filtering Throttling Quality of service
Elevation of privilege	Run with least privilege

Table 1: STRIDE Mitigation Techniques

2.4 Risk Assessment

Risk assessment is a critical task that involves weighting accumulated measurements into a refined indicator, in our case that indicator reflect the risk level of found vulnerabilities, exploits, defenses per threat type generated from STRIDE model as described in section 3. Various risk assessment models such as DREAD [12] and Fenz's [13] work on "An ontology-and Bayesian-based approach for determining threat probabilities" have been proposed but they all offer either offline manual or not fully automated evaluations.

The advantage of the proposed Bayesian threat probability determination [13] is that it gives the risk manager a methodology to determine the threat probability in a structured and comprehensible way. The methodology is illustrated in Figure 3 and each building block has its calculation schema fully documented. In addition to the Bayesian calculation schema, this work uses the security ontology to populate the proposed methodology. Using the same methodology, we populate it using our vulnerabilities, attacks and defenses ontology knowledge bases relevant data toward inferring the risk indicator for the given cloud's assets. Within our experiments studies in Section 4, we were able to improve this Fenz's methodology in terms of sound methods and techniques to gather, store, and provide crucial threat probability calculation components as shown in Figure 3 adopted from Fenz's [13] work and updated to use OKBs as the source of ontological knowledge.

In our proposed architecture, we bring these useful predictive models and metrics into an automated process, therefore to generate a risk indicator for the given cloud system configuration.

3 Architecture

3.1 Design

Our proposed Nemesis architecture design principle is illustrated in Figure 2. Threat models and vulnerability assessment framework are the main pillars used to build our architecture. We have devised two implementations of our pillars toward assessing cloud computing assets. To filter a rich amount of security analytics auto generated by our two implementations (Threat Probability Estimator and Threat Classification), we built four lightweight middleware applications (Risk Estimator, Severity Ranking Engine, Exploitable Vulnerabilities Generator and Suggested Configurations Generator). These applications per any given cloud configuration produce an aggregated risk indicator, threat types severity ranks, exploitable vulnerabilities evaluations,

and suggested new configurations to reduce perceived risk.

In this section, we describe the incorporation of the models and framework, and their implementation into Nemesis. And present a high level view of our middleware applications algorithm designs.

3.1.1 Cloud Configurations

As we have discussed thus far, cloud computing design principle revolve around a set of shared building technologies and customization packages needed to instantiate and maintain various services. To unravel this complex design of any given cloud's assets, we need to extrapolate its configurations into a machine readable format for all foreseeable tasks. In our case study, the proposed architecture - Nemesis, will require the cloud's assets make configurations details for performing threat modeling and risk analysis tasks. The supported formats of cloud configurations details with our architecture are either contained within an ontology knowledge base file (generated from a customized defined ontology) or passed through a defined web facing aggregator portal or as on-fly inputs.

3.1.2 VULCAN Framework

VULCAN Framework [9] plays a big role within our architecture, as a source of security information used to generated various analytics. From VULCAN, we have utilized one of its architectural component called Ontology Knowledge Bases (OKBs). These OKBs contain rich information about various known Information Technology (IT) product's vulnerabilities, attacks and defenses modeled and represented using ontology design, with a powerful supporting semantic relationships.

Within Nemesis architecture, we devised an automatic approach to invoke VULCAN's OKBs for relevant vulnerabilities, attacks and defenses details on demand per each received cloud configuration entity . For each entity, we want to see if there exists:

- Any vulnerability. If so, we return that found vulnerability identifier, description, and severity ranking
- Any attack. If so, we return that found exploit identifier and description
- Any defense. If so, we return that found mitigation identifier and recommendation statement

3.1.3 STRIDE Threat Model

To address some type of questions regarding what are the vectors that can be used to threaten the given cloud's pillars, we used STRIDE threat model [1] to classify them into relevant threat types. More precisely, we devised an automated STRIDE's threat type classification approach to model after each discovered cloud configuration entity and its discovered vulnerabilities. The classification scheme, automate one of Microsoft threat modeling STRIDE's Elevation of Privilege (EoP) card game [14]; Where the EoP card game helps clarify the details of threat modeling and examines possible threats to software and computer systems.

Since this approach is usually done manually by challenging other developers to assign STRIDE threat types relevant to design architecture of the target system (In this case, Cloud System). Within our proposed architecture, we realize this via an implementation named "Threat Classification" used to automate this process as detailed via the unsupervised classification Algorithm 1.

