2021

# A Static Code Analysis And Pattern Recognition Algorithm-Driven, Quantitative, Mathematical Model-Oriented Risk Assessment Framework Of Cloud-Based Health Information Applications

Dennis B. Park
*Indiana State University*

# A STATIC CODE ANALYSIS AND PATTERN RECOGNITION ALGORITHM-DRIVEN, QUANTITATIVE, MATHEMATICAL MODEL-ORIENTED RISK ASSESSMENT FRAMEWORK OF CLOUD-BASED HEALTH INFORMATION APPLICATIONS

---

A Dissertation

Presented to

The College of Graduate and Professional Studies

College of Technology

Indiana State University

Terre Haute, Indiana

---

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy in Technology Management

---

by

Dennis B. Park

July 2021



Keywords: Technology Management, Risk Assessment Framework, Quantitative Risk Calculation, Health Information Applications in Cloud, Data Collection from Tools

# VITA

DENNIS B. PARK

## EDUCATION

Indiana State University, Doctor of Philosophy in Technology Management, 2021

The University of Virginia, Master of Science in Management of Information Technology, 2004

The University of Maryland, Bachelor of Science in Computer Science, 1991

## TEACHING EXPERIENCE

Faculty, 7/2014-6/2019, Harbin Institute of Technology, PRC

Faculty, 6/2014-7/2014, Northeast University, Software College, PRC

Instructor, 9/2012-6/2014, Yanbian University of Science and Technology, PRC

Professor, 1/2012-8/2012, Washington University of Virginia, US

## PROFESSIONAL EXPERIENCE

Various Roles, 1994-2012, Hewlett Packard - Enterprise / Solution Architect; SW Dev Manager

Systems Analyst, 1991-1993, Potomac Systems Engineering

Senior Programmer, 1991-1991, Columbia Research Corporation

COMMITTEE MEMBERS

Committee Chair: Xiaolong Li, Ph.D.

Professor, Department of Electronics, and Computer Engineering Technology

Indiana State University

Committee Member: A. Mehran Shahhosseini, Ph.D.

Professor, College of Technology

Indiana State University

Committee Member: Li-Shiang Tsay, Ph.D.

Associate Professor, Department of Computer Systems Technology

North Carolina A&T State University

## ABSTRACT

According to a survey, the healthcare industry is one of the least cloud-adopting industries. The low adoption reflects the healthcare industry's ongoing concerns about the security of the cloud. Business applications, according to another survey, are among the most vulnerable components of business information systems. Many risk assessment frameworks available today, particularly for health information applications, require significant customization before they can be used. This study created a new framework to assess cloud risks specifically for their health information applications, utilizing data-driven risk assessment methodologies to avoid surveys, interviews, and meetings for data collection. For the feasibility study, the open-source application codes were chosen from over 190 million GitHub repositories using a decision tree method, while a purposive sampling method was used to choose for a simulated patient information database from the healthcare industry. Using these methods, the researcher discovered security warnings and privacy violation suspects and subsequently converted them into quantitative measures to calculate the risks of the cloud-based health information application and a database. The significance of this study lies in the collection of data directly from applications and databases with a quantitative approach for risk calculation.


*Keywords*: Technology Management, Risk Assessment Framework, Quantitative Risk Calculation, Health Information Applications in Cloud, Data Collection from Tools

# PREFACE

I have been involved in health informatics for a long time, but the experience mostly stems from government works such as the Military Health System (MHS) from the Department of Defense (DoD) and Veterans Health Administration (VHA) from the Department of Veterans Affairs (DVA). Each organization has a flagship health care application for active duties and veterans. DVA has developed Veterans Health Information Systems and Technology Architecture (VistA) and DoD has modified it into the Composite Health Care System (CHCS).

At the turn of the century, we have seen the dramatic change of the computing landscape from a client-server model to a cloud-based model. I was drawn to this dramatic change from the perspective of health informatics. It has been very intriguing to see how this new model affects health informatics. Soon, I found that the concerns on information security and privacy are the major obstacles of health informatics in their adoption of cloud-based systems. A standard health organization like Health Level 7 (HL7) does have a workgroup for security and privacy, but the group has more focused on messaging, not so much on cloud computing. Thus, I always wanted to investigate how I can help and improve the adoption rate of cloud computing in health informatics.

# ACKNOWLEDGMENTS

Many unanticipated events had occurred since I joined this Ph.D. consortium program, making the road ahead difficult, including several changes on advisors and my wife's illness. However, with God's unfailing grace, I was able to overcome the difficulties all along. Praise to the Lord! It would also not have been possible without the assistance of my committee advisors: Dr. Li, Dr. Shahhosseini, and Dr. Tsay. Their willingness to stick by me, as well as their advice and guidance, enabled me to reach this conclusion. My sincerest gratitude and appreciation to you all.

Finally, it is difficult to put into words what my wife has gone through during all those years of sacrifices, including times when I was not physically, emotionally, or spiritually present with her. I am eternally grateful to her for her love, encouragement, and endurance, both in the past and now.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER 1

## INTRODUCTION

The world is in a state of constant change. It is even more so in the world of business. Matai (2011) argues that business leaders, therefore, ought to cope with the change for their survival. The status quo is not accepted as the slow response to the change loses many business opportunities. Time is of the essence. To embrace the change, business leaders understand the drivers of the changes to position themselves appropriately. The drivers of the changes may vary based on the type of industrial age consumers live in. For example, the globalization of markets and rapidly evolving technology are the two main drivers of the change today (Lesser et al., 2016). These drivers force businesses to deal with and demand them to respond to the change. Inadvertently, these changes bring many intrinsic uncertainties to organizations with seminal consequences, which could be positive (opportunities) or negative (risks). As such, organizations are in a situation where they ought to adjust to these uncertainties, especially if the uncertainties turn into risks that might potentially cripple the organizations in some form of financial loss and undesired reputation (Donnan & Leatherby, 2019).

One of the latest changes that bring huge impacts to business is cloud computing. The advent of cloud computing has completely changed the landscape of the business world. Raffaelli (2020). illustrates such a phenomenon by contrasting how online companies have emerged and how brick-and-mortar companies have faded away, benefiting from cloud

computing. For example, Amazon has accelerated into the online market, leaving Borders Books bankrupt. As for Google, according to Ingram (2013), its revenue soared to $46 billion while the total newspaper industry's revenue dropped to a little over $20 billion between 2009 and 2013, making numerous newspapers scramble for survival or fold altogether.

The National Institute of Science and Technology (NIST) provides a solid definition of cloud computing in its NIST SP 800-145 (Mell & Grance, 2011). It states, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access with pay per usage to a shared pool of configurable computing resources (i.e., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Therefore, the cloud offers very attractive, and yet financially practical computing power and storage resources to consumers as it enables them to avoid the up-front major investments, resources to install and support, and a long-time commitment to the purchased hardware and software. These computing power and storage resources can rapidly be procured as "services" with minimal financial commitments and with fewer concerns about the purchases of various licenses and installation works (Mell & Grance, 2011). Due to the benefits highlighted above, many industries have embraced cloud computing as an alternative way of conducting business.

However, embracing the cloud is not always rosy. Terdoslavich (2016) reports a recently conducted survey of 1,114 senior IT executives, representing companies ranging from $50 million to more than $2 billion in annual sales. More than 80% of the respondents to the survey said they plan to store data in "new technology environments," like the cloud, big data, or the Internet of Things (IoT). Of those, however, the majority (85%) said they are "concerned" or "very concerned" about security in the cloud. When it comes to a specific industry like

healthcare, it is no different. Information assurance is a huge challenge in healthcare services for their adoption of cloud computing. Cloud Standards Customer Council (2017) reports that despite the significant advantages of the utilization of cloud computing as part of Healthcare IT (HIT), security, privacy, reliability, integration, and data portability are significant challenges and barriers to its adoption and implementation. The survey of TATA Consultancy Services (TCS, 2012, Chapter II) reviewed global industries for the average number of cloud applications per industry and published a report of the outcome in 2011. In the report, the industry of healthcare services scored at 3.39 on the scale of 0 to 9, which reflects the industry as one of the least cloud-adopted industries. A projected figure for 2014 (TCS, 2012, Chapter VII) does not show much difference other than the healthcare services industry has improved slightly over industries like chemicals and media/entertainment/sports. See Figure 1 for the 2012 report.

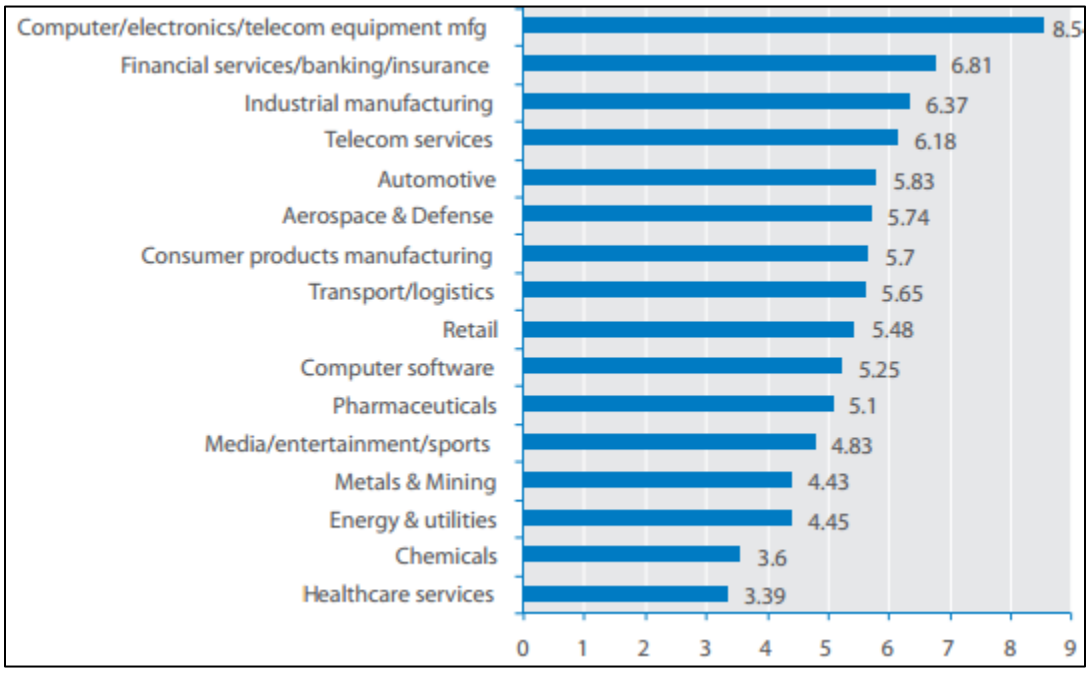



Figure 1. The average number of cloud apps per business functions in 2011

Such a low or reluctant adoption reflects the healthcare industry's ongoing concerns about the cloud's security. The recent data breach incidents of Target and Neiman Marcus

(Bjorhus, 2014) and the latest data breach settlements by two of the nation's biggest healthcare systems in Advocate Health Care Network (Mangan, 2016) and Anthem (Armerding, 2018; De Groot, 2020) have only added more to the concerns. Not only that, but the trend of data breaches is also disturbingly increasing as shown below (Statista, 2017).

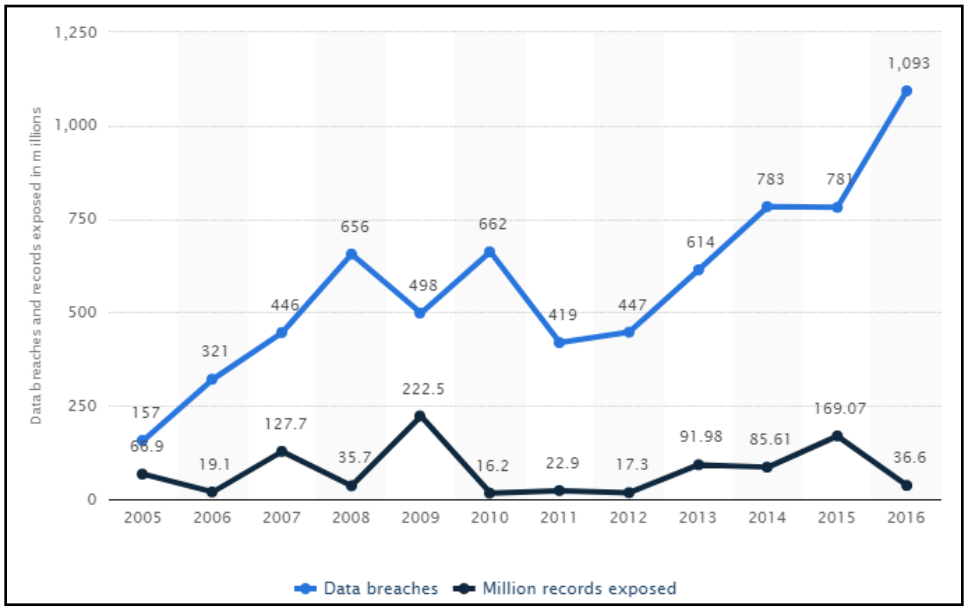



Figure 2. The annual number of data breaches in the U.S. from 2005 to 2016

As part of efforts to help industries deal with information risks in IT in general, many organizations and institutes have developed risk management frameworks. In the area of cloud computing, the National Institute of Standards and Technology (NIST) have finalized the definition of cloud computing in the SP 800-145 in September 2011 (Mell & Grance, 2011). NIST also published the SP 800-30 Revision 1 Guide for Conducting Risk Assessments in September 2012 (National Institute of Standards and Technology, 2012). In non-governmental environments, the recognition of these cloud trends from the information security community ultimately ended up founding an organization like Cloud Security Alliance (CSA) in December 2008. CSA runs CSA Security Trust Assurance and Risk (STAR) and Cloud Control Matrix (CCM) (Tariq, 2018). STAR validates cloud service provider's cloud security based on CCM.

Concerning U.S. health information systems, HIMSS has produced several security toolkits for cloud computing, such as *Sample Risk Assessment for Cloud Computing in Healthcare* (HIMSS, 2020) and *Cloud Security WorkGroup Toolkit* (HIMSS, 2012). The toolkits are guides on how to comply with HIPAA's privacy and security when using the cloud.

## The Research Problem

HIPAA stands for Health Insurance Portability and Accountability Act of 1996. HIPAA laws apply to covered entities and business associates regarding the privacy, security, and accessibility of electronic protected health information (ePHI). In HIPAA terms, covered entities are health care providers, health information clearinghouses, and health plans, which electronically transmit any health information in connection to transactions for which HHS has adopted standards (KirkPatrickPrice, 2020; HHS, 2018). In compliance with HIPAA, the covered entities are concerned with the security and privacy of patients' electronic health records, especially when it comes to cloud adoption (Alder, 2019).

Many experts propose risk assessment and management methods that can gauge security and privacy risks. Those methods adopt the qualitative approach predominantly, and the approach inherently requires many person-hours to conduct a risk assessment. More specifically, if a risk assessor uses the ISO 27001 approach, it may take 8 days to 40 days per 10 assets depending on tools used (Biscoe, 2019). If each asset may have at least 10 threats and each threat may have at least five vulnerabilities, then for small companies with 50 assets they may have 50 x 10 x 5 = 2,500 risks, and it would take 40 days to 200 days to assess the risks (Kosutic, n.d.). As attackers can advance their attacks anytime anywhere without warnings, the lack of timely assessment may lead covered entities to compromise results and consequently possibly jeopardize ePHI in the cloud.

## Research Questions

This research study aims to tackle the stated problem by proposing a new risk assessment method; that is, using a data-oriented quantitative mathematical model-driven risk assessment method that is very specific to health information applications in the cloud. The more specific research questions follow:

RQ-1: What development strategy can yield such a risk assessment framework that concerns specifically cloud-based health information applications and databases?

RQ-2: What methods can transform the qualitative measures generated from software tools such as security warnings and privacy violation suspects into numeric measures for risk calculation?

RQ-3: How can the adopted mathematical models incorporate severities and vulnerabilities for security warnings separately from privacy violation suspect into formulas for risk evaluation?

RQ-4: Is the proposed risk assessment framework feasible?

In association with the above research questions, the author has made the following objectives for this research study.

## The Objectives of the Research Study

The objectives of the research study are:

1) To construct a quantitative risk assessment framework to address the risks of cloud-based health information applications.
2) To avoid surveys, interviews, or meetings for risk data collection
3) To implement the new framework as a feasibility study with open-source health information application and simulated database

## Research Study Organization

This research study is organized as follows to investigate the research questions and achieve the objectives:

Chapter 1:

- Complete the research questions, objectives, and plan.

Chapter 2:

- Complete the literature review for limitations of current qualitative risk assessment frameworks and justify the necessity of constructing a new one that uses a quantitative framework that is specific to health information applications.

Chapter 3:

- Construct a quantitative risk assessment framework that deals with a cloud-based health information application.
- Construct selection criteria to settle on an open-source health information application from many application code repositories and choose a simulated database that can be applied by the new framework.
- Construct an implementation environment for the new framework and execute the framework implementation plan for its feasibility study.

Chapter 4:

- Analyze the implementation results and findings on the new risk assessment framework using software tools and algorithms that support the framework.

Chapter 5:

- Summarize and conclude the research study.

## Significance of the Study

The new risk assessment framework constructed in this study pertains to the overarching goal of reducing security and privacy concerns promptly in the healthcare industry's cloud adoption. To accomplish the overarching goal, this study aims to construct a new framework that is quantitative, software tool-based, and specific to cloud-based health information applications and databases.

The shift from qualitative assessment to quantitative assessment provides the foundation of more repeatable and systematic capabilities and ultimately helps assess the risks in an automated fashion. The shift from data collection that relies on questionnaires, interviews, and surveys that are subjective to the one that utilizes software tool generated data provides the foundation of more data-oriented capabilities and more objective facts. The shift from a general high-level process method to a healthcare industry-specific process offers the opportunity to more focus on implementation-specific techniques.

## Assumptions

The selected cloud-based open-source applications operate in a cloud environment.

## Limitations

The interpretation of HIPAA-related laws and regulations is NOT professional.

The new risk assessment framework shall be limited to open-source health information applications and simulated textual databases.

This research study shall utilize a simulated environment in desktop PC to implement the new risk assessment framework.

This research study shall employ a few software tools to implement the new risk assessment framework. The tools scan source code for security issues and database for privacy issues.

Cloud-based health information application source code is limited to open-source Ruby

on Rails web applications.

The database is limited to a simulated textual database.

## Terminologies

Yun (2018) also clarifies several terms in dealing with his quantitative risk assessment model for personal information in smart cities as follows:

Threat - Illegal usage of personal information without authorization

Threat factor - Potential reason that gives rise to security events by exploiting weaknesses or loopholes of the system.

Threat behavior - Approaches/methods adopted by threat factor.

Threat frequency (or attack rate) - A relative ratio of the number of attacks to different vulnerabilities/loopholes within a certain period

Threat aftereffect (or aftereffect index) - The specific consequence generated by the utilization of personal information by threats.

Aftereffect value - Consequence measurement of existing or future risk events

Vulnerability (defects/loopholes) - Weakness that may lead to security events.

Sensitivity - The internal property of personal information giving rise to influences of risk events due to threats.

Confidentiality (no disclosure to an unauthorized visitor) - Anonymity degree

Risk value - Severity degree

Security demand - Security protection demand

Security degree (security index) - The objective degree to protecting personal information from being exploited by threats.

Preventive measure - Preventive and remedy behaviors made to protect personal information from loopholes of preventive systems to reduce the possibility of utilizing system vulnerability by reducing threats.

## Summary

Chapter 1 has covered the research topic, research questions, and objectives, and plan along with the significance of the study, assumptions, limitations, and terminologies.

# CHAPTER 2

## LITERATURE REVIEW

### EPHI Risks in Cloud Environment

To discuss the methods of assessing ePHI risks in the context of cloud computing, it would be prudent to discuss cloud computing first. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (i.e., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, four deployment models, and three service models ( Mell & Grance, 2011).

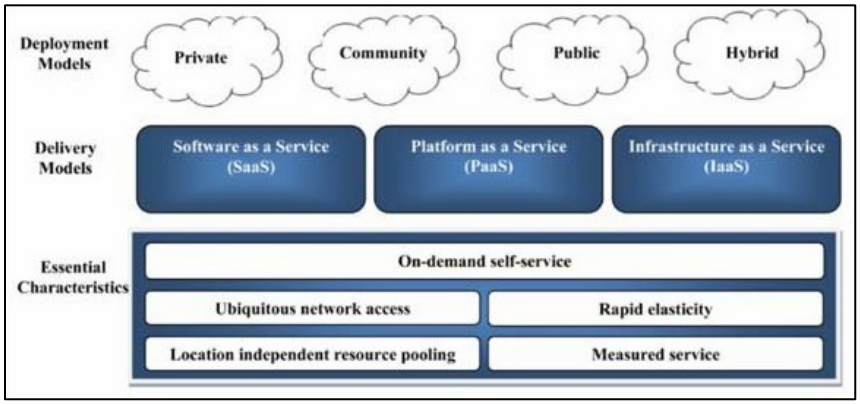



Figure 3. NIST model of cloud computing

Five essential characteristics are:

- On-demand self-service

- Ubiquitous network access
- Rapid elasticity
- Location independent resource pooling
- Measured service.

Four deployment models are:

- Private
- Community
- Public
- Hybrid

Three delivery or service modes are:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

In IaaS, the major concerns are minimal from an application perspective, as almost no changes are required for the existing code. The organizations can use the same authentication and security mechanisms from the original on-premise architecture, and they need to figure out their computing, storage, and network requirements. In PaaS, the organization's applications need to be re-factored to fit into PaaS, which may require code rewrites, library updates, and deployment process modifications (Violino, 2019). In SaaS, as cloud provider offers applications such as Google apps, the organizations need pre-adoption phases of work such as cloud service provider evaluation, subscription, and operations as the organizations turn IT operations over to cloud service providers (Zack & Kommalapati, 2011).

This research study focuses on assessing the risks of health information applications that organizations host or re-host into the cloud environment. Such a move involves many challenges and risks. The biggest one is security and privacy as HIPAA and HITECH laws regulate them.

## HIPAA, HITECH, and Electronic Health Records

The US Congress enacted the Health Insurance Portability and Accountability Act (HIPAA) in 1996, updated it with the privacy rule in 2003, and with the security rule in 2005, to provide a legal means of protecting electronic health records of patients. Developed by the Department of Health and Human Services (HHS), this law together with its subsequent law, the Health Information Technology for Economic and Clinical Health (HITECH), enacted in 2009, provides patients with more control over their personal electronic health information. HITECH also requires business entities to sign business associate agreements (BAAs) with other business entities as well as to provide patients or organizations with data breach notifications in case of security incidents (Office of Civil Rights, 2003). Figure 4 depicts the relationships of patients, covered entities, and business associates who are third-party organizations or persons. See Figure 4 (Skybox security, 2019).

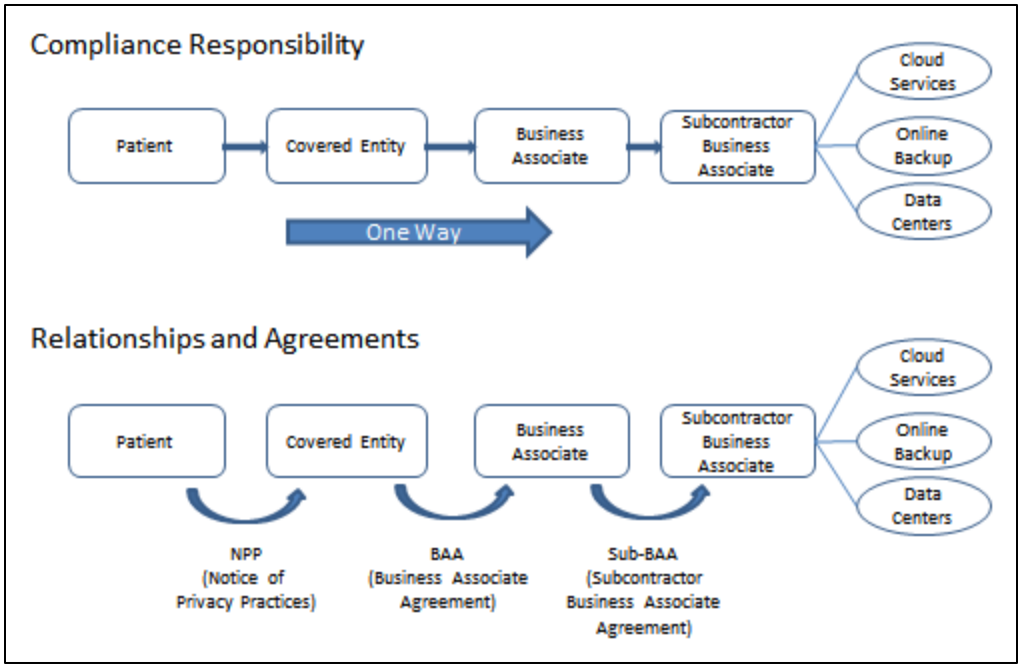



Figure 4. Relationships and agreements among HIPAA stakeholders

As the covered entities transmit and process a large volume of sensitive patient records, security becomes their primary concern (Brook, 2020). Also, data breach incidents have recently increased exponentially, and the incidents have heightened alertness about electronic patient health records in the covered entities (Davis, 2019). The advent of cloud computing complicates the matter greater. Although cloud computing has the benefits of on-demand IT resources acquisitions and operation transfers, it "also brings with it several concerns" which are "intensified due to HIPAA legislation that mandates data privacy and security provisions for safeguarding what is known as 'protected health information' or PHI" (Hold, 2019).

## Privacy and Security Issues of Cloud-based Health Information Applications

### *Privacy*

Privacy issues in cloud computing occur when users lose control over their data because of moving data from one's own IT environment to a third-party vendor's cloud computing environment (Levi, 2012). When that happens, a hard question to ask is who owns the data? Do

the users own the data or do the third-party vendors own the data? No clarification on that matter certainly brings up legitimate responsibility issues.

In addition to the data ownership question, another privacy issue to data is an unauthorized secondary usage of data in the cloud. In this case, the cloud vendor might exercise a right of freedom to sell customers' sensitive data on its servers to others without permission. The third parties then can use them for commercial purposes such as advertisements and other profit-generating marketing schemes (Levi, 2012). Without a specific delineation of responsibilities, it will be hard to pinpoint who should be afflicted when a data breach occurs.

Furthermore, the cloud vendor may distribute user data across the state border, or even the nation's border. If so, who then has jurisdiction over the case when a data breach occurs? As each state or nation carries its own laws and regulations, one who claims jurisdiction may conflict with another who claims jurisdiction over the same case and may not concede its jurisdiction power (Levi, 2012).

Last, but certainly not least privacy issue in the cloud is that when users put their data on the cloud and move the data to another vendor's server, as dynamically being provisioned or vendors legally being switched over, who is responsible for the data left on the previously used servers? Without careful planning, this type of challenge will occur. Hackers who have legitimate access rights to the servers can access those data left on the old servers and cause data breach issues.