Algorithm 1 Threat Classification

Data: Cloud Configuration's Entities
Result: Threat Types per each Entity - Vulnerability Pair
for *Entity in Cloud Configuration* **do**
  Invoke *Nemesis's Vulcan Framework* instance and pass the *Entity name*
  return Relevant found Vulnerabilities details
  for *Each Found Vulnerability* **do**
    Perform *A Similarity Test to all EoP Threat Types descriptions* call, with *Entity's Vulnerability Description* as a parameter
    return Threat Types's Similarity Scores
    Perform *A Classification Task*, with *Threat Types's Similarity Scores* as a parameter
    return Threat type that best suit the given Cloud's Entity - Vulnerability Pair
  end
end

3.1.4 Threat Probability Model

For the given cloud configurations, one approach to manage all threat types that could be found and generated via the "Threat Classification" approach is to rank them and provide an overall risk assessment indicator. We realize this by leveraging a threat probability model to estimate each "cloud's entity - threat type" threat probability value and amass them as weighted values toward generating an aggregated risk indicator via our developed risk estimator application.

Within our architecture, we evaluated Fenz's proposed model [13] on utilizing the security ontology for the Bayesian threat probability determination as shown in Figure 3 and found it to be a best match model to be included into our Nemesis. We then implemented this Threat Probability Determination model as "Threat Probability Estimator" as illustrate via Algorithm 2 using the Bayesian Network Structure and Variable's calculation equations as illustrated in Figure 3 with few extensions to accommodate our cloud's threat ranking needs. For instances, the variables equations are grouped into: Threat Variable, Intermediate Vulnerability Variable, Vulnerability Variable, Attacker Variable, Control Combination Variable and Control Variable and each one has some dependency to each other.

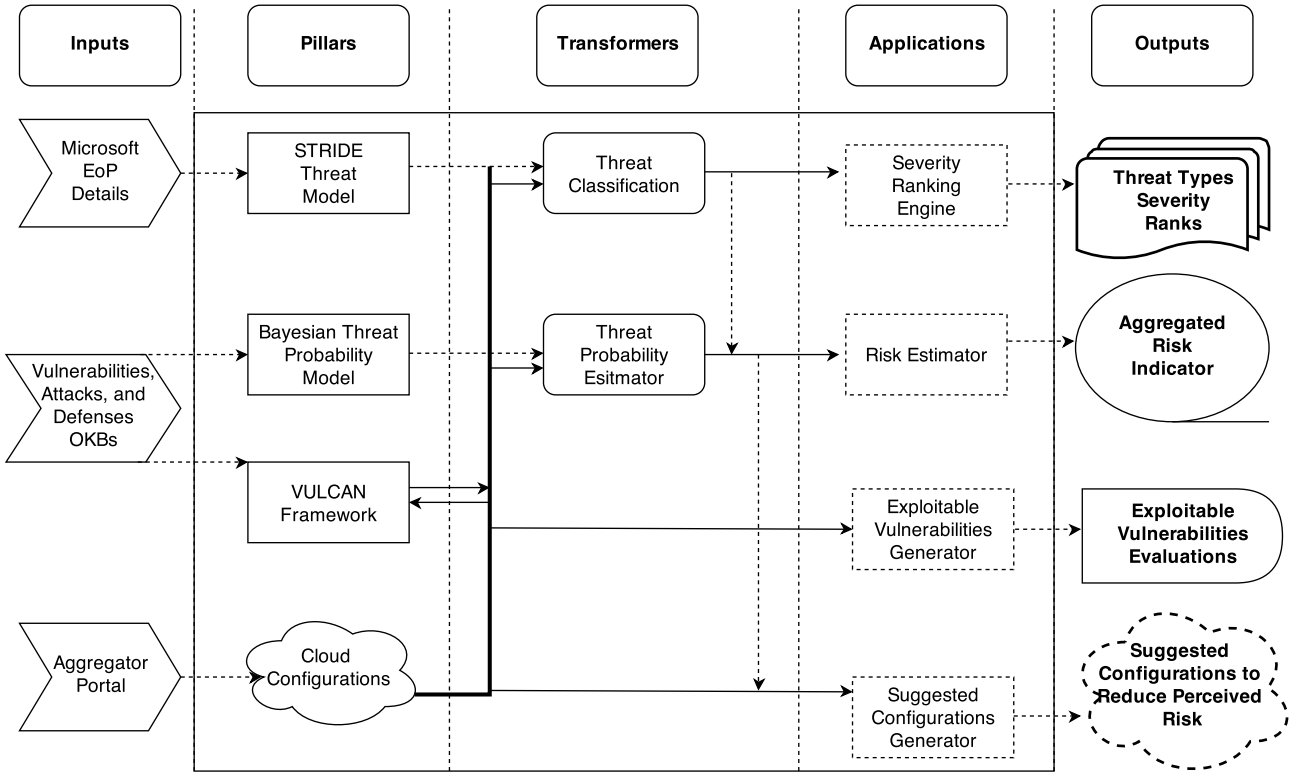

Figure 2: NEMESIS - Architecture

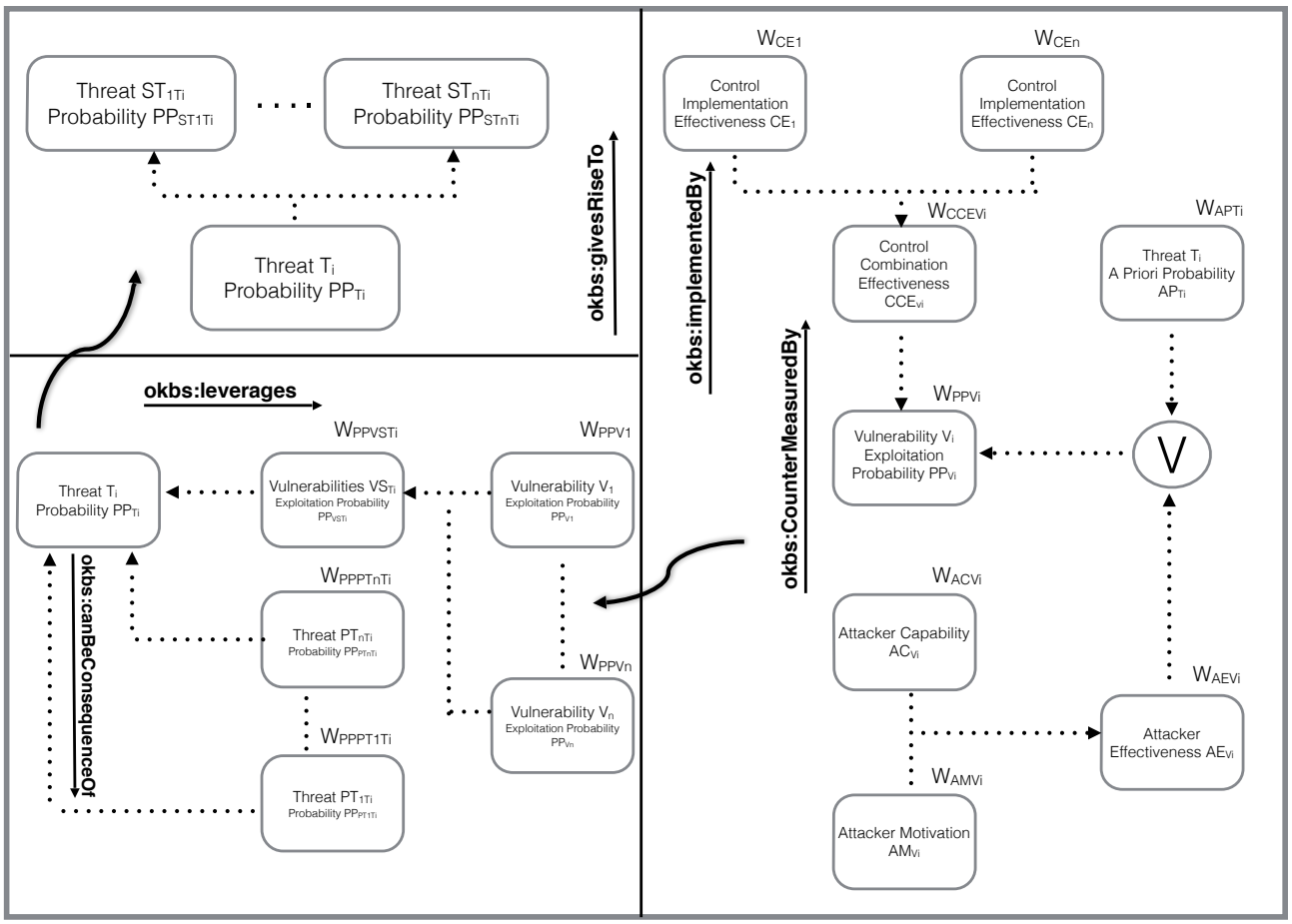

Figure 3: Utilizing OKBs for the Bayesian threat probability determination

Algorithm 2 Threat Probability Estimator

Data: Cloud Configuration's Entities and OKBs graphs
Result: A list of Threat Probabilities Values for all given Cloud's entities
for *Entity in Cloud Configuration* **do**
  Invoke *Nemesis's Vulcan Framework* instance and pass the *Entity name and relevant OKB graph*
  return Relevant found Vulnerabilities identifiers
  for *Each Vulnerability identifier* **do**
    Invoke *controlVariable* subroutine, with *This Vulnerability identifier and relevant OKB graph* as a parameter
    return A qualitative scale of this control for the given vulnerability, its description and a binary value of wether this control is active
    Invoke *ControlCombVariable* subroutine, with *controlVariable outputs and other variant's similar data of This Vulnerability Identifier* as parameters
    return A qualitative scale and its quantitative value
    Invoke *AttackerVariable* subroutine, with *This Vulnerability identifier and relevant OKB graph* as parameters
    return A qualitative scale and found Exploit Description
    Invoke *vulnerabilityVariable* subroutine, with *ControlCombVariable and AttackerVariable outputs* as parameters
    return A quantitative scale
    Append each Vulnerability identifier's vulnerabilityVariable output into a list
  end
  Invoke *intermediateVulnerabilityVariable* subroutine, with *A list of vulnerabilityVariable outputs* as parameters
  return A quantitative scale
  Invoke *ThreatVariable* subroutine, with *intermediateVulnerabilityVariable outputs and a list of aPriori Threat Probabilities Values* as parameters
  return A quantitative scale
  Append each Vulnerability identifier's ThreatVariable output into a list
end

3.1.5 Nemesis's Lightweight Applications

- Risk Estimator application high level design is detailed via Algorithm 3
- Severity Ranking Engine application high level design is detailed via Algorithm 4
- Exploitable Vulnerabilities Generator application high level design is detailed via Algorithm 5 and