*Security*

The security issues in cloud computing can be categorized into confidentiality, integrity, availability, and accountability. Confidentiality comes in two forms: Data protection and data destruction. Data goes through its life cycle from creation to destruction. An appropriate policy

in place protects the data in some form from the time of creation. Another policy must be in place for the secure destruction or archival mechanism at the time of eventual retirement.

As for integrity, data should not, will not, and must not change from its original form without properly authorized permissions. Data integrity ensures that data is retained as intended. Any data tampering is illegal. The ensuring methods of data integrity include, but are not limited to, referential integrity, domain constraints, and triggers. These methods help prevent unintended changes of data from happening. Therefore, from a security perspective, what causes unintended changes to data is a concern. Unintended changes to data can result from some forms of malicious intent. One example can be SQL injection. SQL injection exploits database level vulnerabilities by sending a SQL command with "crafted input data" to the backend database server. It can retrieve personal sensitive information like social security numbers, credit card numbers, or any other invaluable data including financial data (Veracode, n.d.).

As for availability, threats in the cloud can come in the form of data loss. It can stem from natural disasters like fire, earthquake, or hurricane in a traditional sense. However, from the perspective of cloud computing, it extends to human attacks, errors, and accidents. Attackers can maliciously send software viruses to the environments of cloud providers to gain access to the systems. Cloud service users can cause human errors, too. For example, if a user utilizes the encryption capabilities for data, stores the encrypted data in the cloud, and loses the encryption keys, it can be difficult or even impossible to retrieve the data from it. Cloud providers can cause human accidents, as well. As they provide services that are invisible to customers, they can accidentally delete the data while they upgrade or extend the cloud services (Goodin, 2012).

The last part of data security is accountability. It includes vendor lock-in as well as vendor shutdown. For example, the current vendor can abandon its data accountability by filing

bankruptcy or shutting down its business. In other cases, other vendors may take over the current vendor's businesses. When they do that, they might bring in their security measures and policies, which may conflict with their previous vendor's (Chang, 2011).

As such, the migration and operation of ePHI to the cloud environment necessitates proper planning and execution, and risk assessment is part of the planning process. Risk assessment is part of risk management where risk assessment is a specific activity and risk management is an overall management scheme. Many risk assessment frameworks exist today. However, for this research study, the list shall be limited to the following for analysis:

- ISO 27000 family by international organizations
- NIST SP 800-30, 800-37 and HHS by the US federal government
- OCTAVE, TARA, FAIR, CSA, and HIMSS by US institute or individuals

## The Landscape of Risk Assessment Frameworks

### *ISO 27000 family*

The International Standard Organization (ISO) / International Electrotechnical Commission (IEC) has developed the ISO 27000 family to help organizations' Information Security Management System (ISMS). It ensures the security of their sensitive company information (ISO 27001, n.d.). See Table 1 (Vanderburg, n.d.).

Table 1 ISO 27000 family

| ISO 27000 Series | |
|---|---|
| ISO 27001 | ISMS requirements |
| ISO 27002 | ISMS controls |
| ISO 27003 | ISMS implementation guidelines |
| ISO 27004 | ISMS measurements |
| ISO 27005 | Risk management |
| ISO 27006 | Guidelines for ISO 27000 accreditation bodies |

Among the family of ISO / IEC 27000 standards, perhaps ISO 27001 (published in 2005) is the most familiar one and the latest version is ISO 27001:2013. ISO 27001 is a specification for an ISMS and its objective is to provide requirements for establishing, implementing, maintaining, and continually improving an ISMS. Thus, organizations adopt ISO 27001 as a "management framework" to protect their business-critical information. It provides high-level security roadmap requirements and comprehensive security controls in "Annex A," which contains a list of controls and their objectives. As for risk-associated management, ISO 27005 has it. It provides four (4) high-level processes including context establishment, risk assessment, risk treatment, and risk acceptance as shown below in Figure 5.

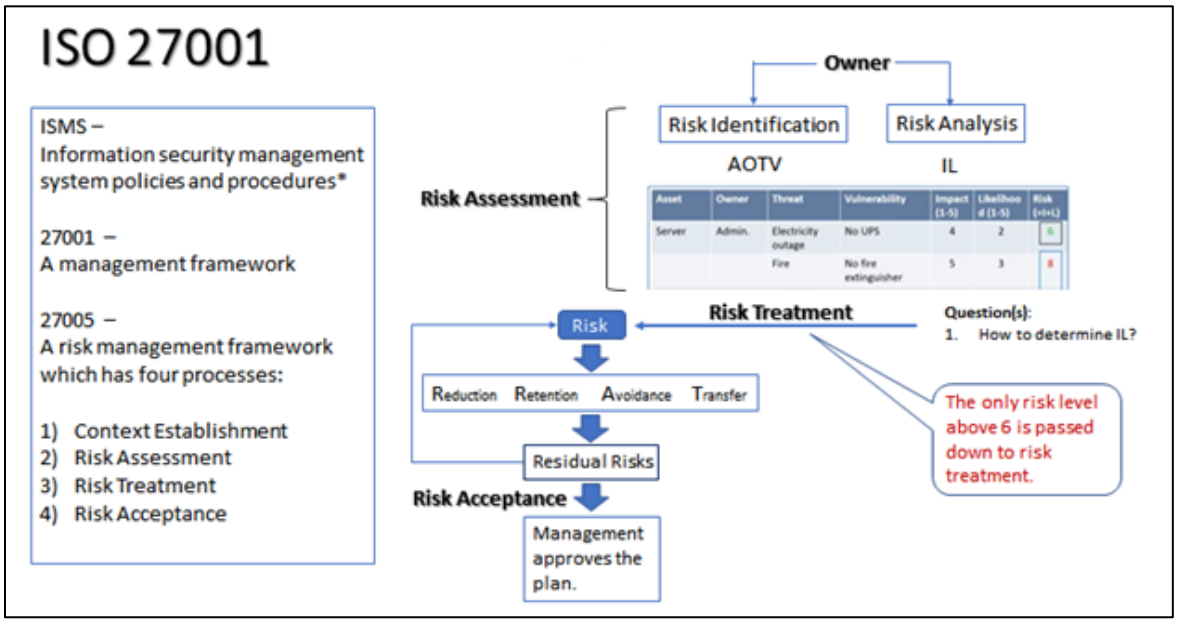



Figure 5. Analytical review summary of ISO 27001

However, this framework does not include specific methodologies (Kosutic, 2014), and also it is apparent that the ISO 27001 did not have HIPAA and HITECH in mind when it was first developed and published in 2005. To adopt the framework for HIPAA and HITECH, significant modifications are required. For example, the ISO 27001 framework's risk assessment begins with its assessment criteria. It can regard the HIPAA rules and regulations as a component

of the assessment criteria. Even the cloud computing requirements can be included in the criteria. However, such modifications become expensive as the HIPAA and HITECH rules and regulations need to be manually weaved into the existing framework.

The merit of the ISO 27001 framework comes with a list of 114 controls identified in Annex A, which is comprehensive and can be extendable. Because of its breadth, it applies to many security risk management projects. However, it would be labor-intensive to map and apply the 114 controls to health information systems and more specifically health information applications in the cloud. The ISO 27001 users must thoroughly analyze all 114 controls and determine their relevancies to cloud-based health information applications, which would take a considerable amount of time to put them in place for use.

Besides, the evaluations depend upon the assessors' judgment and checkmarks for each control item for their compliance. Hence, the evaluation processes are inherently subjective and may not generate consistent results when being evaluated again by a different assessor or even by the same assessor at a different time.

*NIST SP 800-53, 800-37, and 800-30*

As industries deal with information risks in cloud computing, US governmental organizations such as the National Institute of Standard and Technology (NIST), agency of the U.S. Department of Commerce, also have been engaged in this effort. NIST finalized the definition of cloud computing in the SP 800-145 in September 2011 (Mell & Grance, 2011).

From a perspective of information security, NIST has developed several frameworks and guidelines. Some of these are shown in Table 2 (NIST, n.d.). NIST SP 800-37 (2018) notes that "the risk management process described in these publications changes the traditional focus from the stovepipe organization-centric, static-based approaches to certification & accreditation

(C&A) and provides the dynamic environments of complex and sophisticated cyber threats, ever-increasing system vulnerabilities, and rapidly changing missions."

Table 2 Security relevant NIST special publications

| Special Publications (SPs) | |
|---|---|
| SP 800-18 | Guide for system security plan development |
| SP 800-30 | Guide for conducting risk assessments |
| SP 800-34 | Guide for contingency plan development |
| SP 800-37 | Guide for applying the risk management framework |
| SP 800-39 | Managing information security risk |
| SP 800-53 | Security controls catalog and assessment procedures |
| SP 800-60 | Mapping information types to security categories |
| SP 800-122 | Guide to protecting the confidentiality of personally identifiable information (PII) |
| SP 800-128 | Security-focused configuration management |
| SP 800-137 | Information security continuous monitoring |

Among these publications, SP 800-37 provides an overall umbrella framework for handling information system-relevant risks, which was originally based on an 8-step lifecycle. Over time, this has become a 6-step process. One of the processes is "Assess" and SP 800-30 for the "Guide for Conducting Risk Assessments" in September 2012 provides detailed processes for the assessment (NIST SP 800-30, 2012).

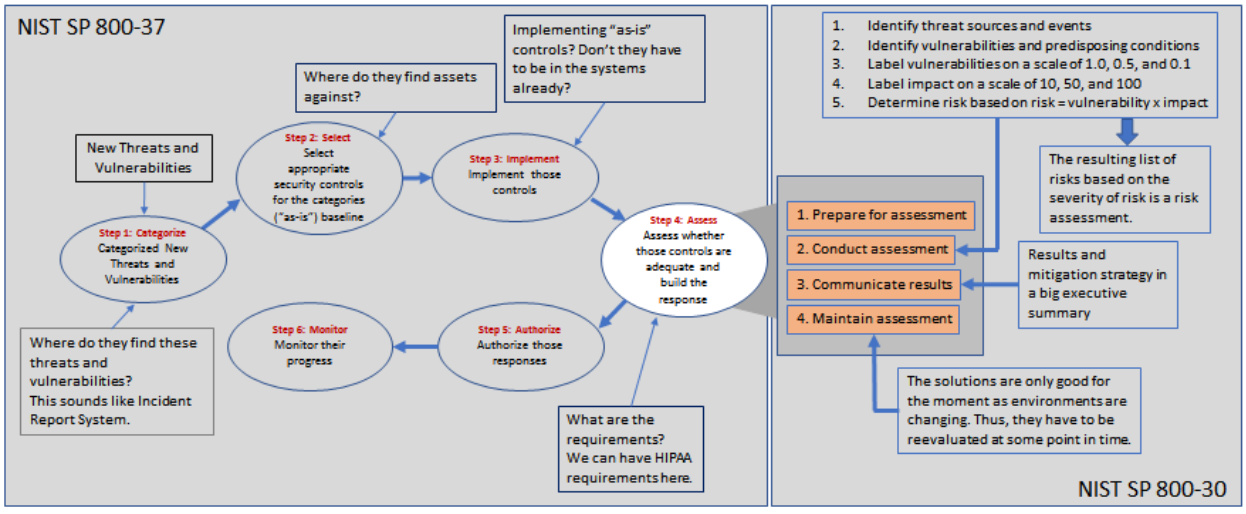



Figure 6. Analytical review summary of NIST SP 800-37 and 800-30

The high-level processes of NIST SP 800-37 and 800-30 lack details. NIST SP 800-37 omits the processes of identifying threats and vulnerabilities, for example. They come as a triggering input for the "categorize" process. The SP 800-37 also currently does not provide a way to capture the targeted information systems concerning threats and vulnerabilities. Rather, they categorize the systems as they come to the process. From that sense, the "categorize" process can extend its scope to include the cloud-computing environment as a category for the assessment.

Stage 4 "assess" assesses the systems that describe the sub-processes in SP 800-30: 1) Prepare for assessment, 2) conduct assessment, 3) communicate results, and 4) maintain assessment as shown in Figure 6. The framework lacks the details and forms a structure, not for the specific issues associated with health information applications in the cloud in general. Such a lack of specificity would not be cost-effective from a practical sense in assessing health information applications in the cloud.

Another drawback of this risk assessment framework is the method for measuring risk. The measurements of risks use a qualitative method in nature. For example, risk measurement is built on the general risk formula as shown below in Formula (1):

$$Risk = Impact \times Vulnerability \qquad (1)$$

As for impact, the risk matrix in Figure 7 assigns subjective values to each impact level (low, medium, and high) without specifying their corresponding rigorous evaluation processes. The examples include 10 for low, 50 for medium, and 100 for high. These arbitrary values are cross-mapped to each threat likelihood criterion. The examples in Figure 7 include 1.0 for high, 0.5 for medium, and 0.1 for low for the threat likelihood criteria. The multiplication of these two crossing values results in a single value of risk level indicator.

| Threat Likelihood | Impact | | |
|---|---|---|---|
| | ***Low*** (10) | ***Medium*** (50) | ***High*** (100) |
| ***High*** (1.0) | **Low** 10 X 1.0 = 10 | **Medium** 50 X 1.0 = 50 | **High** 100 X 1.0 = 100 |
| ***Medium*** (0.5) | **Low** 10 X 0.5 = 5 | **Medium** 50 X 0.5 = 25 | **Medium** 100 X 0.5 = 50 |
| ***Low*** (0.1) | **Low** 10 X 0.1 = 1 | **Low** 50 X 0.1 = 5 | **Low** 100 X 0.1 = 10 |

*Risk Scale: High ( >50 to 100); Medium ( >10 to 50); Low (1 to 10)*[8]

Figure 7. Risk matrix

These computed values carry no significant meanings other than they are used in prioritizing the risks. The risk level is low if the computed value is on the risk scale between 1 and 10, medium if between 10 above and 50, and high if between 50 above and 100. Thus, the medium values 25 and 50, for example, do not carry any distinctions between them other than they are in the same category of medium risk values. This framework requires assembling a group of experts for evaluation and assigning values together based on their judgment, knowledge, and experience. Hence, the evaluation of risks is inherently subjective and arbitrary, lacking the rigorous process of obtaining the values.

*OCTAVE*

The Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) comprise of a suite of tools, techniques, and methods for risk-based strategic assessment and planning technique for security. The OCTAVE method is an approach developed by the Computer Emergency Response Team (CERT) Division of the Software Engineering Institute (SEI) to help an organization achieve business resilience with the sponsorship of the Department of Defense (DoD) (Software Engineering Institute, n.d.). The overall OCTAVE framework looks as shown in Figure 8.

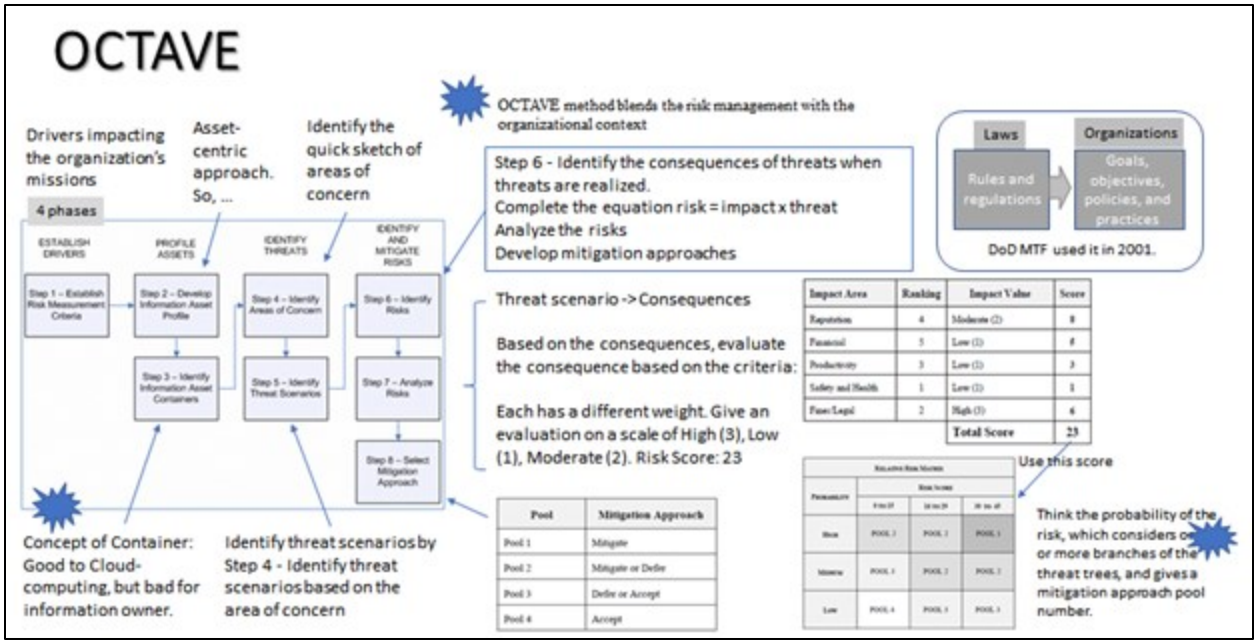



Figure 8. Analytical review summary of OCTAVE

OCTAVE has several different versions that include the OCTAVE original framework, OCTAVE-S, and OCTAVE Allegro. OCTAVE-S and OCTAVE Allegro both have fewer processes than their original version. OCTAVE-S is for smaller organizations, while OCTAVE Allegro for the larger organizations, especially focusing on protecting information-based critical assets (Shantamurthy, 2011). See Table 3 (Caralli et al, 2007).

Table 3 OCTAVE versions

| OCTAVE Versions | |
|---|---|
| September 1999 | OCTAVE Framework, Version 1.0 |
| September 2001 | OCTAVE Framework, Version 2.0 |
| December 2001 | OCTAVE Criteria, Version 2.0 |
| September 2003 | OCTAVE-S v0.9 |
| March 2005 | OCTAVE-S v1.0 |
| June 2007 | OCTAVE Allegro v1.0 |

OCTAVE essentially consists of eight steps and categorizes them into four phases as shown in Figure 8. Phase 1 ("establish drivers") only includes step 1, to establish risk measurement criteria. Phase 2 ("profile assets") includes two steps: Develop an information asset

profile and identify information asset containers. Phase 3 (“identify threats”) also includes two steps: Identify areas of concern and threat scenarios. Finally, in phase 4 (“identify and mitigate risks”) are three steps: Identify risks, analyze risks, and select mitigation approach.

While OCTAVE provides the structure, depth, and measurement of the information asset, this framework lacks the areas of cloud focus and implementation details. As shown in phase 2 and step 3, its target environment is the on-premise IT architecture and not the cloud computing architecture. More specifically, while there are many similarities with ISO 27000 series and NIST SP 800-30, the one important unique OCTAVE concept is “container” in step 3. This container is to measure the level of several different types of information regarding the information assets. It collects technical information such as the issues of software, hardware, and/or a third party, the physical information for internal, external, and third-party issues to the information asset access. It deals with the third-party, natural, and manmade issues of unintended information asset damage. However, usage of this container is more oriented toward the on-premise IT architecture than the cloud-based architecture.

In steps 4 and 5, the framework provides only high-level processes such as “Identify areas of concern and identify threat scenarios” leaving actual implementation details unspecified as OCTAVE uses a more self-directed business-centric approach. While this approach works fine with its flexibility for adaptation, it also burdens users to come up with all necessary implementation procedures to fill in the blanks. The way the DoD Military Health System used this framework depicts the point as shown in Figure 9 (Leo, 2005).

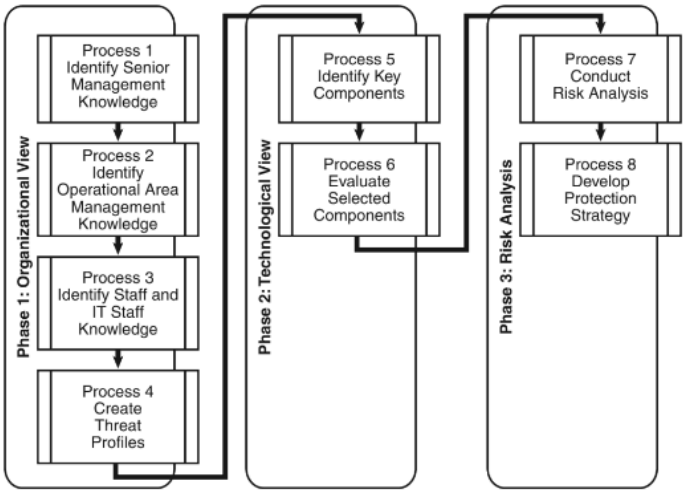



Figure 9. OCTAVE processes used by DoD military health community

Also, OCTAVE lacks guidance on security controls of the information assets compared to other risk assessment frameworks. ISO 27000 guides in selecting security controls. ISO 27002 and NIST SP 800-53 provide a comprehensive list of controls (Brunschwiler, 2013).

*OCTAVE + NIST SP 800-30*

Woody (2006) shows the case study of a mixture of NIST SP 800-30 with OCTAVE and the correlation work between them. OCTAVE focuses more on the process of risk assessment, although not necessarily intended for cloud-based health information applications. By doing so, OCTAVE presents one unique solution that ISO 27001 and NIST SP 800-50 do not have, which is the identification of information systems that can contain, process, or transmit ePHI in terms of containers, and that is excellent, as the container can be cloud services. However, the challenge is how cloud-computing resources can be captured where organizations no longer have control over the systems.

OCTAVE has two distinctive elements with the risk matrix, compared to the previous two frameworks – ISO 27001 and NIST SP 800-30. First, each area of impact carries a ranking. The higher the ranking value is, the higher the impact. The impact level is described as high,

low, and moderate, each weighing 1, 2, and 3, respectively. Thus, if one deals with financial, but the risk is low, then the ranking of financial is 5, and its impact level is low (1), then the resulting score is 5 x 1 = 5. So, when one calculates a risk scenario using the five impact areas (reputation, financial, productivity, safety and health, and fines and health), the sum of each impact area with its impact value gives the total risk score.

| Impact Area | Ranking | Impact Value | Score |
|---|---|---|---|
| Reputation | 4 | Moderate (2) | 8 |
| Financial | 5 | Low (1) | 5 |
| Productivity | 3 | Low (1) | 3 |
| Safety and Health | 1 | Low (1) | 1 |
| Fines/Legal | 2 | High (3) | 6 |
| | | **Total Score** | **23** |

Figure 10. Final risk score

This final risk score, then, shown above, links to one of the four pools depending on the probability of the threat occurrence, and each pool indicates what mitigation approach it takes. For example, if the risk score falls into pool 1, then the mitigation approach is “Mitigate” as opposed to “Defer” or “Accept.” A relative risk matrix does the linking of the final risk score to one of the pools as shown in Figure 11.

| Relative Risk Matrix | | | |
|---|---|---|---|
| Probability | Risk Score | | |
| | 0 to 15 | 16 to 29 | 30 to 45 |
| High | POOL 2 | POOL 2 | POOL 1 |
| Medium | POOL 3 | POOL 2 | POOL 2 |
| Low | POOL 4 | POOL 3 | POOL 3 |

Figure 11. Relative risk matrix

Finally, to obtain the probability of the equation, one needs to fill out a form titled Information Asset Risk Worksheet. In it, assessors must choose whether the probability of the threat occurrence is low, medium, or high. In other words, the framework does not spell out the selection process of obtaining the probability. Due to the lack of rigor in determining the risk probability, the resulting selection values are somewhat arbitrary and may not be consistent.

*FAIR*

Factor Analysis of Information Risk (FAIR) is a framework that can be used to understand, analyze, and measure information risks championed by Jack Jones, the former Chief Information Security Officer (CISO) of Nationwide Mutual Insurance (Violino, 2010). FAIR is a threat and loss-focused process. It provides an excellent way of quantifying threat events and their relative losses. The results indicate the level of risks for the threat. Instead of using the traditional threat impact x likelihood of the threat computation, it goes beyond a threat event probability x loss magnitude. However, this is a different way of saying the same thing, as loss magnitude can correlate with the impact of the threat, and vulnerability with threat probability. As it focuses on quantifying threat events and their relative losses, it fails to include the specific areas of health information applications as well as cloud computing environments. In other

words, it does not address the specific measurement components of HIPAA to health information applications in the cloud. It would have been better if it had offered the risk remediation method based on its threat impact figures. That step is particularly important as it pertains to financial loss in terms of HIPAA violation penalties, pending another audit. Communication with high-level management is another key measurement area of success, as the decision-makers should be informed of risks and their remediation costs, which cannot be found in FAIR. As for cloud computing, many questions remain unanswered as data travel over the Internet and cross-judicial boundaries. In the end, FAIR provides more quantified values for each scale, threat, vulnerability, and overall loss values, but they are not sufficiently prescriptive for cloud-based health information applications.

*TARA*

Intel developed the Threat Agent Risk Assessment (TARA) in 2010 with a notion that it is expensive and impractical to defend every aspect of vulnerabilities. The framework helps to distill an immense amount of information security attacks into a digest of the areas where the attacks are most likely to occur (Violino, 2010). It comes as part of the Mission Assurance Engineering package, which breaks the distilling process into two phases: Crown Jewels Analysis (CJA) and TARA.

While TARA is the most intriguing framework because it has more rigorous scale descriptions compared to other frameworks, it raises several questions, though, from a perspective of measurement components in assessing risks of health information applications in the cloud. First, the CJA phase of the process involves Concept of Operations (CONOPS), workflows, and use cases to determine the priority of the cyber assets, which is an important step in terms of defining the scope and priority. The process may be able to include legal

requirements, but it does not call out health information applications in the cloud nor HIPAA components, which can lead to omissions for the important legal requirements. Also, if the cyber assets are not within the organization's judicial boundaries, will this CJA process still be able to hold the effectiveness of the current processes? In the phase of TARA, more specifically the Cyber Threat Susceptibility Assessment (CTSA) phase of TARA, it is hard to see the linkage between cyber assets and Tactics, Techniques, and Procedures (TTP). TTP is a strategy to analyze and profile the threats associated with the assets. This problem can easily be fixed if the determination of assets pertinent to TTP is performed through the CJA process. The adversary types can be determined with the identified TTP. However, the major problem is the lack of specific methods of linking the assets to TTP. Who determines what and how? Those problems still need to be resolved. Finally, there is no way of determining whether the final solutions apply to cloud-based health information applications, as it is assumed that the TARA methodology is applied only to the information that is stored, processed, and transmitted within an enterprise.

*CSA, HIMSS, and HHS*

As noted earlier, in non-governmental environments, CSA runs STAR with CCM where the STAR is used to validate cloud service provider's cloud security based on CCM. When researchers use the concept of CSA's STAR program, they work under an assumption that the STAR program is fully automated, which is not true at the time of writing this research study. It is rather a labor-intensive process by entering data in the spreadsheet. To make it more dynamic, it needs to find a way to automate the processes filling in the spreadsheet.

Concerning U.S. health information systems, HIMSS has produced several toolkits for cloud computing, such as sample risk assessment for cloud computing in healthcare (HIMSS, 2020), which is also a spreadsheet-oriented toolkit. The accuracy and thoroughness depend on

the data entry person.

HealthIT.gov also suggests the use of its toolkits as guides on how to comply with HIPAA's privacy and security when using the cloud. Its toolkit enhances the HIMSS and CSA's spreadsheet into a software tool called Security Risk Assessment (SRA) tool as shown below.

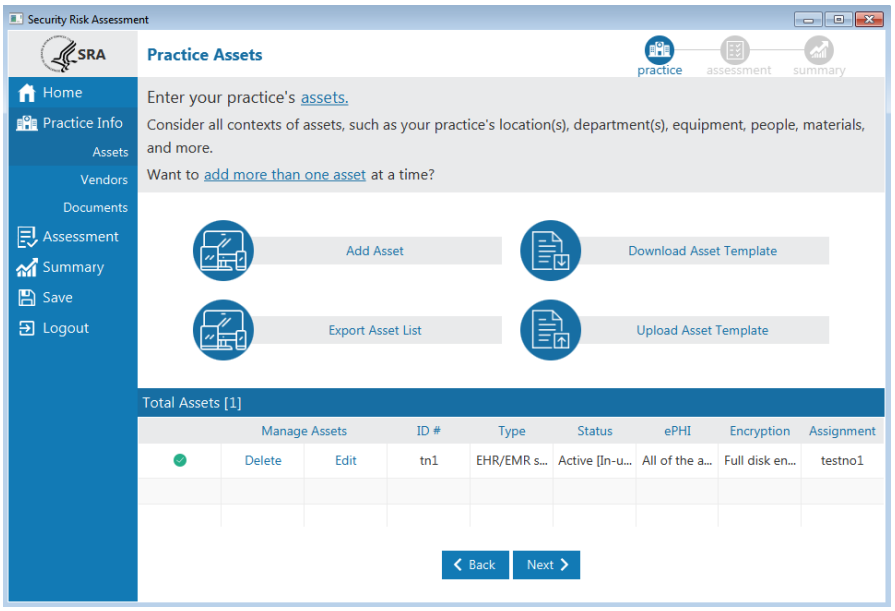



Figure 12. Security risk assessment toolkit

The latest version of it at the time of this research study is v3.0.1 (The Office of the National Coordinator for Health Information Technology, n.d.). This free tool has a user-friendly interface with which one can answer a series of questions, and then the software produces the results. However, it does not address health information applications in the cloud specifically. Also, the tool still depends on who answers the questions, which is a very subjective process.

## Summary

This chapter largely consists of two sections. The first section has been allocated to briefly review HIPAA and HITECH rules, and then the characteristics of cloud computing. The second section has been dedicated to the analysis of several risk assessment frameworks and discovered several shortfalls that make them difficult to use in assessing risks of cloud-based

health information applications. The modification cost of existing risk assessment frameworks to overcome the shortfalls is unpredictable. It would be difficult to assess the level of effort for the modifications. Therefore, adequate methodologies for assessing risks for the cloud-based health information applications are not readily available and could have caused many healthcare organizations to hold back the use of existing risk assessment methodologies. The Office of Civil Rights (OCR) completed its HIPAA audits of 115 covered entities in 2012, and the Office found that the lack of risk assessments was the most common finding (Office of Civil Rights HIPAA, 2016). Therefore, this research study finds it necessary to construct a new risk assessment framework that is specific to cloud-based health information applications.

# CHAPTER 3

# METHODOLOGY

## Quantitative Method

Before delving into the construction of a proposed risk assessment framework, it is necessary to have a general understanding of what risk assessment is. According to Yun (2018), risk assessment is a prediction of possible losses because of risk events. It can adopt a qualitative or quantitative method. The evaluation in qualitative risk assessment is subjective and heavily relies on assessors and evaluation participants; hence, it uses less computational means, and the results may not be consistent. This method may apply to situations where insufficient data or weak mathematical foundations persist. On the other hand, in quantitative risk assessment, the quality of the assessment depends on the accuracy and integrity of data used in the analysis. Aside from concerns on the accuracy and integrity of data, the quantitative assessment can achieve more scientific, rigorous, and consistent results. This research study has selected the quantitative method as an approach for constructing a proposed risk assessment framework.

In his dissertation, Yun (2018) also suggests the quantitative method based on vulnerability and sensitivity instead of traditional vulnerability and impact. The vulnerability concerns security and the sensitivity concerns privacy. With the emphasis on those two factors, he develops the VS-PIRA II model for smart cities. VS stands for vulnerability and sensitivity. PIRA II is the second version of PIRA-personal information risk assessment concerning smart cities. The use of vulnerability and sensitivity as leading factors suits the purpose of this research

study.

For the feasibility testing of the proposed risk assessment framework, this research study has chosen open-source cloud-based health information applications (HIA). The reason why this research study has chosen software application among many other IT aspects is that business applications have topped one of the major concerns of the organizations as shown below (Skybox security, 2019):

Table 4 Vulnerabilities by category

| | 2017 | 2018 |
|---|---|---|
| Business apps | 24% | 23% |
| Internet and mobile | 21% | 22% |
| Dev tools | 16% | 15% |
| Servers & Desktop OS | 16% | 14% |
| Desktop apps | 9% | 10% |
| Networking and security | 9% | 11% |
| IoT | 3% | 3% |
| OT | 1% | 1% |
| Other | 1% | 1% |
| Total | 100% | 100% |

Also, the reason for choosing open-source applications is not just because they are readily available for the study, but also because many large healthcare organizations strategically adopt open-source applications to reduce cost. The examples include the Department of Defense (DoD) Military Health System (MHS) and the Department of Veterans (DVA) Veterans Health Administration (VHA). Both use the same open-source VistA electronic health record systems as a base of their applications, originally Composite Health Care System (CHCS) and VistA, respectively (VistA, n.d.). These applications still use the legacy programming language Massachusetts General Hospital Utility Multi-Programming System (MUMPS) (VA Monograph,

2018), which cannot be candidates for this study. The applications are required to be cloud-based for feasibility testing. Also, this research study focuses on Ruby on Rails applications because it is one of the most popular web programming languages along with PHP and Python (Codecondo, 2017). On the database side, the major data source is Mitre corporation's SyntheticMass (SyntheticMass, n.d.). As the application obtained from an open-source application code repository does not typically contain data, this research study finds it necessary and practical to segregate the work of data from the analysis of application code, and that suits well with the VS-PIRA II-based new proposed risk assessment framework. The SyntheticMass database offers textual data from its database.

## Feasibility Testing Method

As hinted above, the feasibility testing method of this research study adopts a software engineering model of separating business processes from data models because the separation of business processes from data models correlates well with the proposed model and well suites measuring HIPAA security as a vulnerability and privacy as sensitivity. See Figure 13 (Smith & Sarfaty, 1993).

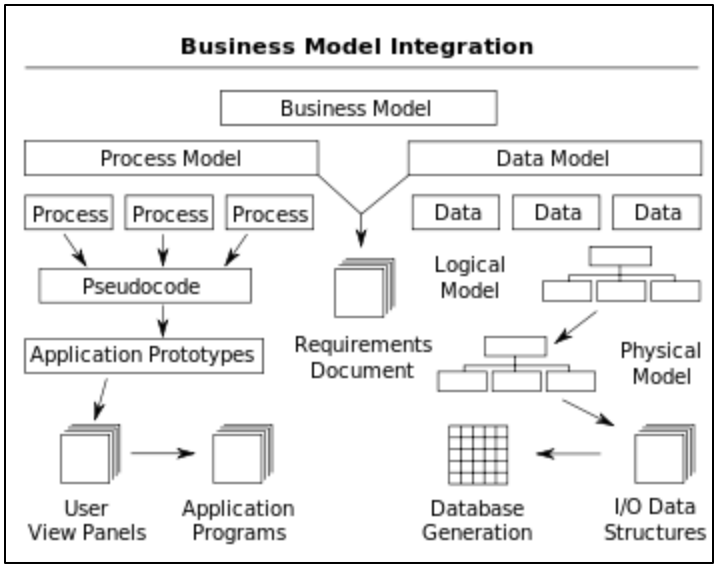



Figure 13. Interactions between business processes and data

## The Selection of Software Tools

This research study has chosen the Open Group Enterprise Architecture Framework (TOGAF) based modeling tool ArchiMate due to its scalability and flexibility to capture the scope of work for the study. As for scalability, it can cover the layers from requirements noted as motivation to infrastructure colored in green. As for flexibility, it can move the components of the model around as well as reuse them as deemed appropriate without a change.

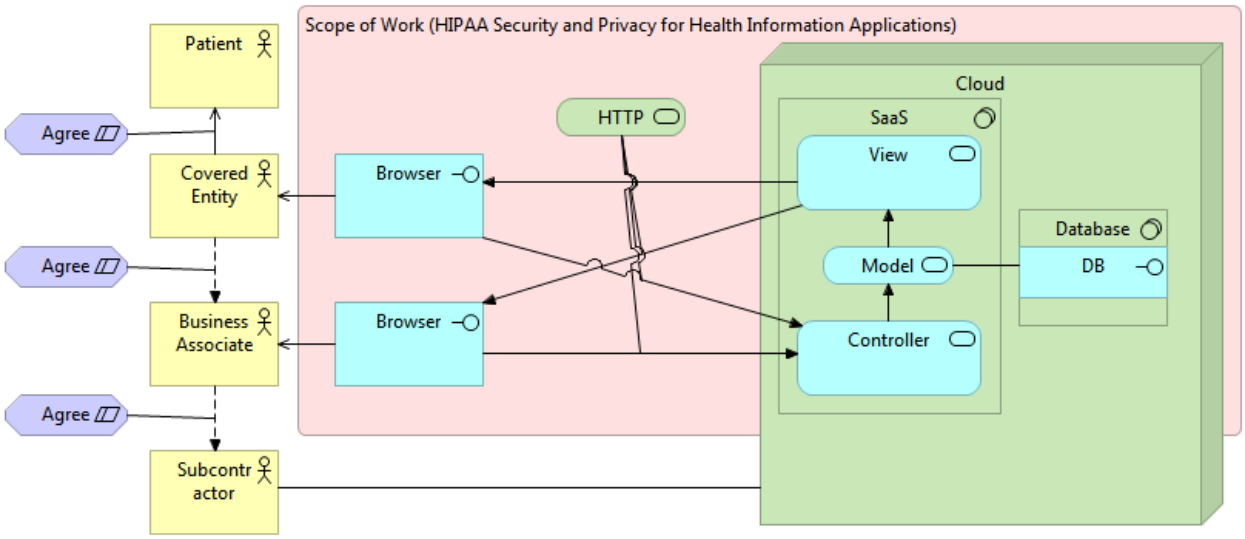



Figure 14. The scope of work

In Figure 14, ArchiMate captures the scope of work boundaries surrounding the study for the risk assessment framework for cloud-based health information applications.

For feasibility testing, this research study has chosen the web programming language, Ruby-on-Rails. “Ruby” is a programming language and “Rails” is a framework that is built with the Ruby programming language. The meaning of a framework in this context is a collection of code, tools, and utilities that provide you one specific structure to work with, and this structure makes code more organized (Castello, 2018). The structure in this study uses the implementation of MVC architectural pattern: Model, view, and controller (MVC), which can reside on the server-side for the cloud.

The selected health information applications in the cloud for this study are assumed to have this MVC architectural pattern as shown in Figure 15 (Pearce, n.d.).

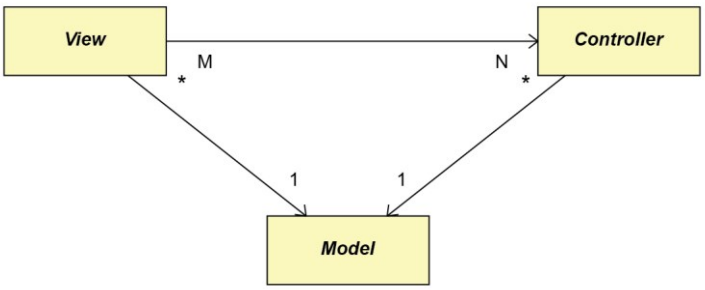



Figure 15. MVC architectural pattern

In the MVC architecture pattern, the application data and logic are encapsulated by the "model" and should be independent of presentation logic which is encapsulated by the "view" and "controller" (Pearce, n.d.). Therefore, users like business entities and/or business associates accessing the cloud-based health information applications interact with the "view," and the users request for information from the controller areas of webpages through the browser and receive the information through the "view." Then, the system works on the "model" that fits the information request and retrieves the information from a database, and sends it to the "view," in a nutshell. The database in this architecture is encapsulated in the "model." Pearce (n.d.) clarifies the interactions between these three components through the sequence diagram in Figure 16 and explains, "Views are responsible for user input and output. A dialog box is a good example of a view. Controllers implement the logic for the allowable transactions that can be performed on the model. The model encapsulates the fine-grained business logic and data" (Pearce, n.d.). See Figure 16 below.

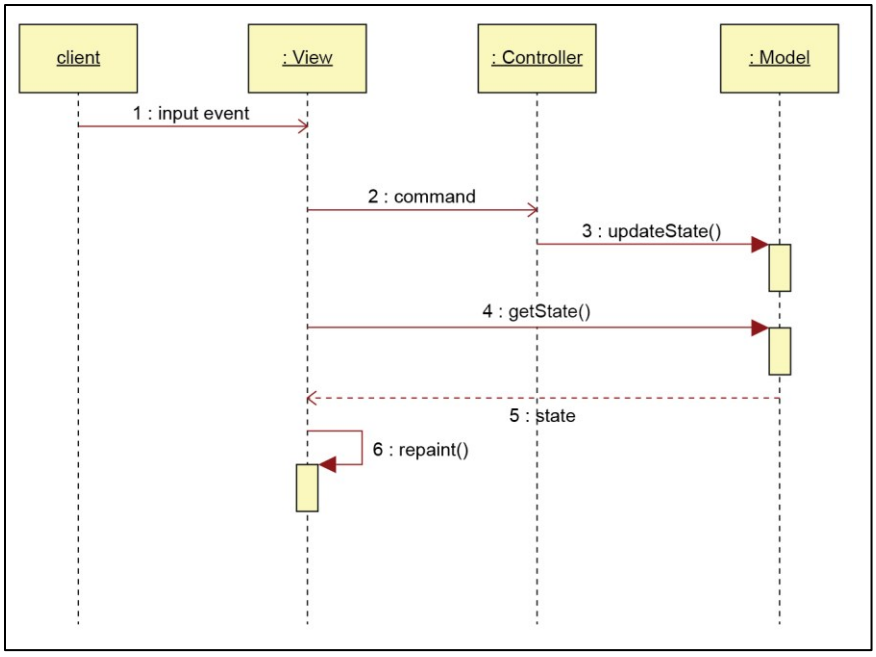



Figure 16. MVC pattern

To achieve the objectives stated in Chapter 1, this research study is to:

1) Develop the scope of work for the study

2) Construct a proposed quantitative risk assessment framework, VS-HRA, that can evaluate risks of cloud-based health information applications as well as data from databases derived from the adoption study of VS-PIRA II (Yun, 2018)

3) Implement the proposed framework as a feasibility study of VS-HRA. The testing process is two-fold: Application code and data from the database following Smith and Sarfaty's (1993) example as shown in Figure 13. The feasibility test used the Objective-Key-Results (OKR) method where OKR is a way to measure if the objectives are met by reviewing key results which can be numeric measures or activities (Chau, 2020). The following are the key results as success measurement criteria:

- Use the decision-making process to find at least one open-source cloud-based health information application code.

- Use the MMC method (Matching, mapping, and correlating) to convert security warnings into measurable numeric values for at least 95% of security warning types.
- Use a distance-based check to identify at least 90% similar for the standard surnames and given names for the practical perspective.
- Use a pattern-matching-based check to detect legitimate SSNs, DoBs, postal addresses, and bank account numbers.
- Use a parity-digit-based check to detect legitimate credit card numbers.

4) Develop selection criteria and methods of choosing:

- Open-source cloud-based health information application code
- Data from a simulated database

5) Develop methods of harmonizing:

- SCA generated security warnings with CVE scores on the application code side
- Pattern recognition algorithm-generated degree of similarity of data against standard data on the database side

## The Adoption Study of VS-PIRA II

*Overall Approach*

VS-PIRA II (Yun, 2018) is a model that appears in a doctoral dissertation approved by Wonkwang University in 2018 in South Korea. VS-PIRA II (Yun, 2018) uses vulnerability and sensitivity as two basic factors instead of traditional vulnerability and impact and depicts their relationships including threat factors and threat aftereffects in Figure 17.

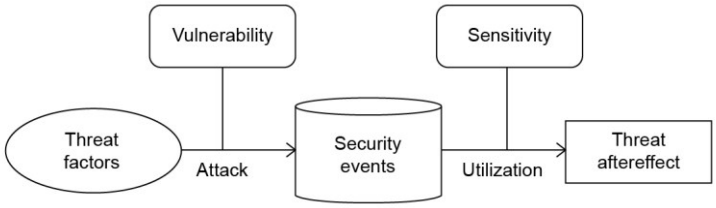



Figure 17. Relationship model

Threat factors are potential reasons for damages to systems or organizations, which may exploit the weaknesses or vulnerabilities of personal information in systems, and in turn, may potentially result in threat aftereffects depending on the sensitivity of personal information. The higher the personal information sensitivity indicates the larger the threat aftereffect value is. In other words, the level of risks is proportional to these two factors. This research study utilizes the same principles of VS-PIRA II and adopts vulnerability and sensitivity of personal information as two dominant factors in constructing a proposed risk assessment framework.

*VS-PIRA II*

Yun (2018) asserts that measuring vulnerability and sensitivity directly is difficult. To address this issue, he adds protection measures for vulnerability and threat aftereffect measures for sensitivity, and refines VS-PIRA into VS-PIRA II as shown below with the added measures.

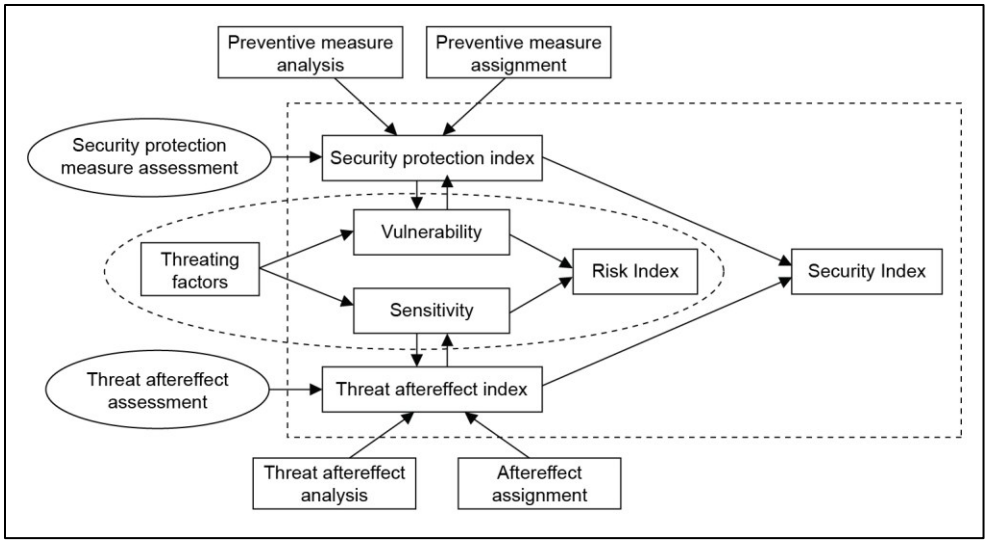



Figure 18. VS-PIRA II model

Yun (2018) provides mathematical models as well for these measurements in VS-PIRA II, but the data collection areas still depend on questionnaires, surveys, and meetings. However, this research study shall not depend on questionnaires, surveys, and meetings to avoid such subjective, less computational, and less consistent results. Rather, this research study shall utilize data that SCA and pattern recognition algorithm tools generate.

*Mathematical Models*

Yun (2018) in his dissertation uses mathematical models with vulnerability and sensitivity as main elements as shown below in Formula (2):

$$R^* = f(V^*, S^*) = V^* \times S^* \qquad (2)$$

For all practical purposes, this research study closely follows his model, but due to the access limitation of health information applications in operation, this study handles vulnerability and sensitivity separately. The vulnerability ($V^*$) in his model consists of two indexes: The first level and the second level index. The first level index is regarded as the level of a preventive measure for each vulnerability of the system, whereas the second level index declares a list of vulnerability types and values. The sensitivity (S*) consists of sensitivity values of personal

information that are based on questionnaires and sensitivity aftereffects that are also based on questionnaires.

*VS-HRA*

Based on those principles, the modified VS-HRA model is as follows:

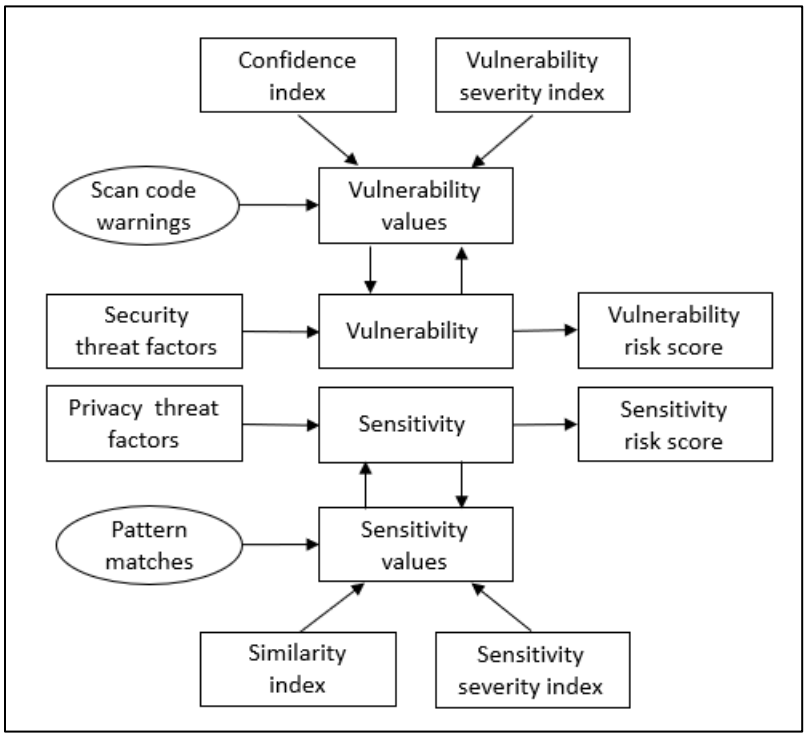



Figure 19. VS-HRA

The major difference between VS-PIRA II and the new model, VS-HRA, this research proposes is a different data collection approach. VS-HRA utilizes software tools such as SCA tools and pattern recognition algorithms to avoid subjective, less computational, and less consistent results. Figure 20 (Mansourov, n.d.) depicts the overall concept of the VS-HRA model as well as the use of static code analysis (SCA) and pattern recognition tools.

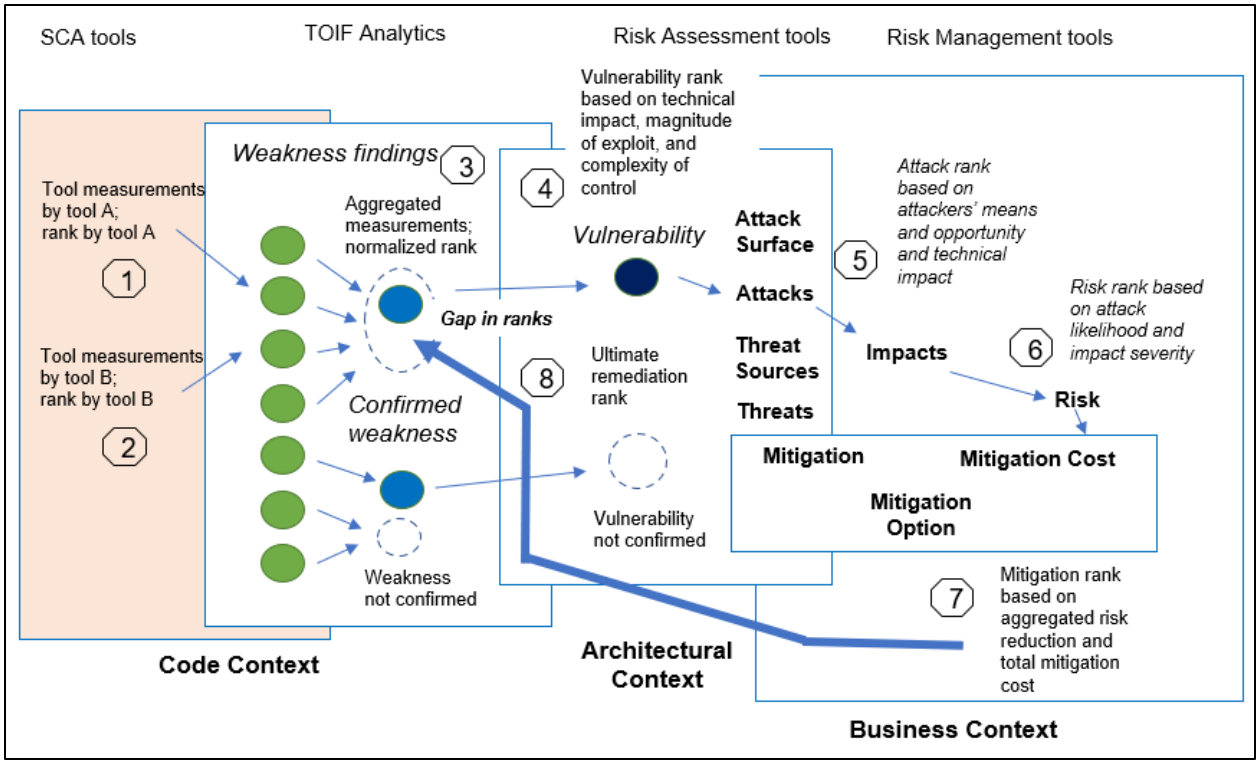



Figure 20. Multiple ranks in the context of risk management

*Vulnerability*

As shown above, the tool-output-integration framework (TOIF) evaluates the confidence level of "confirmed weakness." If it is high, then its exploitability is high. If it is low, then its exploitability may not be high. In other words, the confidence level of the confirmed weaknesses determines the preventive strength of each vulnerability type. The weakness of the system is inversely proportional to the preventive strength. The confidence level is based on the scale of high (3), medium (2), and low (1), following the ranking scheme of ENISA (2012).