- Suggested Configurations Generator application high level design is detailed via Algorithm 6

Algorithm 3 Risk Estimator

Data: A list of Threat Probabilities Values for all given Cloud's entities and their weights
Result: Aggregated Risk Indicator
for *Input Data* **do**
  Invoke *an Aggregator* subroutine and pass the *Input Data* as parameters
  return Aggregated Quantitative Scale
end

Algorithm 4 Severity Ranking Engine

Data: Cloud Configuration's Entities and OKBs graphs
Result: Threat Types Severity Ranks
for *Entity in Cloud Configuration* **do**
  Invoke *Nemesis's Vulcan Framework* instance and pass the *Entity name and relevant OKB graph*
  return Relevant found Vulnerabilities identifiers
  for *Each Vulnerability identifier* **do**
    Invoke *Threat Classification* module, with *This Vulnerability identifier and relevant OKB graph* as a parameter
    return Perceived Entity - Vulnerability's Threat Type
    Invoke *SeverityScore* subroutine, with *his Vulnerability Identifier* as parameters
    return A quantitative severity score
    Append the found severity scores per each vulnerability into a list of threat type classes lists
  end
end
for *All lists of threat type classes's severity scores lists* **do**
  Compute a new list of threat type classes's average severity scores and return it
end

Algorithm 5 Exploitable Vulnerabilities Generator

Data: Cloud Configuration's Entities and OKBs graphs
Result: Exploitable Vulnerabilities Evaluations
for *Input Data* **do**
  Invoke *Nemesis's Vulcan Framework* instance and pass the *Entity name and Vulnerability OKB graph*
  return Counts of relevant found vulnerabilities, their identifiers and Active/Passive state binary value
  for *Each vulnerability identifiers* **do**
    Invoke *Nemesis's Vulcan Framework* instance and pass the *Entity name and Attack OKB graph*
    return Count of found Exploits
    Append this Exploits Count into a list
  end
end
for *All counted list's values* **do**
  Compute their sum with respect to their targeted vulnerabilities and return the Exploitable Vulnerabilities Evaluations
end