Table 5 Confidence level scheme

| Confidence level | Value |
|---|---|
| High | 3 |
| Medium | 2 |
| Weak | 1 |

This research study uses this scheme in combination with the number of security event occurrences as vulnerability. With that scheme, the following mathematical models are the results of the adaptation of Yun's VS-PIRA II model (2018). Park et al. (2021) utilized them in their vulnerability risk assessment implementation. The confidence level of the security warnings consists of three elements as follows:

$$F = \sum_{j=1}^{m} l_j n_j h_j \ \ where\ h\ \in \{0, 1\} \qquad (3)$$

The first element is the availability of its confidence index, $h_j$. If it contains a confidence index value, $h_j = 1$. If not, $h_j = 0$. The second element, $n_j$, is the number of occurrences in the code for the particular warning type. The third element, $l_j$, is the number of confidence index levels and $m$ is the total number of confidence index levels. Then, the confidence level for the particular security warning type is obtained by dividing the summation of the available confidence index times the number of occurrences for each security warning by the summation of confidence index levels as follows:

$$C = \frac{\sum_{j=1}^{m} l_j n_j h_j}{\sum_{j=1}^{m} l_j} \ where\ n\ \leq 3 \qquad (4)$$

The confidence $C$ indicates the confidence level of vulnerability for each security warning type. $n$ is the number that can be adjusted, depending on which scale the study chooses. This study chooses 3 by default.

Each security warning has its own vulnerability risk measure, $v$, calculated as follows:

$$v = C \cdot S \qquad (5)$$

The element $S$ is the severity index. The severity index was already established from the previous SCA-CVE harmonization activities.

The overall aggregated vulnerability risk score of the software application in the cloud can be obtained by the following calculation:

$$V = \frac{\sum_{i=1}^{k} C_i S_i}{\sum_{i=1}^{k} S_i} \qquad (6)$$

$k$ is the number of security warnings generated by the SCA tool. $V$ is the overall vulnerability for software application code. The following chart adapts both Yun (2018) and Microsoft (n.d.).

Table 6 Vulnerability assessment measures

| Vulnerability assessment measure | Vulnerability degree | Descriptions |
|---|---|---|
| $2.5 < V \leq 3.0$ | Most likely | Health information is most likely exploited. |
| $1.7 < V \leq 2.5$ | More likely | Health information is more likely exploited. |
| $1.0 < V \leq 1.7$ | Less likely | Health information is less likely exploited. |
| $V \leq 1.0$ | Unlikely | Health information is unlikely exploited. |

*Sensitivity*

To begin the discussion of sensitive information, it would be prudent to discuss what it entails first. As not all data attributes are treated equally from a perspective of privacy, there is a list of HIPAA PHI attributes. HHS provides the ePHI data elements that need to be protected from abuse or misuse as shown in Table 7 (HHS Health Information Privacy, n.d.):

Table 7 ePHI data elements

| | PHI data elements | | PHI data elements |
|---|---|---|---|
| 1. | Names | 10. | Account numbers |
| 2. | Geographic subdivisions smaller than a state | 11. | Certificate/license numbers |
| 3. | All elements of date (except year) for dates directly related to an individual | 12. | Vehicle identifiers and serial numbers |
| 4. | Telephone numbers | 13. | Medical Device Identifiers |
| 5. | Fax numbers | 14. | Web Universal Resource Locators (URLs) |

Table 7 – cont.

| | PHI data elements | | PHI data elements |
|---|---|---|---|
| 6. | Electronic mail addresses | 15. | Internet Protocol (IP) address numbers |
| 7. | Social security numbers | 16. | Biometric identifiers, including finger and voiceprints |
| 8. | Medical record numbers | 17. | Full face photographic images and any comparable images |
| 9. | Health plan beneficiary numbers | 18. | Any other unique identifying number, characteristic, or code, unless otherwise permitted by Privacy Rule for re-identification |

VS-HRA adopts established schemes to determine the sensitivity of data based on their corresponding standard values. Like vulnerability, sensitivity also consists of measurement values for severity level and its equal relation to sensitivity measure. For example, to calculate severity level, this research study adopts the established color-coded scheme for risk weights that the North Carolina (NC) State University utilizes as its policy for personal information protection. The NC State University's color-coded sensitivity scheme breaks the sensitivity into five levels including purple, red, yellow, green, and white where the white color indicates the unclassified information. The following are the meaning of the color code and the brief description of sensitivity (NC State U, 2019):

- *Purple classification level* - The purple classification level means "ultra-sensitive data" where unauthorized disclosure or loss poses a high risk or impact to the university or its affiliates, or where specific data categories require special privileged access management to support the university's ability to prevent unauthorized data modification, use, or destruction.

- *Red classification level* - The red classification level means "highly sensitive data" where unauthorized disclosure or loss poses a high risk or impact to the university or its affiliates. Authorized users of this data are responsible for managing data confidentiality, integrity, and availability to prevent unauthorized data modification, use, or destruction.
- *Yellow classification level* - The yellow classification level means "moderately sensitive data" where unauthorized disclosure or loss poses a moderate to low risk or impact to the university or its affiliates. Authorized users of this data are responsible for managing data confidentiality, integrity, and availability to prevent unauthorized data modification, use, or destruction.
- *Green classification level* - The green classification level means "normal or not-sensitive" data where unauthorized disclosure or loss poses a low risk or impact to the university or its affiliates. This information may be disclosed to individuals regardless of their university affiliation. Minimal security measures are needed to control the unauthorized modification, use, or destruction of this data.
- *White classification level* - The "unclassified data" means data that is created or collected within the university's data environment and has not been classified by the data steward(s). This data should be controlled at a minimum as yellow/moderately sensitive until the final classification is assigned (NC State U, 2019).

As noted above, the basis of the severity level of sensitive information stems from organizations' policies that usually reflect relevant federal laws and regulations such as HIPAA and HITECH, and the NC State University's data policy provides the identification of HIPAA relevant sensitive data elements and the color code for their severities as shown below in Table 8 (NC State U, 2019):

Table 8 NC State University's data elements and color code

| | Data Element | HIPAA |
|---|---|---|
| 1 | Adult's personal name (last, first, middle) | Yellow (if with PHI) |
| 2 | Social security number | Purple |
| 3 | Citizenship or country | White |
| 4 | Race | Yellow |
| 5 | Sex | Yellow |
| 6 | Marital status or effective date | Red |
| 7 | Spouse or partner name | White |
| 8 | Dependents (relationship to individual or employee) | Red |
| 9 | Birthdate | Red |
| 10 | Death date | White |
| 11 | Birthplace | White |
| 12 | Mother's maiden name | Red |

In the NC State University policy, the personal name has 'yellow,' social security number 'purple,' citizenship or country 'white,' race 'yellow,' sex 'yellow,' marital status 'red,' spouse name 'white,' dependent relationship 'red,' birth date 'red,' death date 'white,' birthplace 'white,' and mother's maiden name 'red.' The meaning of each of color code is as follows: Ultra-sensitive (4), highly sensitive (3), moderately sensitive (2), not sensitive or normal (1), and unclassified (0). Like vulnerability, for this research study, the values for each of the levels are granted 4, 3, 2, 1, and 0 following the ranking scheme of ENISA (2012), 4 being very high business impact, 3 being high business impact, 2 medium, 1 low and 0 very low. For the sake of this study, the data elements and severity values of PHI are limited to those shown in Table 9 and used as a severity index:

Table 9 The severity level values of customized PHI data elements

| | Data elements | Severity level (*S*) | Assigned value for this study |
|---|---|---|---|
| 1 | Name (Last Name) | Yellow | 2 |
| 2 | Name (First Name) | Yellow | 2 |
| 3 | Social Security number (SSN) | Purple | 4 |
| 4 | Birthdate | Red | 3 |
| 5 | Home address | Yellow | 2 |
| 6 | Bank account number | Red | 3 |
| 7 | Credit card number | Purple | 4 |
| | | GSV | 20 |

The gross severity value (GSV) is the summation of each sensitive data element's severity index value assigned, noted in $g$ that is 20. While the severity index is formulated with organizations' policies, the similarity index is built upon each input data. This similarity index is essentially built by the similarity level of two textual data strings, one for input and the other for a standard pattern to be compared with. Various comparison methods are adopted for this research including the distance-based check (Jaro-Winkler formula), pattern-match-based check (regular expressions), and parity-digit-based check (Luhn's algorithm).

The Jaro-Winkler formula is applied to the input data against the referenced data. The following are the descriptions for the Jaro-Winkler formula for measuring the distance between two strings s1 and s2 (Appaloosa Store, 2018):

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \qquad (7)$$

$d_j$ is the Jaro distance. $m$ is the number of matching characters. $t$ is half the number of transpositions. That is the number of different characters divided by 2. $|s_1|$ is the length of the first string $s_1$. $|s_2|$ is the length of the second string $s_2$. The Jaro-Winkler formula provides more

insights on the similarity based on the $p$ constant scaling factor (the default value 0.1 is used.) and the $l$ maximum common prefix length.

$$d_{jw} = d_j + lp(1 - d_j) \qquad (8)$$

The regular expression-based pattern recognition ensures no length restrictions as long as it is more than 2 characters. The pattern can be as simple as it can, but for validation, it can be as restrictive as it can be in the spirit of Postel's principle (Woods, 2018). Postel's principle is: Be liberal in what you accept from others but be conservative in what you do. While it can apply more restrictive regular expressions for textual data validation such as checking on spaces, apostrophes, umlauts, accents, or hyphens, it still leaves a question of whether it is valid data such as names. To improve its validity, one can use the idea that Chen (2012) provides, in which he uses the pre-stored name rule wherein the name includes the given name and the surname and their statistical attribute of the name (Chen, 2012). To that end, this study uses the Census 2010 surnames and the Social Security Administration 2010 given names for the top 100 popular names as reference data (Census, 2010; Social Security Administration, n.d.).

As for the parity digit check method, it relies upon Luhn's algorithm that validates various identity numbers including credit card numbers, which was built by the IBM computer scientist Hans Peter Luhn in the 1950s to protect companies from accidental typing errors (Hussein et al., 2013). To validate a credit card number, while every other odd digit number is multiplied by 3, even digit number is multiplied by 1 (GS1, n.d.). The summation of these numbers should be the multiple of ten. For example, while the account number ranges from three digits to 17 digits, the following example shown in Table 10 consists of nine digits.

Table 10 Check Digit Method

| BAN Format | Digit Positions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Acc-9 | $N_1$ | $N_2$ | $N_3$ | $N_4$ | $N_5$ | $N_6$ | $N_7$ | $N_8$ | $N_9$ |
| **Step 1**: Multiply a value of each position by | | | | | | | | | |
| | $\times 3$ | $\times 1$ | $\times 3$ | $\times 1$ | $\times 3$ | $\times 1$ | $\times 3$ | $\times 1$ | |
| **Step 2**: Sum the results | | | | | | | | | |
| **Step 3**: Subtract the sum from its nearest equal or higher multiple of ten = **Check Digit** | | | | | | | | | |

For example, the given bank account number is 789456124 where the last digit is a check digit, 4. Then, the calculation goes as follows:

$$(7 \times 3) + (8 \times 1) + (9 \times 3) + (4 \times 1) + (5 \times 3) + (6 \times 1) + (1 \times 3) + (2 \times 1) = 86$$

When 86 is subtracted from its next nearest equal or higher multiple of ten, which is 90 in this case, its remainder is 4. That matches with the check digit. Therefore, it is a valid account number.

Using those schemes, to obtain the sensitivity degree (S) of the protected health information of the equation shown in Formula (2), the calculation of severity comes first, the similarity measure next as the recognition of privacy violation suspects, and then the summation of each data element's sensitivity measure.

To obtain the similarity and severity index-based sensitive value of the input data (Yun, 2018), first of all, it obtains the severity value, $W$, that is assigned for PHI data elements by an organization's policy as shown in Table 9 - The severity level values of customized PHI data elements. The individual sensitivity severity index, $s$, was then built with the assigned severity value and the privacy violation indicator, $c$, that detects whether the values exist or not, using the following:

$$s = W_j \cdot c_j \; where \; c \in \{0, 1\} \qquad (9)$$

where $j$ represents a reference index value of the PHI data elements. As for the similarity index, this study adopted the index from the concept of Jaccard similarity. In a simplistic term, the Jaccard similarity provides a similarity ratio by using the following equation (Sowmya et al., 2018):

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Where X and Y represent the sets that contain the n-grams of sentences $S_1$ and $S_2$, respectively. Thus, the similarity index, $M$, of input data for the data element in scope can be obtained by the following arithmetic calculation:

$$M(r, R) = \frac{r}{R} \qquad (10)$$

where $r$ is the number of detected records by a data source scanner and $R$ is the total number of target records for scanning. So, the product of each data element's severity and similarity indexes provides the individual sensitivity value $P$ as follows:

$$P = s_i \cdot M_i \qquad (11)$$

where $i$ represents a reference index value of the PHI data elements in the scope. Now, to obtain the sensitivity risk measure of each sensitive data element, it would be normalized with the gross severity value (GSV) of all data elements, $g$, within the scope:

$$g = \sum_{j=1}^{m} W_j \cdot c_j \; where \; c \in \{0, 1\} \qquad (12)$$

where $m$ is the total number of sensitive data elements defined by HIPAA PHI, but within the scope that is based on the sensitivity level classified by NC State University's

classification. The final sensitivity risk score is graded by adding all the sensitivity risk measures.

$$S = \sum_{i=1}^{n} \frac{P_i}{g} \tag{13}$$

$n$ is the number of data elements in the scope of this research. $S$ would be the final measure that is resulted from the sensitivity assessment for privacy risk and interpreted as shown in Table 11.

Table 11 Sensitivity assessment measures

| Sensitivity assessment measure | Sensitive degree | Descriptions |
|---|---|---|
| $0.83 < S \leq 1.0$ | (high) sensitive | The PHI after utilization gives rise to serious influences on individuals/organizations on multiple aspects; in this condition, multi-aspect protective measures are needed. |
| $0.56 < S \leq 0.83$ | (medium) semi-sensitive | The PHI after utilization gives rise to large influences on certain personal aspects; in this condition, it is needed to adopt preventive measures immediately. |
| $0.33 < S \leq 0.56$ | (low) weak-sensitive | The PHI after utilization gives rise to certain influences on individuals; in this condition, it is needed to adopt certain preventive measures based on personal information risk assessment. |
| $S \leq 0.33$ | (none) insensitive | The PHI after utilization gives rise to no significant personal information, with only minor disturbance; in this condition, it is needed to keep ordinary security strategies. |

To unify the equal ratio of the relation with the values of vulnerability (V) and sensitivity (S), one can finally determine the risk level of the health information applications in the cloud with the multiplication of the two: $R = V \times S$. Due to the practicality of implementation, this research focused on each value independently of the other and demonstrated the values separately.

In summary, VS-HRA can generate more objective, computational, and consistent results with the data-driven models presented above.

## The Implementation of VS-HRA

The implementation of VS-HRA was conducted as a feasibility study with cloud-based health information applications and data from databases.

*Step 1*

Develop the selection criteria of cloud-based health information applications from a perspective of practicality. Real-world applications may be difficult to obtain for the study because they are often proprietary assets. Also, different programming languages are used by different health information systems to build their systems on the cloud platform. Because of the limitations, this research used open-source health information applications hosted on the GitHub server, and the GitHub server is one of the most popular open-source application code repositories.

According to Park et al. (2021), approximately more than 190 million repositories live on GitHub as of November 5, 2020 (GitHub, 2020). However, selecting an application code by sifting through 190 million repositories is a daunting task. Park et al. (2021) approached the task with a hierarchical selection criterion to eliminate unwanted repositories at search levels, which resembles the classification procedure.

Many different algorithms exist for classification procedures in the form of decision trees, including Iterative Dichotomizer 3 (ID3), C4.5 (derived from Unix command c4.5 for decision tree) by Quinlan, and Classification and Regression Trees (CART) by Breiman (Singh & Gupta, 2014). This research study adopted the ID3-based method that Park et al. (2021) uses, specifically following the decision tree shown below.

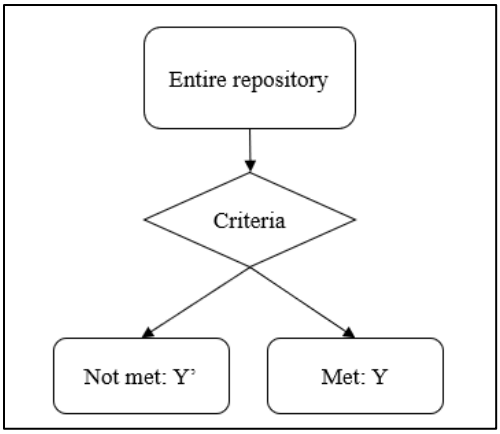



Figure 21. Decision tree for classification procedure

Y means the repositories that meet the criteria. Y' means the ones that do not. Essentially this tree can grow by having its sub-trees according to the ID3 algorithm.

*Step 2*

Once chosen, obtain the required system account information such as application paths, and download the open-source code for evaluation from the implementation environment. In other words, the chosen application repository is technically cloned for evaluation on the implementation platform. This research study focuses on Ruby on Rails applications.

*Step 3*

To select an SCA tool, several selection criteria have been established. *On the application side*, the tool selected should be able to scan Ruby on Rails. Also, the generated warning types should correspond, or have the features that can be mapped, to the CVSS scores (Saucs, 2018; NIST NVD, 2017). *For the database*, the tool should be able to scan the textual database contents with several data pattern-matching abilities.

**Application Code.** This research study has chosen Brakeman's (n.d.) as a vulnerability scanner for the application side. According to Brakeman (n.d.), "Brakeman is a free vulnerability scanner specifically designed for Ruby on Rails applications. It statically analyzes Rails

application code to find security issues at any stage of development." However, the tool comes with several limitations. The potential negatives of using this tool are:

- False positives
- It only knows code.
- Not omniscient

Despite those limitations, Brakeman is a good choice because of no configuration, run anytime, better coverage, flexible testing, and speed. Brakeman also provides a confidence level for each warning type. A ranking scheme for the confidence level is shown in Table 4 above with the ranking values of 3, 2, and 1.

According to Brakeman (n.d.), the value, 3, means that either this is a straightforward warning or the user's input is very likely being used in unsafe ways (high impact). The value, 2, generally indicates an unsafe use of a variable, but the variable may or may not be user input (medium impact). The value, 1, typically means that user input was indirectly used in a potentially unsafe manner (low impact).

The exploitability weight of Brakeman's warning types follows the CVSS ratings (Saucs, 2018; NIST NVD, 2017). However, as this weight is not inherently available from the SCA tool, it necessitates a way of relating the warning types to the CVSS ratings to collect the reasonable scores for each warning type. Park et al. (2021) attempt to bridge those two in the name of harmonization. Their harmonization effort is not perfect, but at least provides a hint of the possibility of bridging those two using the model of Haufe et al. (2016, p. 11) and this research study adopts the harmonization model. The following are the work results of Park et al. (2021):

Table 12 CVE-SCA (Brakeman) harmonization template

| | CVE Harmonization with SCA (Brakeman) | Harmon. Type | # of Vulnerability Report from 2006 to 2017 | Severity Score Mean Values |
|---|---|---|---|---|
| 1 | **DOS:** | | 9 | |
| | Denial of service<br>Dangerous send (Brakeman: DOS potential) | Matching<br>Correlating | | 5.84<br>5.84 |
| 2 | **Code Execution:** | | 13 | |
| | Remote code execution<br>Remote execution in YAML.load<br>Unsafe deserialization (Brakeman: remote code execution) | Mapping<br>Mapping<br>Correlating | | 7.48<br>7.48<br>7.48 |
| 3 | **Overflow:** | | 0 | |
| | Divide by zero | Mapping | | |
| 4 | **Memory corruption:** | | 0 | |
| | | None | | |
| 5 | **SQL injection:** | | 9 | |
| | SQL injection<br>Command injection<br>Dangerous evaluation (like command injection)<br>Dynamic render paths (Brakeman: execute code to modify database)<br>Format validation (Brakeman: attackers insert code) | Matching<br>Mapping<br>Correlating<br>Correlating<br>Correlating | | 6.82<br>6.82<br>6.82<br>6.82<br>6.82 |
| 6 | **XSS:** | | 17 | |
| | Cross-site scripting<br>Mail link (Brakeman: Cross-site scripting)<br>Cross-site scripting (Content tag)<br>Session settings (Brakeman: stealing cross-site scripting)<br>Cross-site scripting (JSON) | Matching<br>Correlating<br>Mapping<br>Correlating<br>Mapping | | 4.30<br>4.30<br>4.30<br>4.30<br>4.30 |
| 7 | **Directory traversal:** | | 5 | |
| | | None | | 5.00 |
| 8 | **HTTP response splitting:** | | 1 | |
| | | None | | 5.00 |
| 9 | **ByPass something:** | | 8 | |
| | SSL verification bypass<br>Attribute restriction (Brakeman: vulnerable to bypass)<br>Mass assignment (Brakeman: bypass) | Mapping<br>Correlating<br>Correlating | | 5.66<br>5.66<br>5.66 |
| 10 | **Gain information:** | | 1 | |
| | Information disclosure<br>Default routes (Brakeman: Not intended)<br>Unsafe redirects (Brakeman: redirect away or to a malicious site) | Mapping<br>Correlating<br>Correlating | | 5.00<br>5.00<br>5.00 |

Table 12 – cont.

| | CVE Harmonization with SCA (Brakeman) | Harmon. Type | # of Vulnerability Report from 2006 to 2017 | Severity Score Mean Values |
|---|---|---|---|---|
| 11 | **Gain privileges:** | | 1 | |
| | Authentication<br>Session manipulation (Brakeman: gain access)<br>Basic authentication<br>Unscoped find (Brakeman: Access any account)<br>Weak hash (Brakeman: creating weak passwords or signatures) | Mapping<br>Correlating<br>Mapping<br>Correlating<br>Correlating | | 0.00<br>0.00<br>0.00<br>0.00<br>0.00 |
| 12 | **CSRF:** | | 0 | |
| | Cross-site request forgery | Matching | | 0.00 |
| 13 | **File inclusion:** | | 0 | |
| | | None | | 0.00 |

Park et al. (2021) assert that it is necessary to complete the scoring to obtain objective scores for the study. They utilize the real data from a Rails software product developed by RubyonRails to do so. Also, they note that each category may obtain more than one vulnerability score from the real CVSS data on RubyonRails which the report has accumulated over 10 years from 2006 to 2017. For that matter, Park et al. (2021) take the mean value approach of each category for the SCA tool's security warning types.

**Data from Database.** As for the database, many database scanning tools exist today. Among them are Scuba, AppDetectivePro, ZenMap, BSQLHacker, Oracle Auditing Tool, and so on to just name a few (Shakeel, 2018). However, few tools check actual privacy contents.

Table 13 Database scanning tools

| Focus area | Tools |
|---|---|
| Weak passwords | Scuba, AppDetectivePro |
| Check configurations | Scuba, AppDetectivePro, Oracle Auditing Tool |
| Missing patchworks | Scuba, AppDetectivePro |
| Database instances | ZenMap |
| Database vulnerabilities | ZenMap |
| SQL injection | BSQL Hacker |

CUSpider is another tool that provides a deep scanning capability of data and captures the number of occurrences of pattern-matched data. However, like Brakeman's, CUSpider comes with several limitations, as well. CUSpider is limited because it does not support certain database types. To use it, the database needs to be dumped into flat files from the database for scanning (Martinez, 2008). CUSpider also does not provide protected health information matching patterns. The lack of adequate tools for this research necessitates the development of each protected health information attribute's matching patterns.

Thus, this research study has decided to use some other tools like R Studio and develop the pattern matching tools using regular expression techniques and others for protected health information attributes. The rationale behind choosing R Studio is that it uses the language R. Many analytical tools exist today, however, as shown below in Table 14, R is one of the industry-accepted data mining tools (Huang, 2016):

Table 14 List of data mining tools and share

| **Data Mining Tool** | **2017 Share** | **2016 Share** | **2015 Share** | **2014 Share** |
|---|---|---|---|---|
| R | 52.0% | 49.0% | 46.9% | 38.5% |
| RapidMiner | 32.8% | 32.6% | 31.5% | 44.2% |
| SQL | 34.9% | 35.5% | 30.9% | 25.3% |
| Python | 52.6% | 45.8% | 30.3% | 19.5% |
| Excel | 28.1% | 33.6% | 22.9% | 25.8% |
| KNIME | 19.1% | 18.0% | 20.0% | 15.0% |
| Hadoop | 15.0% | 22.1% | 18.4% | 12.7% |
| Tableau | 19.4% | 18.5% | 12.4% | 9.1% |
| SAS | - | 5.6% | 11.3% | 10.9% |
| Spark | 22.7% | 21.6% | 11.3% | 2.6% |

Besides, as the application obtained from GitHub.com typically does not contain data, as indicated earlier, this research study finds it necessary to segregate the work of data from the analysis of application code. Mitre corporation's SyntheticMass offers workable data for that

matter. SyntheticMass is a synthetic patient and population health data that simulates the state of Massachusetts. From the data source Mitre's SyntheticMass, include allergies, care plans, conditions, encounters, imaging studies, immunizations, medications, observations, organizations, patients, payer transitions, payers, procedures, and providers. The reengineered database schema looks like the following:

*Step 4*

As noted above, this research study utilizes the VS-HRA model adopted from VS-PIRA II for risk assessment. Park et al. (2021) apply the same model and focuses on the vulnerability side of risk issues only. This research study adds data sensitivity risk issues to them to make a complete set of vulnerability and sensitivity.

**Application Code.** As for application code, the SCA tool generates security warnings. The following risk calculation chart shown in Table 15 was constructed, based on the mathematical models shown earlier.

Table 15. Risk assessment calculation chart

| Security Warnings | Severity Index | Confidence Index | | | Confidence Index Value (F) | Conf. Level (C) | Vul. Risk Measure (v) |
|---|---|---|---|---|---|---|---|
| | | L3 | L2 | L1 | | | |
| $W_1$ | $S_1$ | $1N3$ | $1N2$ | $1N1$ | $F_1 = (3 * 1N3) + (2 * 1N2) + (1 * 1N1)$ | $C_1 = F_1/6$ | $C_1 * S_1$ |
| $W_2$ | $S_2$ | $2N3$ | $2N2$ | $2N1$ | $F_2 = (3 * 2N3) + (2 * 2N2) + (1 * 2N1)$ | $C_2 = F_2/6$ | $C_2 * S_2$ |
| … | | | | | | | |
| $W_n$ | $S_n$ | $3N3$ | $3N2$ | $3N1$ | $F_n = (3 * nN3) + (2 * nN2) + (1 * nN1)$ | $C_n = F_n/6$ | $C_n * S_n$ |

Here the subscript $n$ is the number of security warnings that the SCA tool generates.

**Data from Database.** As for the privacy violation suspects from the database, once the data elements for the sensitive information are selected, the three different pattern matching algorithms (Jaro-Winkler algorithm, string-matching algorithm, and parity-digit check algorithm) generate findings against the target textual database contents.

Applying the data policy of the North Carolina (NC) State University, the severity index is constructed. The similarity index is obtained by the number of pattern-matched records, $r_i$, divided by the total number of records ($R$) as follows:

$$M = r_i \div R \qquad (14)$$

The sensitivity value is the product of the severity index and similarity index. Based on Formulas (9), (10), (11), and (12), the following risk assessment chart shown in Table 16 is constructed for sensitivity data:

Table 16 Similarity index and severity index entries

| Sensitive Data Elements | Severity Index $s_n$ | Similarity Index $M_n$ | Sensitivity Value $P_n$ | Sensitivity Risk Measure $S_n$ |
|---|---|---|---|---|
| Name | $s_1 = W_1 \times \{0,1\}$ | $M_1 = r_1 \div R$ | $P_1 = s_1 \cdot M_1$ | $S_1 = \frac{P_1}{g}$ |
| SSN | $s_2 = W_2 \times \{0,1\}$ | $M_2 = r_2 \div R$ | $P_2 = s_2 \cdot M_2$ | $S_2 = \frac{P_2}{g}$ |
| DoB | $s_3 = W_3 \times \{0,1\}$ | $M_3 = r_3 \div R$ | $P_3 = s_3 \cdot M_3$ | $S_3 = \frac{P_3}{g}$ |
| Address | $s_4 = W_4 \times \{0,1\}$ | $M_4 = r_4 \div R$ | $P_4 = s_4 \cdot M_4$ | $S_4 = \frac{P_4}{g}$ |
| CCN | $s_5 = W_5 \times \{0,1\}$ | $M_5 = r_5 \div R$ | $P_5 = s_5 \cdot M_5$ | $S_5 = \frac{P_5}{g}$ |
| BAN | $s_6 = W_6 \times \{0,1\}$ | $M_6 = r_6 \div R$ | $P_6 = s_6 \cdot M_6$ | $S_6 = \frac{P_6}{g}$ |

Finally, the aggregated sensitivity risk score, $S$, is obtained by the Formula (13).

***Step 5***

Once all the SCA tools for application code and pattern matching algorithms for data from the database are executed and the respective risk assessment calculations are completed, the interpretation of the risk score obtained needs to be conducted as the final step.

## Summary

In summary, this chapter describes a research methodology using a quantitative method. It has constructed a new risk assessment framework, VS-HRA, by utilizing VS-PIRA II (Yun, 2018) as a base model and developed a plan to apply it to the open-source health information applications and textual data from a database as a feasibility study of VS-HRA.

# CHAPTER 4

# RESULTS AND ANALYSIS

## Implementation Environment

The experiment environment for implementing VS-HRA consists of Windows 7 host OS, Linux guest OS on Oracle's Virtual Box running on a machine in a private setting. Fox and Patterson (2014) used this environment to create the course materials to teach "Engineering Software as a Service (SaaS) to expand the traditional software engineering into the cloud. (Refer to Appendix A - Tool installation.) Park et al. (2021) established the same implementation experiment environment for their vulnerability risk assessment case study. This research study additionally adapts the same software engineering model platform principle for the new risk assessment framework implementation to include privacy risk assessment in addition to vulnerability risk assessment. See below the layered implementation architecture for the framework.

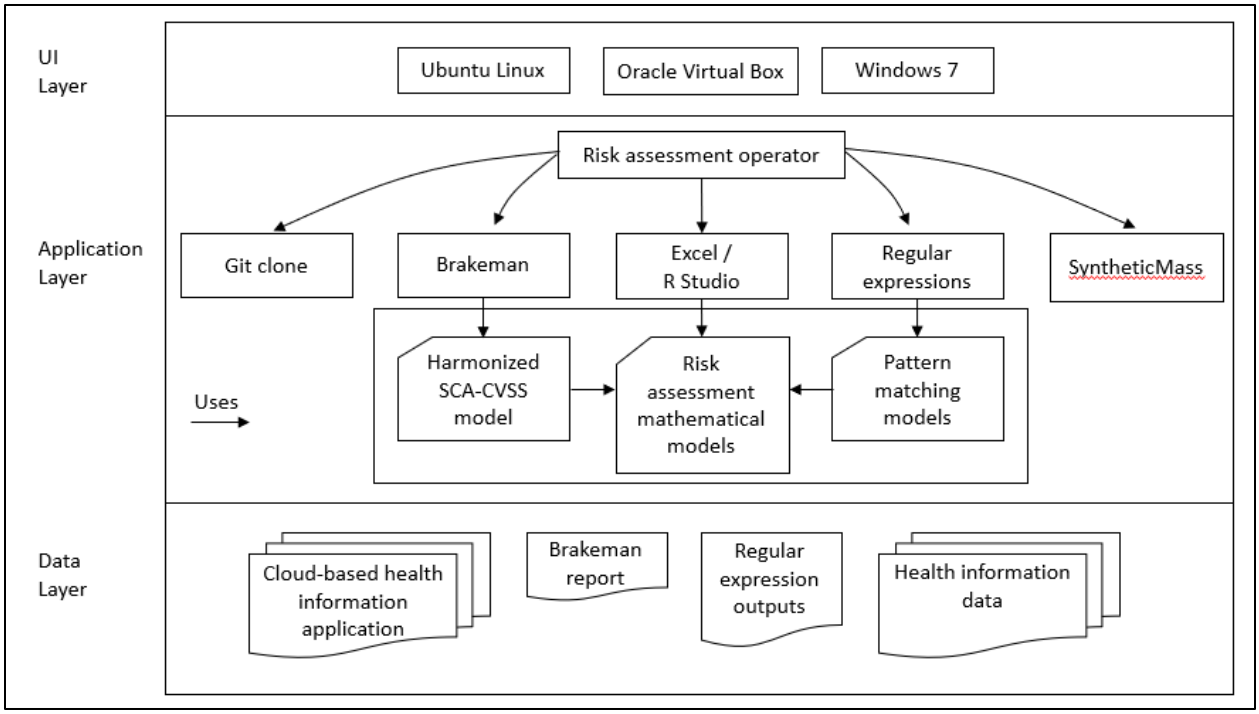



Figure 22. Layered implementation architecture

The implementation architecture above is to depict how several components for the implementation are orchestrated in terms of the 3-tier system architecture. The implementation environment consists of the UI layer, application layer, and data layer. The layered architecture notation used above follows the modified version of Mai et al.'s (2018) and Park et al.'s (2021).

**Execution of Application and Database Selection**

To select an SCA tool, several selection criteria were established. On the application side, the tool selected should be able to scan Ruby on Rails web applications as noted for the selection of the programming language under the section - Feasibility Testing Method. Consequently, this research study selected Brakeman as it is built to scan the Ruby on Rails applications. Also, the generated warning types should correspond, or have the features that can be mapped, to the CVSS scores (Saucs, 2018; NIST NVD, 2017). As for the database side, its tool should be able to scan the textual database contents. The author was not able to find adequate tools that scan

various sensitive data from a database and decided to develop several data scan algorithms for this research. The developed algorithms include one that adopts the Jaro-Winkler algorithms, string-matching regular expression algorithms, and a parity-digit check-based algorithm.

*Application Code*

As mentioned in the methodology section, there are approximately more than 190 million repositories on GitHub as of November 5, 2020 (GitHub, 2020). Park et al's (2021) decision tree method based on ID3 was appropriate for selecting health information applications from such a large resource pool. The method, which is like the classification procedure, employs a hierarchical selection criterion to eliminate unwanted repositories at search levels. Based on this research's requirements for search, Health / Health Information Application (HIA), Ruby / Ruby on Rails, and Cloud are the most essential selection criteria.

According to Jenhani et al. (2008), a decision tree is a flowchart-like hierarchical tree structure made up of three basic elements: decision nodes, attributes, and branches (or edges), and they can be used in two ways: building procedure and classification procedure. The classification procedure appears to be better suited for locating an appropriate application code. As a result, for the sake of simplicity, this study used the classification procedure depicted as shown below (Rokach & Maimon, 2015; Park et al., 2021).

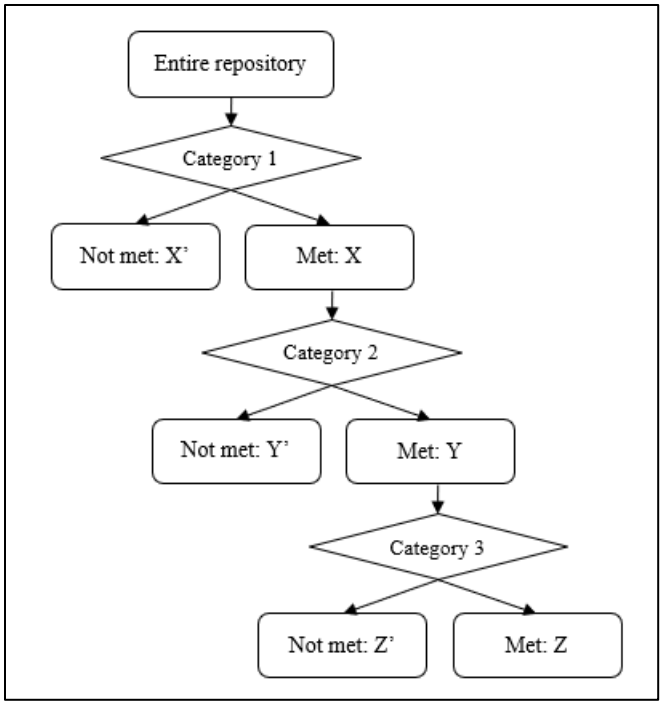



Figure 23. Application selection decision tree

The decision tree shown above removes a set of unwanted application code repositories based on the search criteria 1. The next search criteria 2 is applied to the remaining set, and the next criteria 3 to the further remaining set. Table 17 is the actual results from the application selection decision tree method that was executed against the server (Park et al, 2021).

Table 17 The implementation of the application selection decision tree method

| | 1st criteria | 2nd criteria | 3rd criteria | HIA/HIS/Health | Ruby | Cloud |
|---|---|---|---|---|---|---|
| Tree 1 | HIA | Ruby | Cloud | 252 | 5 | 2 |
| Tree 2 | HIA | Cloud | Ruby | 252 | 2 | 10 |
| Tree 3 | Ruby | Cloud | Health | 959* | 84,937 | 959 |
| Tree 4 | Ruby | Health | Cloud | 156 | 84,937 | 156* |
| Tree 5 | Cloud | Ruby | Health | 959* | 959 | 10,979 |
| Tree 6 | Cloud | Health | Ruby | 413 | 16 | 10,979 |
| Tree 7 | HIS | Ruby | Cloud | 428 | 8 | 2 |
| Tree 8 | HIS | Cloud | Ruby | 428 | 2 | 12 |

Trees 3, 4, 5 are rejected due to the complexity of searching. Trees 1, 2, and 8 are rejected due to the complexity of installation. Tree 6 is accepted to investigate them further for a compatibility check with the SCA tool. HIA stands for Health Information Application Cloud and HIS stands for Health Information System. Note: * means there are too many repositories to manually scan to identify the repositories that meet the search criteria.

The execution of the first search criteria of Tree 6, which is "cloud," resulted in 10,979 repositories as of November 15, 2020. Out of those 10,979 repositories, the next search criteria, "health," has been executed which has resulted in 413 repositories. Out of those 413 repositories, the next criteria, "ruby," has been executed. Table 18 displays the results of 16 repositories in GitHub.

Table 18 Sixteen (16) repositories and their paths in GitHub from Tree 6

| No. | App's paths | Results |
|---|---|---|
| 1 | cloudfoundry-attic/health_manager | ruby-1.9.3-p484 required |
| 2 | coveooss/docker-cloud-health-agent | syntax error |
| 3 | trotter/cloudfoundry-health_manager-cookbook | syntax error |
| 4 | zephirworks/cloudfoundry-health_manager-cookbook | same as #3 |
| 5 | Virksaabnavjot/HealthCare-Heroku-CloudApplication | syntax error |
| 6 | rcyep87/HealthCloud | ruby-2.2.1 required |
| 7 | markleonardireland/healthApp | syntax error |
| 8 | svs990/healthcare | syntax error |
| 9 | maximussivv/irish-health | syntax error |
| 10 | kevm66/4thYear_Cloud-Application-Development-Project | install ruby_parser |
| 11 | ejoonie-irm/omniauth-irm_health | syntax error |
| 12 | tnataly/Mshelath | authentication required |
| 13 | JustinJruby/two_net | syntax error |
| 14 | BSKERRITT/CAD-Health-System-Project | syntax error |
| 15 | Reccy/college-zeal_healthcare_system | syntax error |
| 16 | d/cchm | syntax error |

Park et al. (2021) asserted that Tree 6 did not provide any workable software application repository. Hence, Tree 7 was chosen to work with. The execution of the first search criteria of Tree 7, which is "health information system," resulted in 428 repositories as of November 16, 2020. Out of those 428 repositories, the next search criteria, "ruby," has been executed which has resulted in 8 repositories. Out of those 8 repositories, the next criteria, "cloud," has been executed, resulting in 2 repositories. Table 19 displays the results of 8 repositories including 2 cloud-met repositories in GitHub and the results of the compatibility checking:

Table 19 Eight (8) repositories and their paths in GitHub from Tree 7

| No. | App's paths | Results |
|---|---|---|
| 1 | gwc/medadata | Asks for a new app |
| 2 | siteslave/his-rpc | OK |
| 3 | Reccy/college-zeal_healthcare_system | Syntax error |
| 4 | numvarn/localwisdom | Authentication required |
| 5 | Virksaabnavjot/HealthCare-Heroku-CloudApplication | Syntax error |
| 6 | myoung34/puppet-mirthconnect | Asks for a new app |
| 7 | rubysoftwaredev/RubySET | Asks for a new app |
| 8 | charlesjom/PHIMS | install ruby_parser |

One application code repository was found as a workable solution from Tree 7. The following is the summary of the compatibility checking off all 22 repositories from both Tree 6 and 7, 2 of which are redundant. Park et al. (2021) found that they were mostly incompatible due to various reasons as shown below:

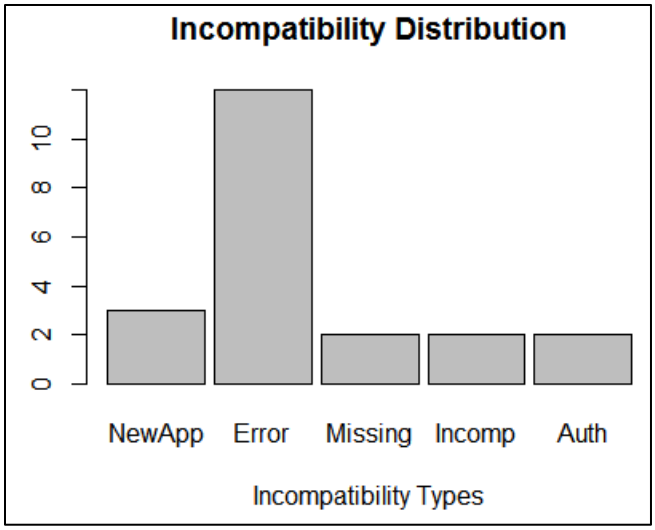



Figure 24. The results of compatibility checking

The compatibility checking has generated 3 repositories asking for a new app, 12 repositories errored out due to syntax errors, 2 missing components, 2 incompatible versions, 2 required for user name and password, and 1 duplication. However, out of those 22 repositories, only one (1) repository was able to make it usable for vulnerability risk assessment testing for this research.

The accepted repository name and its path were: siteslave/his-rpc, and its title is Health Information and Report System. Coincidently, this app was a Rails app. Park et al. (2021) further noted that this application code may or may not move to the cloud, but with the assumption it does, this research study chooses this app for an experiment as the MVC architecture makes it possible to do so.

*Data from Database*

The major data source is Mitre corporation's SyntheticMass (n.d.). As the application obtained from GitHub.com typically does not contain data, this research study finds it necessary to segregate the work of data from the analysis of application code as noted earlier in the Methodology. Mitre corporation's SyntheticMass offers workable data for that matter.

SyntheticMass is a synthetic patient and population health data that simulates the state of Massachusetts. From the data source Mitre's SyntheticMass, include allergies, care plans, conditions, encounters, imaging studies, immunizations, medications, observations, organizations, patients, payer transitions, payers, procedures, and providers. The reengineered database schema looks similar to the one that follows:

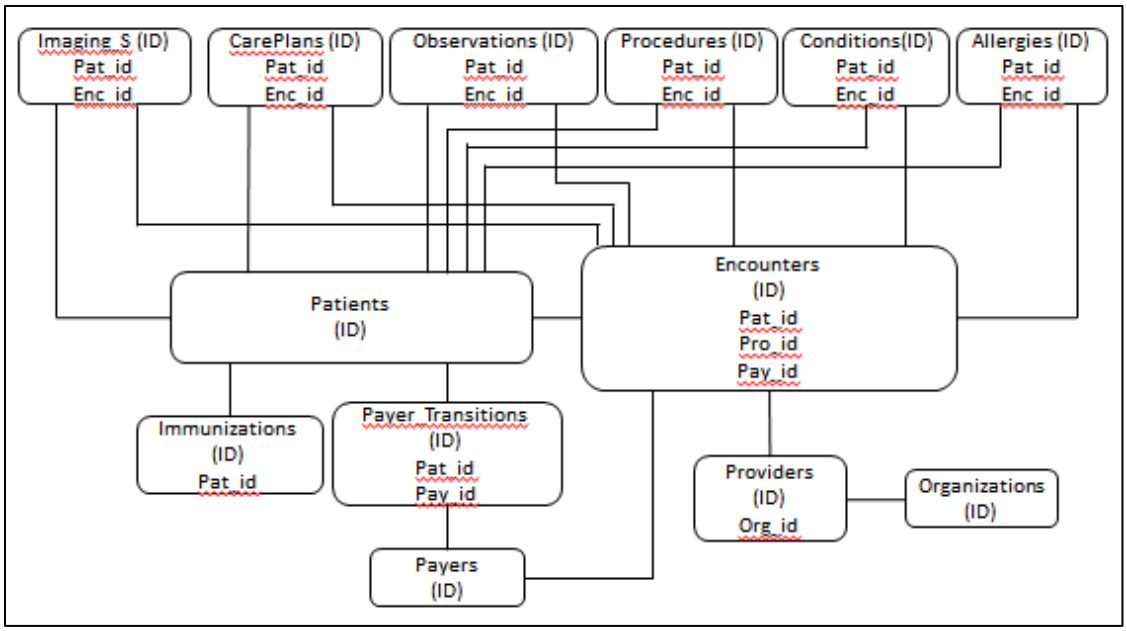



Figure 25. Re-engineered SyntheticMass database schema

## Data Collection – Security Warnings and Privacy Violation Suspects

*Application Code*

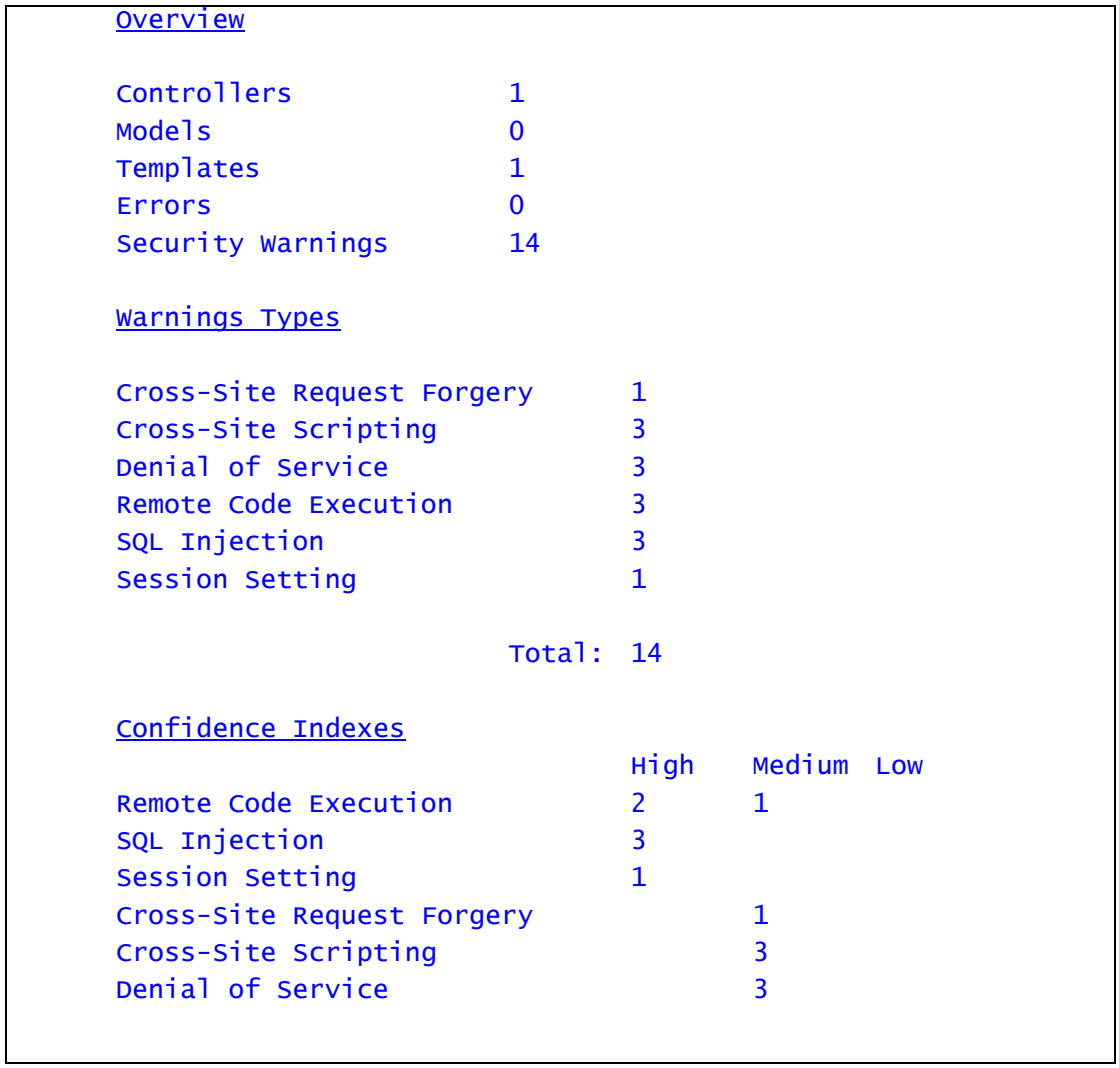
The SCA tool, Brakeman, offers the following security warning types (Brakeman, n.d.): Attribute restrictions, authentication, basic authentication, command injection, cross-site request forgery, cross-site scripting, cross-site scripting (Content tag), cross-site scripting (JSON), dangerous evaluation, dangerous send, default routes, denial of service, divide by zero, dynamic render paths, file access, format validation, information disclosure, mail link, mass assignment, remote code execution, remote execution in YAML.load, session manipulation, session settings, SQL injection, SSL verification bypass, unsafe de-serialization, un-scoped find, unsafe redirect,

and weak hash. *YAML stands for YAML Ain't Markup Language that is a data serialization language (Eriksson & Hallberg, 2011).

When the SCA tool runs against Ruby on Rails application code, it generates several pieces of information including an overview, security warning types, and confidence indexes for those security warnings. The overview consists of controllers, models, templates, errors, and security warnings along with confidence indexes.

*Detected Security Warning Data*

According to Park et al. (2021), the following is the summary of the vulnerability posture of the chosen application that was generated by the execution of the SCA scanner against it.



Overview

| | |
|---|---|
| Controllers | 1 |
| Models | 0 |
| Templates | 1 |
| Errors | 0 |
| Security Warnings | 14 |

Warnings Types

| | |
|---|---|
| Cross-Site Request Forgery | 1 |
| Cross-Site Scripting | 3 |
| Denial of Service | 3 |
| Remote Code Execution | 3 |
| SQL Injection | 3 |
| Session Setting | 1 |
| Total: | 14 |

Confidence Indexes

| | High | Medium | Low |
|---|---|---|---|
| Remote Code Execution | 2 | 1 | |
| SQL Injection | 3 | | |
| Session Setting | 1 | | |
| Cross-Site Request Forgery | | 1 | |
| Cross-Site Scripting | | 3 | |
| Denial of Service | | 3 | |

Figure 26. Brakeman scan results

*Data from Database*

Mitre's SyntheticMass generated 1,173 electronic patient records and the following are the sensitive information scan results that include surnames, given names, social security numbers, date of births, postal addresses, credit card numbers, and bank account numbers:

***a.*** Surnames

The name is divided into surnames and given names for simplicity. The basis of last names for comparison or pattern matching is from the 2010 Census (Census, 2010). The input

data for names are scanned against the most popular top 99 Census surnames in 2010, using the Jaro-Winkler formula.

*Scan method(s):*

Jaro-Winkler algorithm – Measuring the distance between two strings as it provides insights on their similarity

*Scan result(s):*

33 records were found with a strength of ≥ 0.90.

No errors. See algorithm in Appendix E, scan results in Appendix L.

***b.*** Given Names

The basis of first names for comparison or pattern matching is from the 2010 Social Security Administration that contains 50 first names for males and 49 first names for females (Social Security Administration, n.d.). Note that both *dflnamecen* and *dflname* are character vectors. We need to create a data frame for each for a later calculation. This is a Cartesian product of the two vectors, though, which results in a big size of records. Hence, Census data is limited to 99. So, now, the size of data1 is: 99 (SSA data) x 1,173 (Patient data) = 116,127.

*Scan method(s):*

Jaro-Winkler algorithm – Measuring the distance between two strings as it provides insights on their similarity

*Scan result(s):*

33 records were found with a strength of ≥ 0.90.

No errors. See algorithm in Appendix F, scan results in Appendix M.

***c.*** Social Security Numbers (SSNs)

As for SSNs, DoBs, addresses, and it follows the pattern matching scheme which is the universal tool matching any input strings using regular expressions, using the SSN numbering scheme (Social security, n.d.). The similarity level is 0 (low) or 1 (high) if the input data present just alphanumeric letters with spaces or commas or periods for postal addresses, and they are validated with regular expression patterns as follows:

*Scan method(s):*

*SSN pattern:*

"^?!(000)\\d{3}[ -]?!(00)\\d{2}[ -]?!(0000)\\d{4}"

*Scan result(s):*

1,173 records found.

No errors. See algorithm in Appendix G, scan results in Appendix N.

***d.*** Date of Births (DoBs)

As for DoBs, it follows the pattern matching scheme which is the universal tool matching any input strings using regular expressions. The similarity level is 0 (low) or 1 (high) if the input data present just alpha numeric letters with spaces or commas or periods for postal addresses, and they are validated with regular expression patterns as follows:

*Scan method(s):*

*DoB pattern:*

"^(0?[1-9]|1[0-2])/(0?[1-9]|[1-2][0-9]|3[0-1])/((19[0-9]|20[0-1])[0-9])$"

*Scan result(s):*

1,173 records found.