Algorithm 6 Suggested Configurations Generator

Data: Cloud Configuration's Entities and OKBs graphs
Result: Suggested Configurations to Reduce Perceived Risk

```
for Input Data do
    Invoke Nemesis's Threat Probability and Risk Estimator module and pass the Input Data
    return Aggregated Risk Indicator for this Cloud Configuration Profile and store this result for future comparison
    Invoke An Evaluator module and pass the Current Cloud Configuration Profile
    return All other alternative Cloud Configuration Profiles made of pre and post releases of the first Profile's entities
    for Each generated Cloud's Profile do
        Compute its Aggregated Risk Indicator and store it
    end
    Perform A systematic ranking of each Cloud Configuration Profile and its Risk Indicator
    return The best optimal profiles with a lower risk indicator and suggest them back
end
```

3.2 Implementation

We have implemented our proposed Nemesis design architecture on:

- On a MacBook Pro laptop with a 2.3 GHz - Intel Core i7 Processor and 16 GB of RAM
- With Vulnerabilities, Attacks, and Defenses Ontology Knowledge Bases populated with approximately 100,000 instances
- Nemesis - prototype is automated using Java programs (with 700 LOC for the automation of ontology knowledge bases generation and indexing for fast processing) and Python scripts (with 1700 LOC) with supporting libraries such as: NLTK [15], RDFLib [16], Protege Editor's OWL libraries [17]

4 Experiments and Evaluations

4.1 Experiments

To validate and evaluate our proposed architecture - Nemesis, we have designed a cloud environment and deployed its services using OpenStack [2] cloud operating system. For our OpenStack deployment, we used DevStack [18] and deployed one of OpenStack version [19].

Our simple OpenStack deployment system is live with nova-compute, cinderv2-volumev2, novav3-computev3, s3-s3, glance-image, heat-cloudformation, cinder-volume, ec2-ec2, heat-orchestration and keystone-identity services to power Compute and Orchestration OpenStack's projects. For example, we can leverage Glance and Nova services, If we want to deploy any flavored instance running on Ubuntu, or Fedora, or Centos, or RedHat, or Windows operating system. And for various cloud applications automatic design and production, we can leverage Heat service which in turns relies on other services.

On a Dell PowerEdge T620 server with 24 cores and 64 GB of RAM and 2T of storage - running on Ubuntu:* and using VirtualBox:* [20], we created an Ubuntu:* virtual machine for OpenStack-DevStack single node deployment powered with 10 CPUs, 40 GB of RAM and 400 GB of storage. Upon OpenStack successful deployment, QEMU:* hypervisor is used to support the Compute Service.

Via OpenStack dashboard, we are able to instantiate a number of instances such as: *Ubuntu:*, Fedora:*, and Centos:** ones via Compute project and some stacks via Orchestration project using open-source based heat-templates [21] such as: *Word Press Native*, *Chef Server* and *Nova Instance With Cinder Volume Native*.

A sample look of some IT products (* taken randomly from a pool of various releases of products that meet our experiment criteria) that are used to support our cloud infrastructure and services are shown in Table 2 which are in turn used to evaluate our proposed Nemesis architecture.

IT Products	Description
Ubuntu:12.04	- Host and Guest VM OS
VirtualBox:3.2	- Virtualization Product
Grizzly 2013.1.1	- Cloud OS
Ubuntu:12.10	- Cloud image OS
Fedora:17	- Cloud image OS
Centos:6	- Cloud image OS
WordPress:3.0.3	- Web Software
MySql:5.5.29	- RDBMS
RabbitMQ:3.3.5	- AMQP
Qemu:1.3.0	- Hypervisor

Table 2: Our Sample Cloud Configuration

4.2 Evaluations

In this section, we present our findings given the sample of cloud configuration shown in Table 2 that is feed into our Nemesis Architecture and analyzed via Nemesis's lightweight applications as presented in Section 3.1.5.

4.2.1 Risk Estimator application

The given cloud configuration shown in Table 2 is evaluated via this application, where it produces an aggregated risk estimated to *31.93%* of severity. This estimated risk can be left to various interpretations, but first it is a result of our Nemesis's design principle as described in Section 3 and the relevant data about found vulnerabilities, exploits and defenses are logged into an output file to support the estimated risk. Then, in the next subsections, we provide supporting additional useful metrics to illustrate the vulnerability analysis and threat level status of the given cloud configuration.

This application and the next two ones (Exploitable Vulnerabilities Generator and Severity Ranking Engine) had

an execution time that is within *3 minutes* window period; note that this time is dependent on the computing environment and the complexity of the applications Algorithms 3, 5 and 4.