No errors. See algorithm in Appendix H, data results in Appendix O.

***e.*** Postal Addresses

The following regular expressions are adopted from the source provided by FrostedSyntax in 2015.

*Scan method(s):*

*Postal address pattern:*

*Street address*: "\\d+[ ](?:[A-Za-z0-9.-]+[ ]?)+(?:Avenue|Lane|Road|Boulevard|Drive|Street|Ave|Dr|Rd|Blvd|Ln|St)\\.?"

*City*: "(?:[A-Z][a-z.-]+[ ]?)+"

*State*:

"Alabama|Alaska|Arizona|Arkansas|California|Colorado|Connecticut|Delaware|Florida|Georgia|Hawaii|Idaho|Illinois|Indiana|Iowa|Kansas|Kentucky|Louisiana|Maine|Maryland|Massachusetts|Michigan|Minnesota|Mississippi|Missouri|Montana|Nebraska|Nevada|New[ ]Hampshire|New[ ]Jersey|New[ ]Mexico|New[ ]York|North[ ]Carolina|North[ ]Dakota|Ohio|Oklahoma|Oregon|Pennsylvania|Rhode[ ]Island|South[ ]Carolina|South[ ]Dakota|Tennessee|Texas|Utah|Vermont|Virginia|Washington|West[ ]Virginia|Wisconsin|Wyoming|AL|AK|AS|AZ|AR|CA|CO|CT|DE|DC|FM|FL|GA|GU|HI|ID|IL|IN|IA|KS|KY|LA|ME|MH|MD|MA|MI|MN|MS|MO|MT|NE|NV|NH|NJ|NM|NY|NC|ND|MP|OH|OK|OR|PW|PA|PR|RI|SC|SD|TN|TX|UT|VT|VI|VA|WA|WV|WI|WY"

*Scan result(s):*

4 out of 1,173 records were found.

No errors. See algorithm in Appendix I, scan results in Appendix P.

***f.*** Credit Card Numbers (CCNs)

According to Addison (2011), there are eight types of credit cards and their assigned numbers that are called major industry identifiers (MII) are as shown in Table 20:

Table 20. MII and Industry Assignment

| MII | Industry Assignment |
|---|---|
| 1, 2 | Airlines (Diners Club enRoute) |
| 3 | Travel & entertainment (non-banks such as American Express, Diner's Club, JCB, and Carte Blanche) |
| 4 | Banking & financial (Visa, Switch, and Electron) |
| 5 | Banking & financial (Master card and Bankcard) |
| 6 | Merchandising & financial (Discover card, laser, solo, switch, and China UnionPay) |
| 7 | Petroleum |
| 8 | Telecommunications |
| 9 | National assignment |

Pattern matching the credit card is a little complicated, as there are many different types of credit cards and the validation of the card number is difficult. As a result, for this study, the numbers were validated using Luhn's algorithm (a.k.a. modulus 10 algorithm) rather than pattern matching. Many e-commerce sites use the Luhn algorithm as the first line of defense to validate a wide range of credit card numbers (Hussein et al., 2013, p. 269). Thus, the author thinks such a method is sufficient for this study and the validation of card identification numbers was limited to the popular four credit cards in the U.S.: American Express, Visa, Master Card, and Discover. According to Addison (Addison, 2011), the first few digits of the card numbers can always be used to identify the credit card type. In the U.S., one tends to identify the card types along with the card number length of each as follows:

- Visa: 4xxxxxx --- 16-digit long, or 13-digit long (old cards)

- MasterCard: 5xxxxx --- 16-digit long (It begins with 51 through 55 or 2221 through 2720. In this study, only 51 through 55 have been used.)
- American Express: 34xxxx, 37xxxx --- 15-digit long
- Discover: 6011xx, 65xxxx --- 16-digit long

If the MII is 9, the next three digits of the issuer identifier are the 3-digit country codes defined in ISO 3166 (Malviya, n.d.) and the remaining final two digits of the issuer identifier can be defined by the national standards body of the specified country in whatever way wished.

*Scan method(s):*

*Luhn's algorithm* – Many e-commerce transactions use the Luhn algorithm as the first line of defense to validate a wide range of credit card numbers (Hussein et al., 2013, p. 269).

*Scan result(s):*

12 records found.

No errors. See Algorithm in Appendix J, scan results in Appendix Q.

***g.*** Bank Account Numbers (BANs)

When it comes to your bank account number, it essentially consists of three numbers in the U.S., namely, a bank routing number, a personal account number, and your sequential check number. The routing number is sometimes called the Routing Transition Number (RTN), but more predominantly American Bankers Association (ABA) number (U.S. Bank, n.d.). In this study, we focus on the personal bank account number. In short, the basis of BANs is the first 9 digits for routing number, next 10-12 digits (3$^{rd}$ through 17$^{th}$ are the actual placement of the digits) for account number, and final 3-4 digits for check number which depends on each bank's

system. Thus, the actual input data present nine digits in their correct places for ABAs and 3 to 17 digits for BANs.

*Scan method(s)*:

*ABA routing number and bank account number patterns:*

ABA routing number: ^\\d{9}

Bank account number: ^\\d{3,17}

*Scan result(s):*

None found.

No errors. See Algorithm in Appendix K, data results in Appendix R.

*Detected Privacy Violation Data*

The detected privacy violation suspect information on the data is summarized as follows:

The total number of the entire raw test dataset is 1,173.

The result of the pattern matching operations for each privacy data element is included in a vector named 'H' as follows in R Studio:

```
$ H

[1]   33   42 1173 1173    4   27    0
```

Figure 27. The result of pattern matching operations

The barplot of the result is performed as shown below.

```
$ barplot(rev(as.matrix(H)),main="Privacy Violation Suspects",xlab="
# of Records Matched",ylab="Sensitive Data Elements",names.arg=c("BA
N","CCN","ADDR","DOB","SSN","FNAME","LNAME"),cex.names=0.7,horiz=T,l
as=1)
```

Figure 28. The barplot of the result using R

The following depicts the result of each privacy information pertaining to ePHI.


Figure 29. Detected privacy violation suspect information

Figure 30 describes the result of each privacy information pertaining to ePHI.

```
# LName = 33 records detected out of 1,173
# FName = 42 records detected out of 1,173
# SSN = 1,173 records detected out of 1,173
# DoB = 1,173 records detected out of 1,173
# Addr = 4 records detected out of 1,173
# CCN = 27 records detected out of 1,173
# BAN = 0 records detected out of 1,173
```

Figure 30. Results of privacy violation suspect information

Assumption: For the purpose of simplification, the “Name” field is separated into two fields - FName for the first name and LName for the last name - with the same severity level, and then later they are combined into one for sensitivity risk calculation.

## Risk Calculations

*Application Code*

Table 21 is the completed chart for the vulnerability risk assessment calculation.

Table 21 Security vulnerability risk calculation

| | | | Confidence Index ($h_j = 6$) | | | Conf. Index Value | Conf. Level | Vul. Risk Measure |
|---|---|---|---|---|---|---|---|---|
| | Severity Index | | $h_3 = 3$ | $h_2 = 2$ | $h_1 = 1$ | | | |
| | Security Warnings | $S_n$ | High | Med. | Weak | $j_i h_i$ | $C_i$ | $C_i S_i$ |
| 1 | Remote code execution | 7.8 | 2 | 1 | | 8 | 1.33 | 10.40 |
| 2 | SQL injection | 6.82 | 3 | | | 9 | 1.50 | 10.23 |
| 3 | Session setting | 4.3 | 1 | | | 3 | 0.50 | 2.15 |
| 4 | Cross-site request forgery | 0 | | 1 | | 2 | 0.33 | 0.00 |
| 5 | Cross-site scripting | 4.3 | | 3 | | 6 | 1.00 | 4.30 |
| 6 | Denial of service | 5.84 | | 3 | | 6 | 1.00 | 5.84 |
| | Total: | 29.06 | | | | 34 | 5.67 | 32.92 |
| | | | | | | | V = | 1.13 |

As shown above, in the scan results, 6 security warning types were generated along with their confidence indexes. The confidence level is built following the Formula (4). For example, the "remote code execution" warning type has 2 highs and 1 medium, and three confidence levels are ranging from 3 to 1.

$$C_1 = \frac{\sum_{i=1}^{m} l_i n_i h_i}{\sum_{i=1}^{m} l_i} = \frac{(3 * 2 * 1) + (2 * 1 * 1) + (1 * 0 * 0)}{3 + 2 + 1} = 1.33$$

To calculate for vulnerability risk measure for each security warning type following the Formula (5). Using the same example as above,

$$V_1 = C_1 S_1 = (7.8 * 1.33) = 10.40$$

Finally, to calculate for the aggregated vulnerability risk score following the Formula (6):

$$V = \frac{\sum_{i=1}^{n} C_i S_i}{\sum_{j=1}^{n} S_i} = \frac{(7.8 * 1.33) + (6.82 * 1.5) + (4.3 * 0.5) + (0 * 0.33) + (4.3 * 1.0) + (5.84 * 1.0)}{7.8 + 6.82 + 4.3 + 0 + 4.3 + 5.84}$$

$$= 1.13$$

*Data from Database*

The calculation for privacy violation risk is twofold: Individual calculation for data elements and their aggregation. Accordingly, the individual and aggregated level assessment and analysis are carried out. The results shown in Table 22 indicate the number of records detected for privacy violation suspects under the similarity index. The value, $c$, of the Formula (9) is the indicator of detection with the boolean value of detected (1) or non-detected (0). Hence, each data element's severity index, $s$ , is obtained by applying the Formula (9) with its severity value, $W$. Note that the detected values, 33, 42, for data elements "LName" and "FName" respectively are combined into 75 for the data element "Name" for calculation.

Table 22 Privacy violation risk calculation

| Sensitive data elements | Severity index $s_n$ | Similarity index $M_n$ | Sensitivity value $P_n$ | Sensitivity Risk Measure $S_n$ |
|---|---|---|---|---|
| Name | 2 x 1 = 2 | 75 / 1,173 = 0.06393 | 0.12786 | 0.008524 |
| SSN | 4 x 1 = 4 | 1,173 / 1,173 = 1 | 4.00000 | 0.266667 |
| DoB | 3 x 1 = 3 | 1,173 / 1,173 = 1 | 3.00000 | 0.200000 |
| Address | 2 x 1 = 2 | 4 / 1,173 = 0.00341 | 0.00682 | 0.000455 |
| CCN | 4 x 1 = 4 | 27 / 1,173 = 0.02302 | 0.09208 | 0.006139 |
| BAN | 3 x 0 = 0 | 0 | 0.00000 | 0.000000 |
| GSV or ($g$): | 15 | | | |
| | | | $S_n$ = | 0.481785 |

The individual similarity index, $M$, is obtained by checking how many records are the exact match out of the total records, using the Formula (10). Each data element's sensitivity value is then the product of sensitivity severity index and its similarity index, and the normalized individual sensitivity level $S$ can be obtained by applying the Formula (11) and GSV noted in $g$ in Formula (12). The final aggregated sensitivity risk measure is obtained by the Formula (13).

## Risk Analysis and Assessment

*Application Code*

$V$ would be the final score that is resulted from the software application risk assessment and the following describes what it means with it:

The resulted risk score: 1.13

This $V$ score falls between 1 and 1.7. Based on Yun's chart (2018) that is captured in Table 5 - Vulnerability assessment measures, its risk is relatively low and less likely exploited. The application code when put in operation may give rise to certain security influences on individuals. In this condition, it needs to adopt certain preventive measures based on personal information risk assessment.

*Data from Database*

$S$ would be the final score that is resulted from the database risk assessment and the following describes what it means with it:

The resulted risk score: 0.482

This $S$ score falls between 0.33 and 0.56, which falls into the category of weak-sensitive. The corresponding prescriptive description (Yun, 2018) of it is as shown in Table 10 – Sensitivity assessment measures: The PHI after utilization gives rise to sensitivity influences on certain personal aspects. In this condition, it is needed to adopt preventive measures based on personal information risk assessment.

## Summary

The goal of implementing VS-HRA was to integrate static code analysis and pattern recognition algorithms into the risk assessment processes. VS-HRA tackled the GitHub server as a data source as it is one of the popular application code-repositories, and Mitre corporation's

SyntheticMass as a data source for the database as the application obtained from GitHub.com typically does not contain data. Both SCA and pattern recognition algorithms generated the data that was needed: Security warnings of the chosen application code and privacy violation suspect counts out of the 1,173 electronic patient records from SyntheticMass. The results of the application code and data scanning were converted into numerical measures and calculated with the given mathematical models. In the end, VS-HRA successfully integrated the results of SCA security warnings and privacy violation suspect counts into the risk assessment processes, which generated the vulnerability risk score (1.13) and sensitivity risk score (0.482) and interpreted as low risks.

# CHAPTER 5

## SUMMARY, RECOMMENDATIONS, AND CONCLUSIONS

This chapter presents a summary of the findings derived from the research study, which aimed to build a new risk assessment framework to assess security and privacy risks for cloud-based health information applications and databases. It also made recommendations for future research projects, followed by conclusions and contributions to the scientific community.

The study was essentially conducted in two phases. First, it developed a proposed risk assessment framework that is specific for health information applications in the cloud. Second, it implemented the newly developed risk assessment framework against actual cloud-based health information applications and databases. The cloud-based health information applications were selected through the open-source health information applications from GitHub.com and a simulated health information database from SyntheticMass by Mitre corporation. The open-source health information applications were selected using a decision tree method and the simulated database using a purposive sampling method.

### Summary of Findings

This study's findings are summarized below following research questions proposed in Chapter 1. The research questions are divided into four categories: New risk assessment development strategy, transformation methods on warnings and suspects, mathematical models,

and feasibility study. The following sub-sections analyzed and discussed the research findings according to each of those categories:

*Research Question 1 – Development Strategy*

The goal of the first research question was to find a development strategy that can yield such a risk assessment framework that concerns specifically cloud-based health information applications and databases separately. The literature review reveals that TARA is more reliant upon business operations, use cases, and end-to-end workflows, and not the application code. ISO 27001, NIST-RMF, and OCTAVE, in general, look for assets, identify relevant threats to the assets, and see if any controls are associated with the threats. However, none of them walk through the business application code in the way shown in this research study. Although a business application can be considered an asset, it looks for vulnerabilities as a whole system, not working on the actual application code. They tend to point to risks that exist in the operation of the business information systems and their controls. For the case of FAIR, it relies on loss event frequency and loss magnitude, which forms a binary tree approach, but ultimately it relies on assets, threats, and controls, as well. To be used with health information applications, all of these require extensive customization.

Thus, the development approach used in this research study was to utilize one of the software engineering practices that separate business processes for software application code from data in the database so that the risks associated with them can be addressed separately in the name of security vulnerabilities and potential privacy incidents. The separation of concerns provides many benefits in software engineering including reduced complexity, improved reusability, and simpler evolution (Tarr et al., 1999, p. 112). Within the context of this research,

such a separation of concerns provided the flexibility of choosing health information applications in the cloud as well as selecting databases for implementation to perform a feasibility study with.

This research study adopted the VS-PIRA II model to reflect the separation of concerns, specifically separating databases from applications. The model provided the foundation of working on business applications and databases separately which resulted in the newly named model, VS-HRA. The major differences between VS-PIRAII and VS-HRA lie in the approach of obtaining data. VS-PIRA II focuses on data collected from surveys, interviews, and meetings whereas the VS-HRA model employed health information applications as data as well as separate textual data in databases.

To obtain data from health information applications, this study utilized static code analysis (SCA) tools while it used pattern-recognition algorithms to extract the data from databases. The SCA tool examined vulnerabilities of applications and generated security warnings for those security vulnerabilities. For databases, this study used several pattern recognition algorithms to identify privacy violation suspects out of the textual data.

For the actual implementation of the model, this research study used the programming language, Ruby-on-Rails due to the author's familiarity, which was suitable for cloud-based software development, and its SCA tool is available in Brakeman for Ruby on Rails. Databases, on the other hand, are typically proprietary, and there are few options for gaining direct access to data stored in them other than using a simulated database. Thus, this research study developed several pattern recognition algorithms to investigate potential privacy violations on the simulated data.

Because the quality of data is linked to the quality of the tools, tool dependency is an issue. It can be detrimental if SCA and pattern recognition tools are not available for some

programming languages. Even if they do exist, the quality of the tools may be inadequate and problematic. As a result, anyone who takes this approach should keep this limitation in mind.

Furthermore, because this approach is platform-agnostic, it can be difficult to implement because each platform has its own set of constraints. For example, when the application environment changes from one platform to another like from Azure to AWS, the new hosting platform may present previously unknown challenges and risks. From a perspective of a database, it would not be able to adequately assess the risks if the new environment may present non-compatible textual data formats or introduce previously non-existent garbage characters.

Nonetheless, the VS-HRA provided an adequate model for detaching applications from databases and working on them independently with the SCA tools and pattern recognition algorithms for cloud-based health information application code and textual data, respectively.

*Research Question 2 – Transformation Methods on Warnings and Suspects*

The goal of the second research question was to find a method of transforming the qualitative measures generated from software tools such as security warnings and privacy violation suspects into numeric measures for risk calculation. To convert qualitative security warnings and privacy violation suspects into numerical measures, a reasonable firm ground must have been established from authoritative sources of references for a security warning and privacy violation translation. As discussed in literature reviews, NIST-RMF, TARA, and FAIR use high, medium, and low as metrics, or variations on those. All of these require human interpretation and lack the granular level of meaningful scientific metrics for which this research study strives.

The Common Vulnerabilities Exposes (CVE) scores from CVSS were useful in establishing them as an authoritative source of risk translation for applications. CVSS is a reliable source of risk scoring systems for them. According to the Forum of Incident Response

and Security Teams (FIRST) website, CVSS was created to promote an open and universal vulnerability scoring system. Prior to the advent of CVSS, many industry security vendors and non-profit organizations developed systems to rank information system vulnerabilities, but they were limited in terms of coverage and interoperability among them (Schiffman, 2005). The CVE classifies vulnerabilities and scores them using CVSS and provides a prioritized list of vulnerabilities to an application (Imperva, n.d.).

Every application presents different CVE scores for vulnerability as the security posture of each application is different from the other. That is, no fixed scores were available for each vulnerability. To address the shortcomings of CVE scores, this research study used the mean values of CVE scores available for the same vulnerability or category across different applications for the severity level of security warnings for this study.

As for databases, the NC state university's data policy was found as the authoritative source of reference and utilized as a basis for risk translation for data. The university's data policy provided a severity indicator for each specific sensitive data element using a color-coded sensitivity scheme that breaks the sensitivity into five classification levels including purple, red, yellow, green, and white. To provide numeric measures for each of these classification levels, the study used the ENISA's ranking system.

Although this study found a reliable source of vulnerability scoring system, it found some challenges: 1) It was challenging to translate the CVE scores for the sample Ruby-on-Rail application that this study worked on; 2) A different SCA tool may provide a different set of security warnings that necessitate a need for a standardized set of security warnings out of SCA tools.

As for databases, pattern-recognition algorithms may not catch every single instance of privacy violation suspects. This study used a small set of data to determine the feasibility of implementation. However, when it comes to a real environment, the size of data matters. Especially for the name comparison, it needs to deal with a massive amount of data using the Cartesian product of the examined data set and the standard data set. Thus, data manipulation and data processing algorithms need to be improved.

While identifying areas of limitation for improvement, the study provided practical solutions for transforming security warnings and privacy violation suspects into numeric measures. This study developed the matching, mapping, and correlating (MMC) method for CVE scores to establish the severity index of applications. For the databases, it used the ENISA ranking system for the NC State University's data policy for severity classification levels. The MMC method as well as the ENISA method validated the soundness and adequacy of the scanner-based, data-driven risk assessment framework as well as their effectiveness in its implementation.

*Research Question 3 – Mathematical Models*

The goal of the third research question was to find a way that the mathematical models can incorporate severities and vulnerabilities for application code, and severities and privacy violation suspect into formulas while sustaining the co-existence of two separate paths of risk evaluation. Most of the risk assessment frameworks analyzed in the literature review used either the addition method (impact + likelihood) as shown in ISO 27001, the multiplication method (impact x likelihood) as shown in FAIR, or the matrix method (vertical line for impact and horizontal line for likelihood, and the crossing cell having values like high, medium, or low, or variations of it) as shown in NIST-RMF. TARA provided more advanced methods that contain a

list of pre-analyzed risk scores for each threat type and pre-assessed effectiveness scores for each remedy. Thus, when evaluating an asset's risk, it identified which threat types it is associated with, and which remedy types are associated with the threat types. Essentially, based on the strength of the remedies, risks are prioritized. Nonetheless, these risk assessment frameworks are not concerned with the separation of application code and database for their processes. They are all intertwined with each other and it is costly to separate them for customization.

Hence, the mathematical models for this research study used the separation of concerns approach found in software engineering. Such an approach provided consistency and cohesiveness between the VS-HRA and the mathematical models. The mathematical models included the severity values of security warning types and the confidence indexes along with the number of occurrences on the application code side. As for databases, the mathematical models included the severity values of privacy violation suspects and the similarity indexes along with the number of occurrences. As noted earlier, the severities of both application code and database stemmed from the CVE scores and the NC State's data policy values, respectively.

To perform risk calculation based on the reviewed risk assessment frameworks, it could have taken some time to finish the risk calculation as surveys, interviews, or meetings needed to be conducted to collect the data for calculation. Their calculation also could have resulted in different outcomes every time the evaluation is conducted with different assessors. The different assessors may have different assessment education, experiences, and techniques. Even with the same assessors, they may have different opinions each time depending on the situation. On the contrary, VS-HRA's mathematical models are relatively quick and can be run repeatedly resulting in the same outcome, or different when improved application code and more protected textual data exist. Besides, the VS-HRA mathematical models specifically focused on potential

security loopholes that may exist in every line of application code at a development time. Also, they focused on every data element that is sensitive by the definition of HIPAA and its privacy violation suspects of the textual data for the data persisted by their data transactions.

*Research Question 4 – Feasibility Study*

The goal of the fourth research question was to evaluate whether the proposed risk assessment framework is feasible. This research study conducted the implementation of VS-HRA as a feasibility study using the Objective-Key-Results (OKR) method. OKR is a way to measure if the objective is met by reviewing key results which can be numeric measures or activities (Chau, 2020).

**Application Code**. As for application code, the objectives were to 1) use decision-making process to find at least one open-source cloud-based health information application code for this research study, and 2) use the matching, mapping, and correlating (MMC) method to convert security warnings into measurable numeric values for at least 95% of security warning types.

Key Results:

1) The decision tree method generated a total of 9 trees using three different search criteria. Each tree resulted in the following repository candidates: Tree 1 = 2; Tree 2 = 2; Tree 3 = 959; Tree 4 = 156; Tree 5 = 959; Tree 6 = 16; Tree 7 = 8; Tree 8 = 2. Trees 3, 4, 5 are rejected due to the complexity of searching. Trees 1, 2, and 8 are rejected due to the complexity of installation. The remaining trees 6 and 7 were investigated for use. Each has 16 and 8 repositories, respectively. Tree 6 did not provide a workable repository. One workable repository was found in Tree 7 out of 8

repositories, using the 2nd criteria. This process can be a one-off solution, but for this research study, it was necessary and sufficient.

2) To convert security warnings into measurable numeric values, it was necessary to harmonize the terminologies between Brakeman's and CVE's terminologies. To minimize any discrepancy between the two, this research study developed the MMC method that utilized matching, mapping, and correlating techniques. Matching occurs when the two terms are equal. Mapping occurs when the two terms are different but connotatively equal. For example, "overflow" is caused by some numbers "divided by zero". Hence, the different terms "overflow" and "divided by zero" are connotatively equal. Also, correlating occurs when the two terms are contextually equal or close when compared to each other. For example, "dangerous evaluation" is similar to "command injection". Unauthorized users could enter malicious inputs for the user's data input screen, and the corresponding application code mistakenly interprets it as a real command. Thus, the "injected command" in the user dialogue box goes into the "dangerous evaluation" of its associated code. Using those three MMC rules, the results of the conversion are shown below in Table 23:

Table 23 MMC conversion success rate

| | # of successful conversion |
|---|---|
| Matching | 4 |
| Mapping | 10 |
| Correlating | 14 |
| No MMCs* | 4 |

Note: *No MMCs: No matching, mapping, nor correlating

The total of Brakeman's warning types was 32, but the MMC rules could not find fit for the remaining 4, which provided the success rate of 87.5%. Although it did not meet the target percentage, the given SCA-generated security warning types all fell

within 87.5%. Nonetheless, this is a future research area to improve the full harmonization scheme for accuracy.

**Data from Database**. As for data from a database, the objectives were to 1) use distance-based check to detect names that are at least 90% similar for the standard given names and surnames, 2) use pattern-recognition-based check to detect legitimate SSNs, DoBs, and postal addresses, and 3) use parity-check-digit-based check (a.k.a. Luhn test) to detect legitimate credit card numbers and bank account numbers.

Key Results:

1) For the distance-based check on detecting names, the study set the objective of meeting at least 90% similar to the standard names for the given names and surnames. 90% was set by default for the study. However, it can be configurable depending on the study requirements. For this study, 33 out of 1,173 surnames and 42 out of 1,173 given names were found with a similarity that is greater than or equal to 0.90 (90%) using the Jaro-Winkler algorithm. The distance-based check provided enough evidence that the algorithm worked against the textual name data for the potential privacy violation.
2) For the pattern-recognition-based check on detecting SSNs, DoBs, postal addresses, and bank account numbers, the study set the objective of meeting the legitimacy of the government-regulated formats. Using the known government-accepted formats, the study found: SSNs = 1,173; DoBs = 1,173; Postal Addresses = 4; Bank Account Numbers = 0. That means, the entire data given for SSNs and DoBs exactly follow the government-accepted patterns. While finding zero (0) matches for bank account numbers, the study also found only 4 legitimate postal addresses. This study found it

necessary to improve the robustness of the recognition method for the postal address. The reason being is that this study solely relied on pattern recognition for the postal addresses, but later found that the U.S. Postal Office provides APIs for the U.S. postal addresses. Using the APIs, the precision of the postal address recognition could be greatly improved.

3) For the parity-check-digit-based check (a.k.a. Luhn test) on detecting legitimate credit card numbers, the study set the objective of failing the validity check for the numbers. The study found: Credit card numbers = 27. As far as the validity check for the numbers is concerned, the Luhn test resulted in a satisfactory result.

While the methods mentioned above generated the data that was needed to examine the VS-HRA framework and mathematical models, several areas were identified as needing future research efforts. The following subsection addressed those areas.

## Recommendations for Future Research

- This research study isolated each sensitive data element to work on individually, but future research is needed to identify privacy violation suspects by combining one or more sensitive data elements that may lead to personally identifiable information in violation of HIPAA privacy regulations.
- This research study used an open-source SCA tool to extract security-warning messages for health information applications when it ran against the application code, and this research discovered one of the most popular cloud application programming languages in Ruby on Rails, through personal experience, and Brakeman provided SCA capabilities for the Ruby on Rails applications. However, many SCA tools exist for other programming languages. As

research, the VS-HRA model can be re-evaluated with other programming languages as can the performance of Brakeman.