4.2.2 Severity Ranking Engine application

For a threat focused security approach, the given cloud configuration shown in Table 2, is ranked based on this application design principle illustrated in Algorithm 4. For instance, Spoofing is the most eminent threat facing this cloud configuration, and Information Disclosure is the least.

Threat Types	**Severity Rank (0-10)**
Spoofing	4.01
Tampering	1.97
Repudiation	1.23
Information Disclosure	0.86
Denial of Service	3.28
Elevation of Privilege	1.13

Table 3: Threat Types - Severity Rank Evaluations

4.2.3 Exploitable Vulnerabilities Generator application

For the given cloud configuration shown in Table 2 feed into this application, it produces the exploitable vulnerabilities metric for each cloud configuration entity as illustrated in Table 4. In this case, two cloud entities stand out where Fedora:17 has a value of 1 and WordPress:3.0.3 has a value of 4; this call for an immediate attention to fix them since that for all found vulnerabilities, there exist some known attacks ready to exploit them. And for other entities with a metric value of 0, though there was not a known attack found, still the count of vulnerabilities discovered require an attention and proper mitigation plan (if it is available, our Nemesis's will log it into the output file of relevant data).

IT Product	**Vulnerabilities Count**	**Exploitable Vulnerabilities Metric**
Ubuntu:12.04	97	0
Virtualbox:3.2	2	0
Grizzly:2013.1.1	1	0
Ubuntu:12.10	87	0
Fedora:17	10	1
Centos:6	7	0
Wordpress:3.0.3	32	4
MySql:5.5.29	48	0
Rabbitmq:3.3.5	0	0
Qemu:1.3.0	8	0

Table 4: Exploitable Vulnerabilities Metric Evaluations

4.2.4 Suggested Configurations Generator application

The new generated alternative recommended cloud configuration shown in Table 5 is evaluated via the Risk Estimator application, and an aggregated risk was estimated to *25.88%* of severity. Compared to the original given cloud configuration shown in Table 2, there is a *6.05%* perceived estimated risk reduction. Each recommended cloud configuration entity was selected from a pool of its variant versions by picking up the perceived higher version with a lower aggregated risk compared to the original given cloud's entity.

IT Products	**Pre- and Post-Releases Count**
Ubuntu:13.04	22
Virtualbox:4.0.20	75
Grizzly:2013.1.4	8
Ubuntu:13.04	22
Fedora:18	15
Centos:6	1
Wordpress:3.7	121
MySql:6.0.10-bzr	366
Rabbitmq:3.3.5	0
Qemu:2.0.0	78

Table 5: Alternative Recommended Cloud Configuration

4.2.5 Challenges

Our proposed Nemesis's architecture design principle is built on proven models and frameworks and its delivery outputs are beneficial to security community who want to assess their cloud's or organization IT's assets in terms of threat modeling and risk assessment. And within our experiments we were able to proof and reaffirm our claims and discover some limitations. Some of the main limitations of our proposed automated architecture are:

- The open-source vulnerabilities, attacks and defenses data feeds are limited, where all discovered vulnerabilities for instances are not all publicly known, and even if some of it are known, they are not properly formatted and or organized together for various automated tasks.
- For all data feeds that are available, for example in our experiments we were able to gather and use over 100,000 data feed instances, and we have seen a high disproportion of how many known vulnerabilities vice their countermeasures.
- And for our recommendation of alternative cloud configuration with a lower perceived risk approach, we have seen that the convention of versioning various IT product is not standardized throughout different product's vendors and this hinder our reach of accessing all available pool of versions for a better recommendation.

5 Related Work

5.1 Security Assessment of Cloud Computing

Ristov et al. [22] work on security assessment of OpenStack is reflected in our work. Here, they assessment the Open-

Stack Node and its instances, using a third party tool and its configuration to discover what are the known present vulnerabilities and reported their results. In our approach, we consider additional cloud services's configurations for assessment as well via Nemesis architecture; as we have presented thus far in Section 3, via Nemesis we achieve complete vulnerability assessment, threat modeling and risk assessment of the given cloud's assets automatically.

Donevski et al. [23] work on analyzing virtual machine security threats that might occur by other tenants or outside the cloud; revolved around discovering new vulnerabilities for different networking configurations with the cloud network controller using a third party security assessing tool. Though this work does not perform any detailed threat modeling per se, but they point out a critical threat variant of new vulnerabilities that comes from the composition of shared technologies within a cloud setting. In our proposed architecture, we emphasis how this new vulnerabilities come in the picture and how we perform a detailed threat modeling for each found vulnerabilities and how all found vulnerabilities impact the threat level of any given cloud setting.

Amartya et al. [24] work on "Off-line Risk Assessment of Cloud Service Provider" have in common with our proposed architecture by their methodology of assessing threats present in a client's application/cloud service using Stride model [1]. Our approach automate the threat modeling methodology of using EoP card game [14] based STRIDE model, instead of manually generating threat types as traditionally been done like in Amartya et al. [24] approach, in addition our risk assessment approach within Nemesis architecture is automatic.

5.2 Ontological Approach Toward Security Analysis of Systems/Cloud Computing

In Fenz [13], [25] and Settas et al. [26] works have two main components in common with our proposed Nemesis architecture such as: The use of Ontological approach and Bayesian Networks key building blocks towards various security assessment works. Additional ontological security related works by Fenz et al. that motivated us to build our Nemesis's Risk assessment model are:

- On Ontology-based Generation of IT-Security Metrics [27], Here we use various measurements generated from Nemesis's Vulcan and Threat modeling subroutines to compute and produce useful metrics regarding the status of the given cloud's assets such as: "Threat type's severity average score" metric, and an additional metric that we adapted from Walter [28] webinar presentation's geared for the cloud setting is the "Unpatched critical vulnerabilities" for the given Cloud metric.
- On Ontology- and Bayesian-based Threat Probability Determination [29], which we modified for cloud setting and automated into our Nemesis's threat ranking subroutine; and
- On Security Ontology: Simulating Threats to Corporate Assets [30], here we performed a thorough threat modeling task and generated all possible threat types for all discovered vulnerabilities of the given cloud's assets.

In Kamongi et al. works [9] and [31], leverage the use of ontological approach towards the design of a vulnerability assessment framework - VULCAN and its incorporations into the ranking methodology of Cloud System's Vulnerabilities and our proposed Nemesis's architecture.

These excellent contribution works show how ontology back-bone is critical when one is attempting to model any security related study, and enables us to use any known predictive model like Bayesian Networks in parallel or as an add-on.

6 Conclusion and Future Work

6.1 Conclusion

In this paper, we have proposed and implemented a prototype of Nemesis, our novel automated risk assessment architecture for Cloud. Our proposed architecture design principle revolve around collecting measurements about what type of known vulnerabilities exist for the given Cloud's assets; how they can be exploited and how they can be mitigated; then use them toward the generation of a set of customized outputs such as: Aggregated risk estimated value, Exploitable vulnerabilities metric evaluations, Threat types - severity rank evaluations, New recommended Cloud Configurations with a lower perceived risk along with a detailed summary of relevant nemesis evaluation data.

6.2 Future Work

In our studies with vulnerability assessment of various technologies, we have been focusing on what's known to be the weakness that leads to exploitation. Also what are the new weaknesses that come from a composition of technologies with inherent weaknesses. As we move ahead with the security breaches trend, we see that this set is limited to properly assess new weaknesses/vulnerabilities categorized as "Zero-day" [32] ones. These Zero-day's vulnerabilities are hard to detect since, in most case there have never been a predecessor or known variant of its form.

One approach to keep up with Zero-day vulnerabilities is to keep updated the security repositories like vulnerability ontology knowledge base with latest feed from initiatives like the newly announced Google Project Zero [33] which is trying to preempt and discover zero-day type of vulnerabilities and publish them. Within our work, we are leveraging security intelligences that come from collected analytics of any system, for instance cloud computing resources management and provisioning intelligences. Within this scope, we can dynamically assess any given cloud environment and be able to detect and prevent new zero-day type of weaknesses.

References

[1] Microsoft Corporation. The stride threat model, 2014. URL `http://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx`.

[2] OpenStack project. Openstack: The open source cloud operating system, 2014. URL https://www.openstack.org/software/.

[3] IBM Cloud. What is cloud?, 2014. URL http://www.ibm.com/cloud-computing/us/en/what-is-cloud-computing.html.

[4] AlertLogic. Cloud security report honeypot findings - 2014, September 2014. URL http://alertlogic.com/wp-content/uploads/2014/08/alertlogic-HoneypotFindings2014-infographic.pdf.

[5] Cloud Security Alliance. The notorious nine - cloud computing top threats in 2013, 2014. URL https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf.