- This research study dealt with several issues when the CVE scores were converted into fixed numeric values for Brakeman's security warning types. The first issue with the conversion was that there are no governing bodies to promote, develop, and maintain the terminology harmonization for security warning types. This research found it necessary to build one. Another issue with the conversion was to identify a method of converting descriptive security warnings into numeric measures based on the CVE scores. This researcher adopted a method that Haufe et al. (2016) used and took one step further by breaking it down into three categories: matching, mapping, and correlating (MMC method). Although the MMC method was systematic, it could have been improved by adopting a more rigorous scientific process. The last issue with the conversion was the quality of the results. The method used was to take the values of the same category and then the mean value of the values. Future research can investigate an improved method of making a quality conversion instead of solely relying on mean values.
- This research study was challenged with pattern-matching algorithms for surnames and given names. The basis of surnames for pattern matching is from the 2010 Census (Census, 2010) whereas the basis of given names is from the 2010 Social Security Administration (Social Security Administration, n.d.). The textual privacy data for names were scanned against the most popular top 99 Census surnames in 2010 with the most popular top 50 male given names and 49 female given names from SSA, using the Jaro-Winkler algorithm. As this research was intended to check the feasibility of this proposed risk assessment framework, the records for comparison were limited to the first 99 records combined with male and

female names. The reason for not taking the entire records was the limitation of the pattern matching method. The Cartesian product of pairs for checking the similarity with the health information patient names resulted in 1,000 x 1,173 = 1,173,000 comparison records. Future work is needed to improve the algorithm or investigate more efficient methods as the input data could increase exponentially.

- This research study concentrated on a distance-based name matching method for detecting privacy violations and set the distance to 90% by default. However, many techniques exist for determining similarity between names in natural language processing (NLP) such as Jaccard similarity and cosine similarity (Sowmya et al., 2018). Future research is needed to investigate the accuracy and precision of name matching algorithms from both a syntactic and semantic standpoint.
- This research study recognized a need of working with postal addresses. The SSNs, DoBs, and postal addresses for comparison work relied on the pattern recognition schemes, using regular expressions. Among these three sensitive data elements, postal addresses could increase their precision by working with the U.S. Postal Service (USPS, n.d.) in the future. The U.S. Postal Service provides application programming interface (API) information for actual implementation. Many commercial companies already use the service by integrating the API into their services such as UPS (UPS, n.d.).
- Like SSNs, DoBs, and postal addresses, the bank account numbers in this research work also relied on the pattern recognition schemes, using regular expressions, for comparison. As routing numbers consist of three segments: Federal Reserve Routing Symbol (first 4 digits), the Financial Institution Identifier (the next 4 digits), and a check digit number, at least the accuracy of this detection can be improved by adopting the validation of routing numbers

using the check digit method (RN, n.d.). The account numbers vary from one financial institution to another financial institution. So, for all practical purposes, this study left the account numbers intact.

- This research study discovered an opportunity to improve the detection accuracy of credit card numbers as well as bank account numbers. Luhn's algorithm was used to scan the given credit card numbers. This parity-digit check method found the given numbers in errors if they are not checked. In other words, the errored numbers mean they are likely safe as they are not checked. However, the accuracy of this detection can be improved with the integration of other registries such as the American national standards institute that issues Bank Identity Numbers (BINs) and commercial databases maintaining the BINs (Merritt, n.d.).

## Practical Implications

The decision tree method required extensive manual operations when it was utilized to select the Ruby on Rails application out of millions of code repositories that exist in GitHub. For example, to test and identify whether a given application can be executed and can be usable for the Brakeman tool within the implementation environment, this study worked on them one by one by hand, which resulted in low efficiency. However, in a real environment, this manual operation can be removed as one has to deal with a given application, and thus there is no need for sifting processes. For this research study, it was effective enough in terms of removing unfitted repositories to arrive at one that is fit for the implementation of VS-HRA.

## Conclusions

The findings from the actual implementation for the research study provided a possibility of enabling healthcare entities to drill down each security and privacy risk separately, and relatively quickly by automating the processes and procedures that could increase the efficiency

of risk assessment. The software scanner tool-generated data provided invaluable insights at an application development time on the security and privacy posture of the cloud-based health information applications and their databases. To be more effective and efficient in pursuing this way of risk assessment methods, it essentially requires a more systematic approach to convert security warnings for code and privacy violation suspects into measurable numeric risk values. Also, integrating all the parts of the models into an entire risk assessment expert system requires thoughtful integration engineering efforts.

**Contribution(s) of the Study to the Scientific Community**

- This research provided a possibility of usage of various warning data generated from SCA and pattern-recognition algorithms as replacement of human surveys, interviews, and meetings for risk assessment data in health information applications.
- This study provided a mechanism of building relationships between different terminologies using the MMC method that can be a viable option for the risk assessment methods.
- The shift from qualitative assessment to the quantitative assessment provided the foundation of more repeatable and scientific capabilities. This shift ultimately could help to assess the risks in a more automated fashion.
- This study focused on healthcare industry-specific processes for risk assessment framework. However, the framework laid the groundwork for other industries' business applications to be used for risk assessment.

# REFERENCES


Addison, D. (2011, March 29). Anatomy of a credit card number and the utility of the BIN. Retrieved from: http://www.dirigodev.com/blog/ecommerce/anatomy-of-a-credit-card-number/

Alder, S. (2019, March 19). *The HIPAA risk analysis: Guidance and tools for HIPAA covered entities and business associates*. HIPAA Journal. Retrieved from https://www.hipaajournal.com/hipaa-risk-analysis-guidance-tools/

Alder, S. (2020, June 26). *What is the relationship between HITECH, HIPAA, and electronic health and medical records?* HIPAA Journal. Retrieved from https://www.hipaajournal.com/relationship-between-hitech-hipaa-electronic-health-medical-records/

Appaloosa Store. (2018, April 5). String similarity algorithms compared. Retrieved from: https://medium.com/@appaloosastore/string-similarity-algorithms-compared-3f7b4d12f0ff

Biscoe, C. (2019, March 12). How long does an ISO 27001 risk assessment take? Retrieved from https://www.vigilantsoftware.co.uk/blog/how-long-does-an-iso-27001-risk-assessment-take

Bjorhus, J. (2014, February 17). *Feb. 13: Target breach started as phishing expedition*. Star Tribune. Retrieved from https://www.startribune.com/feb-13-target-breach-started-as-phishing-expedition/245226831/?c=y&page=2

Brakeman. (n.d.). Introduction to Brakeman. Retrieved from https://brakemanscanner.org/docs/introduction/

Brook, C. (2020, December 1). *What is a health information system?* Digital Guardian. Retrieved from https://digitalguardian.com/blog/what-health-information-system

Brunschwiler, C. (2013, April 9). Lean risk assessment based on OCTAVE Allegro. Retrieved from https://blog.compass-security.com/2013/04/lean-risk-assessment-based-on-octave-allegro/

Caralli, R., Stevens, J.F., Young, L.R. & Wilson, W.R.. (2007, May). Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process. doi:10.1184/R1/6574790.v1

Castello, J. (2018, October). What is ruby on rails and why is it useful? Retrieved from https://www.rubyguides.com/2018/10/what-is-ruby-on-rails/

Census. (2010). Frequently occurring surnames from the 2010 census. Retrieved May 5, 2020 from https://www.census.gov/topics/population/genealogy/data/2010_surnames.html.

Chang, L. (2011, July 6). Cloud computing – Availability issues and controls. Retrieved from http://www.slideshare.net/lylcheng88/cloud-computing-availability-issues-and-controls

Chau, M. (2020). OKR: Objectives and Key Results FAQs 2021.The official 7Geese blog. Retrieved January 21, 2021 from https://7geese.com/okr-objectives-and-key-results-faqs/#4.

Chen, D. (2012). *CN 102622379 A - Real Name Detection Method and Equipment. Abstract*. The Lens - Free & Open Patent and Scholarly Search. Retrieved from https://www.lens.org/lens/patent/CN_102622379_A

Cloud Standards Customer Council. (2017, February). *Impact of cloud computing on healthcare. (Version 2.0)*. Retrieved from http://www.cloud-council.org/deliverables/CSCC-Impact-of-Cloud-Computing-on-Healthcare.pdf

Codecondo. (2017, November 6). PHP vs Python vs Ruby: The battle of web programming languages. Retrieved from https://codecondo.com/the-battle-of-web-programming-languages/

Davis, J. (2019, July 23). The 10 biggest healthcare data breaches of 2019, so far. Retrieved from https://healthitsecurity.com/news/the-10-biggest-healthcare-data-breaches-of-2019-so-far

De Groot, J. (2020, December 1). *The history of data breaches*. Digital Guardian. Retrieved from https://digitalguardian.com/blog/history-data-breaches

Donnan, S. & Leatherby, L. (2019, July 23). *Globalization isn't dying, it's just evolving*. Bloomberg. Retrieved from https://www.bloomberg.com/graphics/2019-globalization/

El Arass, M., Souissi, N., & Tikito, I. (2017, April). Data lifecycles analysis: Towards intelligent cycle. Conference paper, April 2017. DOI:10.1109/ISACV.2017.8054938.

ENISA. (2012, December). Cloud computing: Benefits, risks, and recommendations for information security. Rev. B. Retrieved from https://resilience.enisa.europa.eu/cloud-security-and-resilience/publications/cloud-computing-benefits-risks-and-recommendations-for-information-security

Eriksson, M., & Hallberg, V. (2011). *Comparison between JSON and YAML for data serialization*. Retrieved from http://www.csc.kth.se/utbildning/kth/kurser/DD143X/dkand11/Group2Mads/Rapport_Malin_Eriksson_Viktor_Hallberg.pdf

Fox, A. & Patterson, D. (2014, March 4). Engineering software as a service. First edition.

GitHub. (2020). GitHub number of repositories. Retrieved from https://github.com/search

Goodin, D. (2012, November 7). Virtual machines used to steal crypto keys from other VM on same server. Retrieved from http://arstechnica.com/security/2012/11/crypto-keys-stolen-from-virtual-machine

GS1. (n.d.). How to calculate a check digit manually. Retrieved from: https://www.gs1.org/services/how-calculate-check-digit-manually

Haufe, K., Brandis, K., Colomo-Palacios, R., and Stantchev, V. (2016, October). Security management standards: A mapping. Conference on enterprise information systems / international conference on project management/conference on health and social care information systems and technologies, CENTERIS / ProjMAN / HCist 2016, October 5-7, 2016.

HHS Health Information Privacy. (n.d.). Guidance regarding methods for de-identification of protected health information in accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule. Retrieved from https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html

HHS. (2018, August 17). *Audit protocol.* HHS.gov Health Information Privacy. Retrieved from https://www.hhs.gov/hipaa/for-professionals/compliance-enforcement/audit/protocol/index.html

HIMSS. (2012, February). *Cloud security workgroup toolkit*. Retrieved February 7, 2021 from https://www.himss.org/sites/hde/files/d7/HIMSSorg/Content/files/PrivacySecurity/CS05_Cloud_Security_Top10_questions_FINAL.pdf.

HIMSS. (2020, August 14). *Sample Risk Assessment for Cloud Computing in Healthcare.* Retrieved December 14, 2020 from https://www.himss.org/resources/sample-risk-assessment-cloud-computing-healthcare.

Hold, D. (2019, March 25). HIPAA compliance in the cloud: Who's responsible? Retrieved from https://www.infosecurity-magazine.com/opinions/hipaa-compliance-cloud-1-1/

Huang, R. (2016, June 2). 9 of the best free data mining tools. Retrieved from https://www.springboard.com/blog/9-best-free-data-mining-tools/

Hussein, K.W., San, N.F.M., Mahmod, R., & Abdullah, M.T. (2013). Enhance Luhn Algorithm for Validation of Credit Cards Numbers. *International Journal of Computer Science and Mobile Computing*, 2(7), 262-272. Retrieved March 29, 2021 from https://ijcsmc.com/docs/papers/July2013/V2I7201373.pdf

Imperva. (n.d.). CVE vulnerability. Retrieved 4/16/2021 from https://www.imperva.com/learn/application-security/cve-cvss-vulnerability/

Ingram, M. (2013, April 11). *Two charts that tell you everything you need to know about the future of newspapers*. Gigaom. Retrieved March 29, 2021 from https://gigaom.com/2013/04/11/two-charts-that-tell-you-everything-you-need-to-know-about-the-future-of-newspapers/

ISO 27001. (n.d.). ISO/IEC 27000 family – Information security management systems. Retrieved from https://www.iso.org/isoiec-27001-information-security.html

Jenhani, I., Amor, N. B., and Elouedi, Z. (2008, January 20). Decision trees as possibilistic classifiers. Retrieved from https://core.ac.uk/download/pdf/82526051.pdf

KirkPatrickPrice. (2020, October 28). *What is HIPAA*. Retrieved from https://kirkpatrickprice.com/video/what-is-hipaa/

Kosutic, D. (n.d.). ISO 27001 risk assessment: How to match assets, threats, and vulnerabilities. Retrieved from https://advisera.com/27001academy/knowledgebase/iso-27001-risk-assessment-how-to-match-assets-threats-and-vulnerabilities/

Kosutic, D. (2014). ISO 31000 and ISO 27001 – How are they related? Retrieved from https://advisera.com/27001academy/blog/2014/03/31/iso-31000-and-iso-27001-how-are-they-related/

Leo, R. (2005). The HIPAA program reference handbook. CRC Press.

Lesser, R., Reeves, M., & Harnoss, J.D. (2016, July 26). *Saving globalization and technology from themselves*. BCG Henderson Institute. Retrieved from https://www.bcg.com/publications/2016/strategy-globalization-saving-globalization-technology-from-themselves.aspx

Levi, S. D. (2012, April 3). Cloud Computing: Understanding Security and Jurisdiction Issues. Retrieved from http://www.skadden.com/insights/cloud-computing-understanding-security-and-jurisdictional-issues

Mai, P.X., Goknil, A., Shar, L.K. Pastore, F. & Briand, L.C. (2018). Modeling security and privacy requirements: a use case-driven approach. Retrieved from https://core.ac.uk/download/pdf/287750542.pdf

Malviya, H. (n.d.). Anatomy of credit card numbers. Retrieved February 6, 2021 from https://dl.packetstormsecurity.net/papers/general/creditcard-anatomy.pdf.

Mangan, D. (2016, August 5). *Huge data breach at health system leads to biggest ever settlement*. CNBC. Retrieved from https://www.cnbc.com/2016/08/04/huge-data-breach-at-health-system-leads-to-biggest-ever-settlement.html

Mansourov, N. (n.d.). Ranking weakness findings. Retrieved from https://www.oasis-open.org/committees/download.php/62624/Ranking%20weakness%20findings.pdf

Martinez, L. (2008). CUSpider - PII scanning application. Retrieved from https://cuit.columbia.edu/content/cuspider-pii-scanning-application

Matai, D. K. (2011, April 13). *What is the key to survival in a constantly changing environment?* Business Insider. Retrieved from http://www.businessinsider.com/what-is-the-key-to-survival-in-a-constantly-changing-environment-2011-3

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology*. NIST. Retrieved from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Merritt, C. (n.d.). How to determine the issuing bank for a Visa Card. Retrieved January 19, 2021 from https://smallbusiness.chron.com/determine-issuing-bank-visa-card-67184.html

Microsoft. (n.d.). Microsoft exploitability index. Retrieved December 26, 2020 from https://www.microsoft.com/en-us/msrc/exploitability-index.

NC State U. (2019). Determining sensitivity levels for shared data Retrieved from https://oit.ncsu.edu/it-security/data-framework/determining-sensitivity-levels-for-shared-data/#personal

NIST. (n.d.). Computer security resource center. SP 800 series. Retrieved February 7, 2021 from https://csrc.nist.gov/publications/sp.

NIST NVD. (2017, October 19). National vulnerability database. CVSS v2.0

NIST SP 800-30. (2012, September). *NIST SP 800-30. Rev. 1. Guide for Conducting Risk Assessments.* https://doi.org/10.6028/NIST.SP.800-30r1.

NIST SP 800-37. (2018, December). Risk management framework for information systems and organizations. Revision 2.

Office of Civil Rights. (2003, May). *OCR Privacy Brief: Summary of HIPAA privacy rule.* HIPAA Compliance Assistance. Retrieved from https://www.hhs.gov/sites/default/files/privacysummary.pdf

Office of Civil Rights HIPAA. (2016). HIPAA privacy, security, and breach notification audit program. Retrieved from https://www.hhs.gov/hipaa/for-professionals/compliance-enforcement/audit/phase1/index.html

Office of the National Coordinator for Health Information Technology (ONC). (n.d.). *Security Risk Assessment (SRA) Tool*. HealthIT.Gov. https://www.healthit.gov/topic/privacy-security-and-hipaa/security-risk-assessment-tool

Park, D.B., Li, X., Shahhosseini, A.M., & Tsay, L-S. (2021). A Static Code Analysis-based, Mathematical Model-driven Vulnerability Risk Assessment Framework for Health Information Applications in Cloud, International Journal of Forensic Engineering and Management, Vol. 1, No. 2.

Pearce, J. (n.d.). Model-View-Controller pattern. Retrieved from http://www.cs.sjsu.edu/~pearce/modules/patterns/enterprise/presentation/mvc.htm

Raffaelli, R.L. (2020, January). *Reinventing Retail: The Novel Resurgence of Independent Bookstores*. Harvard Business School.

Rokach, L. & Maimon, O. (2015). Data mining with decision trees: theory and applications. 2nd edition. Chapter 1, pp 12-16.

RN. (n.d.). Routing number. Retrieved January 20, 2021 from https://www.routingnumber.com/

Shakeel, I. (2018, February 27). Top 10 database security tools you should know. Retrieved from https://resources.infosecinstitute.com/certification/top-10-database-security-tools-know/

Saucs. (2018). Retrieved from https://www.saucs.com/cve/CVE-2018-19274

Schiffman, M. (2005, June 7). A complete guide to the Common Vulnerability Scoring System (CVSS) v1 archive. Retrieved 4/16/2021 from https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198

Shantamurthy, D. (2011, July 13). OCTAVE risk assessment method examined up close. Retrieved from https://www.computerweekly.com/tip/OCTAVE-risk-assessment-method-examined-up-close.

Singh, S. & Gupta, P. (2014, July). Comparative study ID3, CART, and C4.5 decision tree algorithm: A survey. *International Journal of Advanced Information Science and Technology (IJAIST)*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.685.4929&rep=rep1&type=pdf

Skybox security. (2019). *Research report: 2019 vulnerability and threat trends*. Retrieved from https://lp.skyboxsecurity.com/rs/440-MPQ-510/images/Skybox_Report_Vulnerability_and_Threat_Trends_2019.pdf

Smith, P.R. & Sarfaty, R. (1993, May 1). *Creating a strategic plan for configuration management using computer-aided software engineering (CASE) tools*. OSTI.Gov. Retrieved from https://www.osti.gov/biblio/10160331

Social security. (n.d.). The SSN numbering scheme. Retrieved from: https://www.ssa.gov/history/ssn/geocard.html

Social Security Administration. (n.d.). Top names over the last 100 years: Popular names for births in 1919-2018. Retrieved May 5, 2020 from https://www.ssa.gov/cgi-bin/popularnames.cgi.

Software Engineering Institute. (n.d.). Operationally critical threat, asset, and vulnerability evaluation (OCTAVE). Retrieved from http://www.cert.org/resilience/products-services/octave/

Sowmya, V., Raju, M.S.V.S.B., & Vardhan, B.V. (2018). Analysis of lexical, syntactic and semantic features for semantic textual similarity. *International Journal of Computer Engineering and Technology (IJCET)*, *9*(5), 1–9. Retrieved 04/25/2021 from http://www.iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_9_ISSUE_5/IJCET_09_05_001.pdf

Statista. (2021, March 3). *Cybercrime: number of breaches and records exposed 2005-2020.* Retrieved from: https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/

SyntheticMass. (n.d.). Mitre: Synthetic patient and population health data that simulates the state of Massachusetts. Retrieved from: https://synthea.mitre.org/

Tariq, M.I. (2018, November 4). *Analysis of the effectiveness of Cloud Control Matrix for hybrid cloud computing.* NADIA | IJFGCN. Retrieved from: http://dx.doi.org/10.14257/ijfgcn.2018.11.4.01

Tarr, P., Ossher, H., Harrison, W., & Sutton, S. M. (1999). N-degrees of separation. *Proceedings of the 21st International Conference on Software Engineering - ICSE '99*, 107–119. Retrieved 4/18/2021 from https://doi.org/10.1145/302405.302457

TCS. (2012). The state of cloud application adoption in large enterprises. *The state of adoption of cloud applications (pp. 6-13)*. TATA consultancy services. Retrieved from https://www.tcs.com/content/dam/tcs/pdf/industries/global-trend-studies/cloud-study.pdf

TCS. (2012). The state of cloud application adoption in large enterprises. *Difference in cloud adoption across global industries (pp. 44-53)*. TATA consultancy services. Retrieved from https://www.tcs.com/content/dam/tcs/pdf/industries/global-trend-studies/cloud-study.pdf

Terdoslavich, W. (2016, February 23). *Security concerns continue amid cloud adoption*. InformationWeek. Retrieved from http://www.informationweek.com/strategic-cio/security-and-risk-strategy/security-concerns-continue-amid-cloud-adoption/d/d-id/1324413

UPS. (n.d.). Validate an address. Retrieved January 20, 2021 from https://www.ups.com/address_validator/search?loc=en_US.

U.S. Bank. (n.d.). Find your U.S. Bank checking account routing number. Retrieved from: https://www.usbank.com/bank-accounts/checking-accounts/checking-customer-resources/aba-routing-number.html

USPS. (n.d.). Address information: USPS Web Tools – Application Programming Interface User Guide, version 6.0 (07/31/2020). Retrieved January 20, 2021 from https://www.usps.com/business/web-tools-apis/address-information-api.htm.

VA Monograph. (2018, September 21). Retrieved from https://www.va.gov/VA_Monograph_2019.

Vanderburg, E. (n.d.). Information security compliance: ISO 27000. Retrieved from https://www.tcdi.com/iso-27000-certification-history-overview/#targetText=The%20ISO%2027000%20series%20provides,www.27000.org).&targetText=Communications%20and%20operations%20management%20%E2%80%93%20management,controls%20in%20systems%20and%20networks.

Veracode. (n.d.). *SQL Injection. Vulnerabilities and SQL injection prevention*. Retrieved from https://www.veracode.com/security/sql-injection

Violino, B. (2010). IT risk assessment frameworks: Real-world experience. *CSO Online*. Retrieved from http://www.csoonline.com/article/2125140/metrics-budgets/it-risk-assessment-frameworks--real-world-experience.html

Violino, B. (2019, July 19). *What is PaaS? Platform-as-a-service explained.* InfoWorld. Retrieved from https://www.infoworld.com/article/3223434/what-is-paas-software-development-in-the-cloud.html

VistA. (n.d.). Retrieved from http://worldvista.org/AboutVistA

Woods, A. (2018, April 6). Name validation regex for people's names. Retrieved from: https://andrewwoods.net/blog/2018/name-validation-regex/

Woody, C. (2006). *Applying OCTAVE: Practitioner's report*. Retrieved from http://repository.cmu.edu/sei/395/

Yun, H.B. (2018, June) *A Study on Risk Assessment Model Based Personal Information Protection in Smart Cities*, Doctoral dissertation, June, Wonkwang University, South Korea. Retrieved May 6, 2020 from http://wonkwang.dcollection.net/public_resource/pdf/200000105456_20201103123225.pdf.

Zack, W.H. & Kommalapati, H. (2011, October 3). The SaaS Development Lifecycle. InfoQ. Retrieved from https://www.infoq.com/articles/SaaS-Lifecycle/

# APPENDICES

# APPENDIX A: TOOL INSTALLATION

Perform the cloning by:

```
$ git clone https://github.com/siteslave/his-rpc

Cloning into 'his-rpc'...
remote: Enumerating objects: 66, done.
remote: Total 66 (delta 0), reused 0 (delta 0), pack-reused 66
Unpacking objects: 100% (66/66), done.
```

3) Using the SCA tool, perform the SCA scanning for the selected application.