[6] Electronic Frontier Foundation. Threats, surveillance self-defense, 2014. URL https://ssd.eff.org/risk/threats.

[7] NIST. National vulnerability database, 2014. URL http://nvd.nist.gov/.

[8] Offensive Security 2014. The exploit database, 2014. URL http://www.exploit-db.com/.

[9] Patrick Kamongi, Srujan Kotikela, Krishna Kavi, Mahadevan Gomathisankaran, and Anoop Singhal. Vulcan: Vulnerability assessment framework for cloud computing. In *Software Security and Reliability (SERE), 2013 IEEE 7th International Conference on*, pages 218–226. IEEE, 2013.

[10] Canonical Ltd. Ubuntu for cloud, 2014. URL http://www.ubuntu.com/cloud.

[11] Microsoft Corporation. Identifying techniques that mitigate threats, 2014. URL http://msdn.microsoft.com/en-us/library/ee798428(v=cs.20).aspx.

[12] Wikipedia. Dread: Risk assessment model, 2014. URL http://en.wikipedia.org/wiki/DREAD:_Risk_assessment_model.

[13] Stefan Fenz. An ontology-and bayesian-based approach for determining threat probabilities. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 344–354. ACM, 2011.

[14] Microsoft Corporation. Microsoft eop game, 2014. URL http://www.microsoft.com/security/sdl/adopt/eop.aspx.

[15] NLTK Project. Natural language toolkit, 2014. URL http://www.nltk.org/.

[16] RDFLib Project. Rdflib library, 2014. URL https://github.com/RDFLib/rdflib.

[17] Stanford Center for Biomedical Informatics Research. Protege editor, 2014. URL http://protege.stanford.edu/.

[18] Openstack Foundation. Devstack, August 2014. URL http://devstack.org/.

[19] Openstack Foundation. Openstack releases, August 2014. URL https://wiki.openstack.org/wiki/Releases.

[20] Oracle Inc. https://www.virtualbox.org/, August 2014. URL https://www.virtualbox.org/.

[21] OpenStack project's heat Developers. Heat templates, August 2014. URL https://github.com/openstack/heat-templates.

[22] Sasko Ristov, Marjan Gusev, and Aleksandar Donevski. Openstack cloud security vulnerabilities from inside and outside. In *CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 101–107, 2013.

[23] Aleksandar Donevski, Sasko Ristov, and Marjan Gusev. Security assessment of virtual machines in open source clouds. In *Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on*, pages 1094–1099. IEEE, 2013.

[24] Amartya Sen and Sanjay Madria. Off-line risk assessment of cloud service provider. In *International Workshop on Cloud Security Auditing (CSA 2014)*. IEEE, 2014.

[25] Stefan Fenz. An ontology-based approach for constructing bayesian networks. In *Data & Knowledge Engineering*, volume 73, pages 73–88. Elsevier, 2012.

[26] Dimitrios Settas, Antonio Cerone, and Stefan Fenz. Enhancing ontology-based antipattern detection using bayesian networks. In *Expert Systems with Applications*, volume 39, pages 9041–9053, 8 2012.

[27] Stefan Fenz. Ontology-based generation of IT-security metrics. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1833–1839. ACM, 1 2010.

[28] Williams Walter. Information security metrics, 2014. URL https://www.brighttalk.com/webcast/574/108369.

[29] A Min Tjoa and Stefan Fenz. Ontology- and bayesian-based threat probability determination. In *Proceedings of the Junior Scientist Conference 2008*, pages 69–70. Vienna University of Technology, 11 2008.

[30] Stefan Fenz, Edgar R. Weippl, Markus Klemen, and Andreas Ekelhart. Security ontology: Simulating threats to corporate assets. In *Information Systems Security, Second International Conference, ICISS 2006*, volume $4332_2006, pages 249 - -259, 122006$.

[31] Patrick Kamongi, Srujan Kotikela, Mahadevan Gomathisankaran, and Krishna Kavi. A methodology for ranking cloud system vulnerabilities. In *Proceedings of The Fourth International Conference on Computing, Communication and Networking Technologies (ICCCNT'13)*, 2013.

[32] Wikipedia. Zero-day attack, 2014. URL http://en.wikipedia.org/wiki/Zero-day_attack.

[33] Google Inc. Announcing project zero, 2014. URL http://googleonlinesecurity.blogspot.com/2014/07/announcing-project-zero.html.