Perform the scanning for the selected apps by issuing a command as follows to execute Brakeman.

```
$ git clone https://github.com/presidentbeef/brakeman.git

Cloning into 'brakeman'...
remote: Enumerating objects: 688, done.
remote: Counting objects: 100% (688/688), done.
remote: Compressing objects: 100% (350/350), done.
remote: Total 70833 (delta 286), reused 553 (delta 193), pack-reused 70145
Receiving objects: 100% (70833/70833), 31.30 MiB | 1.85 MiB/s, done.
Receiving deltas: 100% (33806/33806), done.

$ gem build brakeman.gemspec

Successfully built RubyGem
Name: brakeman
Version: 4.10.0
File: brakeman-4.10.0.gem

$ gem install brakeman-*.gem
```

failed, it looks for ruby version ≥ 2.1

Installation program `rbenv` failed, and so, used rvm - ruby version manager.

```
$ rvm install 2.1.1
```

https://rvm.io/rvm/basics

Successful

```
$ gem install brakeman-*.gem
```

Reran: successful

```
$ cd his-rpc
$ brakeman -o output_file.text -o output_file.csv
```

Final versions of the tools:

Ubuntu 12.04.5 LTS

ruby 2.0.0p247 (2013-06-27 revision 41674) [i686-linux]

Rails 2.3.14

rvm 1.29.10 (latest)

brakeman 4.10.0

# APPENDIX B: BRAKEMAN REPORT

saasbook@saasbook:~/his-rpc$ cat output_file.text

== Brakeman Report ==

Application Path: /home/saasbook/his-rpc

Rails Version: 3.2.8

Brakeman Version: 4.10.0

Scan Date: 2020-11-12 22:46:32 -0400

Duration: 2.337530638 seconds

Checks Run: BasicAuth, BasicAuthTimingAttack, CSRFTokenForgeryCVE, ContentTag, CookieSerialization, CreateWith, CrossSiteScripting, DefaultRoutes, Deserialize, DetailedExceptions, DigestDoS, DynamicFinders, EscapeFunction, Evaluation, Execute, FileAccess, FileDisclosure, FilterSkipping, ForgerySetting, HeaderDoS, I18nXSS, JRubyXML, JSONEncoding, JSONEntityEscape, JSONParsing, LinkTo, LinkToHref, MailTo, MassAssignment, MimeTypeDoS, ModelAttrAccessible, ModelAttributes, ModelSerialize, NestedAttributes, NestedAttributesBypass, NumberToCurrency, PageCachingCVE, PermitAttributes, QuoteTableName, Redirect, RegexDoS, Render, RenderDoS, RenderInline, ResponseSplitting, RouteDoS, SQL, SQLCVEs, SSLVerify, SafeBufferManipulation, SanitizeMethods, SelectTag, SelectVulnerability, Send, SendFile, SessionManipulation, SessionSettings, SimpleFormat, SingleQuotes, SkipBeforeFilter, SprocketsPathTraversal, StripTags, SymbolDoSCVE, TemplateInjection, TranslateBug, UnsafeReflection, UnsafeReflectionMethods, ValidationRegex, VerbConfusion, WithoutProtection, XMLDoS, YAMLParsing

== Overview ==

Controllers: 1

Models: 0

Templates: 1

Errors: 0

Security Warnings: 14

== Warning Types ==

Cross-Site Request Forgery: 1

Cross-Site Scripting: 3

Denial of Service: 3

Remote Code Execution: 3

SQL Injection: 3

Session Setting: 1

== Warnings ==

Confidence: High

Category: Remote Code Execution

Check: JSONParsing

Message: json gem 1.7.5 has a remote code execution vulnerability. Upgrade to json gem 1.7.7

File: Gemfile.lock

Line: 49

Confidence: High

Category: Remote Code Execution

Check: YAMLParsing

Message: Rails 3.2.8 has a remote code execution vulnerability. Upgrade to Rails 3.2.11 or disable XML parsing

File: Gemfile.lock

Line: 71

Confidence: High

Category: SQL Injection

Check: SQLCVEs

Message: Rails 3.2.8 contains a SQL injection vulnerability (CVE-2013-6417). Upgrade to Rails 3.2.16

File: Gemfile.lock

Line: 71

Confidence: High

Category: SQL Injection

Check: SQLCVEs

Message: Rails 3.2.8 contains a SQL injection vulnerability (CVE-2012-5664). Upgrade to Rails 3.2.18

File: Gemfile.lock

Line: 71

Confidence: High

Category: SQL Injection

Check: SQLCVEs

Message: Rails 3.2.8 contains a SQL injection vulnerability (CVE-2013-0155). Upgrade to Rails 3.2.11

File: Gemfile.lock

Line: 71

Confidence: High

Category: Session Setting

Check: SessionSettings

Message: Session secret should not be included in version control

File: config/initializers/secret_token.rb

Line: 7

Confidence: Medium

Category: Cross-Site Request Forgery

Check: CSRFTokenForgeryCVE

Message: Rails 3.2.8 has a vulnerability that may allow CSRF token forgery. Upgrade to Rails 5.2.4.3 or patch

File: Gemfile.lock

Line: 71

Confidence: Medium

Category: Cross-Site Scripting

Check: I18nXSS

Message: Rails 3.2.8 has an XSS vulnerability in i18n 0.6.1 (CVE-2013-4491). Upgrade to Rails 4.0.2 or i18n 0.6.6

File: Gemfile.lock

Line: 47

Confidence: Medium

Category: Cross-Site Scripting

Check: ContentTag

Message: Rails 3.2.8 `content_tag` does not escape double quotes in attribute values

(CVE-2016-6316). Upgrade to Rails 3.2.22.4

File: Gemfile.lock

Line: 71

Confidence: Medium

Category: Cross-Site Scripting

Check: NumberToCurrency

Message: Rails 3.2.8 has a vulnerability in number helpers (CVE-2014-0081). Upgrade to Rails 3.2.17

File: Gemfile.lock

Line: 71

Confidence: Medium

Category: Denial of Service

Check: MimeTypeDoS

Message: Rails 3.2.8 is vulnerable to denial of service via mime type caching (CVE-2016-0751). Upgrade to Rails 3.2.22.1

File: Gemfile.lock

Line: 71

Confidence: Medium

Category: Denial of Service

Check: XMLDoS

Message: Rails 3.2.8 is vulnerable to denial of service via XML parsing (CVE-2015-3227). Upgrade to Rails 3.2.22

File: Gemfile.lock

Line: 71

Confidence: Medium

Category: Denial of Service

Check: HeaderDoS

Message: Rails 3.2.8 has a denial-of-service vulnerability (CVE-2013-6414). Upgrade to Rails 3.2.16

File: Gemfile.lock

Line: 71

Confidence: Medium

Category: Remote Code Execution

Check: DefaultRoutes

Message: Rails 3.2.8 with globbing routes is vulnerable to directory traversal and remote code execution. Patch or upgrade to Rails 3.2.18

File: config/routes.rb

# APPENDIX C: CVE AND SCA HARMONIZATION

| No | CVE Harmonization with SCA (Brakeman) | Harmon. Type | # of Vul. Rpt from 2006 to 2017 | Severity Score Mean Values |
|---|---|---|---|---|
| **1** | DOS: | | 9 | |
| | Denial of service | Matching | | 5.84 |
| | Dangerous send (Brakeman: DOS potential) | Correlating | | 5.84 |
| **2** | Code Execution: | | 13 | |
| | Remote code execution | Mapping | | 7.48 |
| | Remote execution in YAML.load | Mapping | | 7.48 |
| | Unsafe deserialization (Brakeman: remote code execution) | Correlating | | 7.48 |
| **3** | Overflow: | | 0 | |
| | Divide by zero | Mapping | | |
| **4** | Memory corruption: | | 0 | |
| | | None | | |
| **5** | SQL injection: | | 13 | |
| | SQL injection | Matching | | 6.82 |
| | Command injection | Mapping | | 6.82 |
| | Dangerous evaluation (similar to command injection) | Correlating | | 6.82 |
| | Dynamic render paths (Brakeman: execute code to modify database) | Correlating | | 6.82 |
| | Format validation (Brakeman: attackers insert code) | Correlating | | 6.82 |
| **6** | XSS: | | 17 | |
| | Cross-site scripting | Matching | | 4.30 |
| | Mail link (Brakeman: Cross-site scripting) | Correlating | | 4.30 |
| | Cross-site scripting (Content tag) | Mapping | | 4.30 |
| | Session settings (Brakeman: stealing cross-site scripting) | Correlating | | 4.30 |
| | Cross-site scripting (JSON) | Mapping | | 4.30 |
| **7** | Directory traversal: | | 5 | |
| | | None | | 4.72 |
| **8** | HTTP response splitting: | | 1 | |
| | | None | | 5.00 |
| **9** | ByPass something: | | 8 | |
| | SSL verification bypass | Mapping | | 5.66 |
| | Attribute restriction (Brakeman: vulnerable to bypass) | Correlating | | 5.66 |
| | Mass assignment (Brakeman: bypass) | Correlating | | 5.66 |
| **10** | Gain information: | | 1 | |
| | Information disclosure | Mapping | | 5.00 |
| | Default routes (Brakeman: Not intended) | Correlating | | 5.00 |
| | Unsafe redirects (Brakeman: redirect away or to a malicious site) | Correlating | | 5.00 |
| **11** | Gain privileges: | | 0 | |
| | Authentication | Mapping | | 0.00 |
| | Session manipulation (Brakeman: gain access) | Correlating | | 0.00 |
| | Basic authentication | Mapping | | 0.00 |
| | Unscoped find (Brakeman: Access any account) | Correlating | | 0.00 |
| | Weak hash (Brakeman: creating weak passwords or signatures) | Correlating | | 0.00 |
| **12** | CSRF: | | 0 | |
| | Cross-site request forgery | Matching | | 0.00 |
| **13** | File inclusion: | | 0 | |
| | | None | | 0.00 |

# APPENDIX D: DATABASE UPLOADING

From MITRE's SyntheticMass

```
$ files2 <- list.files(pattern = "csv$")

[1] "allergies.csv"    "careplans.csv"          "conditions.csv"
[4] "encounters.csv"   "imaging_studies.csv"    "immunizations.csv"
[7] "medications.csv"  "observations.csv"       "organizations.csv"
[10] "patients.csv"    "payer_transitions.csv"  "payers.csv"
[13] "procedures.csv"  "providers.csv"
```

Each file can be uploaded into an analysis tool such as R Studio in this study. For example, for the patient records:

```
$ df <- read.csv("patients.csv")
Total: 1,173 records
```

APPENDIX E: DISTANCE-BASED CHECK FOR SURNAMES

```
> install.packages(stringdist)
> library(stringdist)

> df <- read.csv(file.choose(), header=T)
> dfcensus <- read.csv(file.choose(),header=T)
```

```
> data1 <- expand.grid(dfcensus[1:99],df$LAST)
```

Note: Both *dflnamecen* and *dflname* are character vectors. We need to create a data frame for each for a later calculation. This is a Cartesian product of the two vectors, though, which results in a big size of records. Hence, Census data is limited to 99.

So, now, the size of data1 is:
99 (Census data) x 1,173 (Patient data) = 116,127.

```
jwtest <- function(x,y) {

  return(paste(x,y,format(1 -
round(stringdist(toupper(x),toupper(y),method="jw",p=0.1),2), nsmall
= 2)))

  }

findmaxval <- function(x) {

  colnames(x) <- c('origin')

  data3 <- str_split_fixed(x$origin, " ", 3) # 3 columns
  data4 <- data3[data3[,3]>0.9,]
  data4[!duplicated(data4[,2]),]

  }
```

# APPENDIX F: DISTANCE-BASED CHECK FOR GIVEN NAMES

```
# First Names

> dfssa <- read.csv(file.choose(),header=T)
```

Note: Both *dflnamecen* and *dflname* are character vectors. We need to create a data frame for each for a later calculation. This is a Cartesian product of the two vectors, though, which results in a big size of records. Hence, Census data is limited to 99.

So, now, the size of data1 is:
99 (SSA data) x 1,173 (Patient data) = 116,127.

```
> library(stringdist)
> library(stringr)
> data_a <- expand.grid(dfssa$FNAME, df$FIRST)
> data_b <-
data.frame(jwtest(data_a$Var1[1:116127],data_a$Var2[1:116127]))
```

APPENDIX G: PATTERN-BASED CHECK FOR SOCIAL SECURITY NUMBERS

```
ssntest <- function(x) {
  xv <- length(x)
  xc <- 0

  for (i in 1:xv) {

    if (grepl("^(?!(000))\\d{3}[ -](?!(00))\\d{2}[ -
](?!(0000))\\d{4}",x[i], perl=TRUE)) {
      xc = xc + 1
      print(paste(x[i],"..1"))
    }
    else {
      print(paste(x[i],"..0"))
    }
  }
  print(paste("total found ...",xc))
}
```

APPENDIX H: PATTERN-BASED CHECK FOR DATE OF BIRTHS

```
dobtest <- function(x) {
  xv <- length(x)
  xc <- 0

  for (i in 1:xv) {

    if (grepl("^(0?[1-9]|1[0-2])/(0?[1-9]|[1-2][0-9]|3[0-1])/((19[0-
9]|20[0-1])[0-9])$",x[i], perl=TRUE)) {
      xc = xc + 1
      print(paste(x[i],"..1"))
    }
    else {
      print(paste(x[i],"..0"))
    }
  }
  print(paste("total found ...",xc))
}
```

APPENDIX I: PATTERN-BASED CHECK FOR POSTAL ADDRESSES

```
addrtest <- function(x) {
  xv <- length(x)
  xc <- 0
  addrp <- c("\\d+[ ](?:[A-Za-z0-9.-]+[
]?)+(?:Avenue|Lane|Road|Boulevard|Drive|Street|Ave|Dr|Rd|Blvd|Ln|St)\\
.?")
  cityp <- c("(?:[A-Z][a-z.-]+[ ]?)+")
  statep <-
c("Alabama|Alaska|Arizona|Arkansas|California|Colorado|Connecticut|Del
aware|Florida|Georgia|Hawaii|Idaho|Illinois|Indiana|Iowa|Kansas|Kentuc
ky|Louisiana|Maine|Maryland|Massachusetts|Michigan|Minnesota|Mississip
pi|Missouri|Montana|Nebraska|Nevada|New[ ]Hampshire|New[ ]Jersey|New[
]Mexico|New[ ]York|North[ ]Carolina|North[
]Dakota|Ohio|Oklahoma|Oregon|Pennsylvania|Rhode[ ]Island|South[
]Carolina|South[
]Dakota|Tennessee|Texas|Utah|Vermont|Virginia|Washington|West[
]Virginia|Wisconsin|Wyoming|AL|AK|AS|AZ|AR|CA|CO|CT|DE|DC|FM|FL|GA|GU|
HI|ID|IL|IN|IA|KS|KY|LA|ME|MH|MD|MA|MI|MN|MS|MO|MT|NE|NV|NH|NJ|NM|NY|N
C|ND|MP|OH|OK|OR|PW|PA|PR|RI|SC|SD|TN|TX|UT|VT|VI|VA|WA|WV|WI|WY")

  for (i in 1:xv) {

    if (grepl(addrp,x[i], perl=TRUE)) {
      if (grepl(cityp,x[i], perl=TRUE)) {
        if (grepl(statep,x[i])) {
          xc = xc + 1
          print(paste(x[i],"..1"))
        }
      }
    }
    else {
      #print(paste(x[i],"..0"))
    }
  }
  print(paste("total found ...",xc))
  }
```

APPENDIX J: PARITY-BASED CHECK FOR CREDIT CARD NUMBERS

```
install.packages("checkLuhn")
library(checkLuhn)
```

```
luhntest <- function(x) {
  xv <- length(x)
  xc <- 0
  for (i in 1:xv) {
       if (as.numeric(checkLuhn(x[i])) == 1 ) {
          xc = xc + 1
          print(paste(i,x[i],".. 1"))
          }
       else {
          print(paste(i,x[i],".. 0"))
          }
  }
  print(paste("total found ...",xc))
}

dfccn <- data.frame(df$CCN)
luhntest(dfccn[,1])
```

APPENDIX K: PATTERN-BASED CHECK FOR BANK ACCOUNT NUMBERS

```
ababantest <- function(x) {
  xv <- length(x)
  xc <- 0
for (i in 1:xv) {
  xstr <- strsplit(x[i]," ")
  xstrl1 <- sapply(xstr,"[[",1)
  xstrl2 <- sapply(xstr,"[[",2)
  if (as.numeric(grepl("\\d{9}",xstrl1)) == 1 &&
as.numeric(grepl("\\d{3,17}",xstrl2)) == 1) {
    xc = xc + 1
    print(paste(i,x[i],".. 1"))
  }
  else {
    print(paste(i,x[i],".. 0"))
  }
}
print(paste("total found ...",xc))
}
```

# APPENDIX L: SCAN RESULTS FOR SURNAMES

```
$ data2 <- data.frame(jwtest(data1$var1[1:116127],data1$var2[1:116127]))
$ findmaxval(data2)
      [,1]         [,2]             [,3]
 [1,] "SMITH"      "Smith67"        "0.94"
 [2,] "THOMPSON"   "Thompson596"    "0.95"
 [3,] "MOORE"      "Moore224"       "0.93"
 [4,] "RODRIGUEZ"  "RodrÃ-guez701"  "0.90"
 [5,] "ROBERTS"    "Roberts511"     "0.94"
 [6,] "MILLER"     "Miller503"      "0.93"
 [7,] "TURNER"     "Turner526"      "0.93"
 [8,] "WARD"       "Ward668"        "0.91"
 [9,] "COLLINS"    "Collins926"     "0.94"
[10,] "JOHNSON"    "Johnson679"     "0.94"
[11,] "MITCHELL"   "Mitchell808"    "0.95"
[12,] "KING"       "King743"        "0.91"
[13,] "MURPHY"     "Murphy561"      "0.93"
[14,] "HILL"       "Hilll811"       "0.90"
[15,] "BROWN"      "Brown30"        "0.94"
[16,] "SMITH"      "Smitham825"     "0.90"
[17,] "WHITE"      "White193"       "0.93"
[18,] "WALKER"     "Walker122"      "0.93"
[19,] "ANDERSON"   "Anderson154"    "0.95"
[20,] "ADAMS"      "Adams676"       "0.93"
[21,] "BAILEY"     "Bailey598"      "0.93"
[22,] "DAVIS"      "Davis923"       "0.93"
[23,] "PARKER"     "Parker433"      "0.93"
[24,] "CARTER"     "Carter549"      "0.93"
[25,] "JONES"      "Jones311"       "0.93"
[26,] "HERNANDEZ"  "HernÃ¡ndez971"  "0.90"
[27,] "WILLIAMS"   "Williamson769"  "0.92"
[28,] "HILL"       "Hills818"       "0.90"
[29,] "RODRIGUEZ"  "Rodriguez71"    "0.96"
[30,] "ORTIZ"      "Ortiz186"       "0.93"
[31,] "SMITH"      "de"             "JesÃºs414 0.43"
[32,] "HARRIS"     "Harris789"      "0.93"
[33,] "PRICE"      "Price929"       "0.93"

33 out of 1,173 (strong = > 0.90)
```

APPENDIX M: SCAN RESULTS FOR GIVEN NAMES

```
$ data_a <- expand.grid(dfssa$FNAME, df$FIRST)
$ library(stringdist)
```

```
$ data_b <- data.frame(jwtest(data_a$Var1[1:116127],data_a$Var2[1:116127]))
$ library(stringr)
$ findmaxval(data_b)
      [,1]        [,2]            [,3]
 [1,] "Elizabeth" "Elizabet136"   "0.92"
 [2,] "Angel"     "Angele108"     "0.91"
 [3,] "Angel"     "Angel97"       "0.94"
 [4,] "Jacob"     "Jorge"         "Luis88 0.57"
 [5,] "William"   "Williams176"   "0.93"
 [6,] "Matthew"   "Matthew562"    "0.94"
 [7,] "Gabriel"   "Gabriella773"  "0.92"
 [8,] "Gabriel"   "Gabriel934"    "0.94"
 [9,] "Michael"   "Michel472"     "0.90"
[10,] "Jacob"     "Julio"         "CÃ©sar525 0.54"
[11,] "Daniel"    "Daniel959"     "0.93"
[12,] "Sofia"     "Sofia418"      "0.93"
[13,] "Grace"     "Grayce293"     "0.90"
[14,] "Ryan"      "Ryan260"       "0.91"
[15,] "Christian" "Christiane220" "0.94"
[16,] "Nicholas"  "Nickolas58"    "0.90"
[17,] "Zoe"       "Zoe32"         "0.91"
[18,] "Justin"    "Justin359"     "0.93"
[19,] "Evan"      "Evan94"        "0.93"
[20,] "John"      "John539"       "0.91"
[21,] "Ashley"    "Ashley34"      "0.95"
[22,] "Luke"      "Luke971"       "0.91"
[23,] "James"     "James276"      "0.93"
[24,] "Bella"     "Bella510"      "0.93"
[25,] "Andrew"    "Andrew29"      "0.95"
[26,] "Christian" "Christa452"    "0.90"
[27,] "Alexis"    "Alexis664"     "0.93"
[28,] "Michael"   "Michale231"    "0.91"
[29,] "Jacob"     "MarÃ-a"        "Soledad68 0.51"
[30,] "Joseph"    "Joseph689"     "0.93"
[31,] "Landon"    "Landon622"     "0.93"
[32,] "Michael"   "Micheal721"    "0.91"
[33,] "Riley"     "Riley817"      "0.93"
[34,] "Gabriel"   "Gabriele201"   "0.93"
[35,] "Jacob"     "Miguel"        "Ã\u0081ngel46 0.00"
[36,] "Savannah"  "Savanna736"    "0.92"
[37,] "Brianna"   "Briana139"     "0.90"
[38,] "Anthony"   "Anthony633"    "0.94"
```

```
[39,] "Jacob"     "JosÃ©"         "Luis472 0.57"
[40,] "Lily"      "Lily490"      "0.91"
[41,] "Daniel"    "Daniella68"   "0.92"
[42,] "Jack"      "Jacki842"     "0.90"

   42 out of 1,173
```

# APPENDIX N: SCAN RESULTS FOR SSNS

```
$ ssntest(df$SSN)
[1] "999-18-6398 ..1"
[1] "999-37-6555 ..1"
[1] "999-70-1676 ..1"
[1] "999-40-3975 ..1"
[1] "999-37-3077 ..1"
[1] "999-34-5949 ..1"
[1] "999-76-5105 ..1"
[1] "999-95-5120 ..1"
[1] "999-68-9241 ..1"
[1] "999-70-4325 ..1"
...Omits the rest due to the size of the output
[1] "total found ... 1,173"
```

# APPENDIX O: SCAN RESULTS FOR DOBS

```
$ dobtest(df$BIRTHDATE)
[1] "01/23/2008 ..1"
[1] "06/09/2017 ..1"
[1] "09/17/1909 ..1"
[1] "08/04/1962 ..1"
[1] "06/17/1979 ..1"
[1] "09/14/1965 ..1"
[1] "01/10/1957 ..1"
[1] "09/30/1951 ..1"
[1] "08/08/1957 ..1"
[1] "09/17/1909 ..1"
[1] "10/24/1986 ..1"
[1] "09/30/1951 ..1"
[1] "09/30/1951 ..1"
[1] "05/08/1957 ..1"
[1] "09/17/1909 ..1"
[1] "09/30/1951 ..1"
[1] "09/17/1909 ..1"
[1] "11/10/1921 ..1"
[1] "09/17/1909 ..1"
...Omits the rest due to the size of the output
[1] "total found ... 1,173"
```

# APPENDIX P: SCAN RESULTS FOR POSTAL ADDRESSES

```
$ addrtest(paste(df$ADDRESS,df$CITY,df$STATE))
[1] "943 Smitham Flat Unit 10 Stow Massachusetts ..1"
[1] "654 Bayer Plaza Dracut Massachusetts ..1"
[1] "401 Trantow Lane Stow Massachusetts ..1"
[1] "368 Murazik Ferry Unit 12 Dracut Massachusetts ..1"
[1] "total found ... 4"
```

# APPENDIX Q: SCAN RESULTS FOR CREDIT CARD NUMBERS

```
$ luhntest(dfccn[,1])
[1] "1 4547506746720070 .. 0"
[1] "2 4051071820873030 .. 0"
[1] "3 4592591276143130 .. 0"
[1] "4 4540957202630310 .. 0"
[1] "5 4014177128346410 .. 1"
[1] "6 4014177172448470 .. 0"
[1] "7 4550064326773850 .. 0"
[1] "8 4540957256788700 .. 0"
[1] "9 4065656460781380 .. 0"
[1] "10 4347890536326470 .. 0"
[1] "11 4687844741802850 .. 0"
[1] "12 4171972850040800 .. 0"
[1] "13 4506264733053880 .. 0"
[1] "14 4014177161663280 .. 0"
[1] "15 4020374682317870 .. 0"
[1] "16 4421933558602130 .. 0"
[1] "17 4050143005286450 .. 0"
[1] "18 4347874235135840 .. 0"
[1] "19 4598816762363330 .. 1"
[1] "20 4162068640027220 .. 0"
[1] "21 5574727753845440 .. 0"
[1] "22 5372075632315710 .. 1"
[1] "23 5574840164462440 .. 0"
[1] "24 5372072424355030 .. 0"
[1] "25 5372071011181060 .. 0"
[1] "26 5574841561420480 .. 0"
[1] "27 5372079550801070 .. 0"
[1] "28 5372077747654210 .. 0"
[1] "29 5574841285704540 .. 0"
[1] "30 5574726655306720 .. 0"
[1] "31 5574297022125100 .. 0"
[1] "32 5372072414602860 .. 1"
[1] "33 5372070551400120 .. 0"
[1] "34 5574236613122630 .. 0"
[1] "35 5574842132110880 .. 0"
[1] "36 5574840773288470 .. 0"
[1] "37 5372074472733120 .. 0"
[1] "38 5574297881244130 .. 0"
[1] "39 5574729473167130 .. 0"
[1] "40 5372074067455810 .. 1"
[1] "41 349869475056428 .. 1"
[1] "42 349811712717255 .. 1"
[1] "43 349385830483538 .. 1"
```

```
[1] "44 349907557550320 .. 1"
[1] "45 349690227771431 .. 1"
[1] "46 349380547428504 .. 1"
[1] "47 349977864672836 .. 1"
[1] "48 349965235816331 .. 1"
[1] "49 348085250347111 .. 1"
[1] "50 349714201838402 .. 1"
[1] "51 349621268302703 .. 1"
[1] "52 370024615601855 .. 1"
[1] "53 349977474786604 .. 1"
[1] "54 349958424663085 .. 1"
[1] "55 370024173432743 .. 1"
[1] "56 349977303845720 .. 1"
[1] "57 349343171088234 .. 1"
[1] "58 349811478640022 .. 1"
[1] "59 349610266111042 .. 1"
[1] "60 348059803254165 .. 1"
[1] "61 6011300534721610 .. 0"
[1] "62 6011203277051260 .. 0"
[1] "63 6011306352653200 .. 0"
[1] "64 6011208445766750 .. 0"
[1] "65 6011384462004230 .. 0"
[1] "66 6011202885857270 .. 0"
[1] "67 6011309708328510 .. 0"
[1] "68 6011200250721550 .. 0"
[1] "69 6011293787631810 .. 0"
[1] "70 6011301737286130 .. 0"
[1] "71 6011307354206500 .. 0"
[1] "72 6011004807146650 .. 0"
[1] "73 6011387540668840 .. 1"
[1] "74 6011202982166710 .. 0"
[1] "75 6011200284282870 .. 0"
[1] "76 6011294361605150 .. 1"
[1] "77 6011206768807240 .. 0"
[1] "78 6011301723604380 .. 0"
[1] "79 6011381464057060 .. 0"
[1] "80 6011208451731820 .. 0"
[1] "total found ... 27"
```

APPENDIX R: SCAN RESULTS FOR BANK ACCOUNT NUMBERS

```
>ababantest(dfababan)
[1] "1 72403004 856667 .. 0"
[1] "2 56008849 12345678901234 .. 0"
[1] "3 21000021 9876543210 .. 0"
[1] "4 11401533 NA .. 0"
[1] "5 91000019 NA .. 0"
[1] "6 NA NA .. 0"
[1] "7 NA NA .. 0"
[1] "8 NA NA .. 0"
[1] "9 NA NA .. 0"
[1] "10 NA NA .. 0"
[1] "11 NA NA .. 0"
[1] "12 NA NA .. 0"
[1] "13 NA NA .. 0"
[1] "14 NA NA .. 0"
[1] "15 NA NA .. 0"
[1] "16 NA NA .. 0"
[1] "17 NA NA .. 0"
[1] "18 NA NA .. 0"
[1] "19 NA NA .. 0"
[1] "20 NA NA .. 0"
[1] "21 NA NA .. 0"
[1] "22 NA NA .. 0"
[1] "23 NA NA .. 0"
[1] "24 NA NA .. 0"
[1] "25 NA NA .. 0"
[1] "26 NA NA .. 0"
[1] "27 NA NA .. 0"
[1] "28 NA NA .. 0"
[1] "29 NA NA .. 0"
[1] "30 NA NA .. 0"
[1] "31 NA NA .. 0"
[1] "32 NA NA .. 0"
[1] "33 NA NA .. 0"
[1] "34 NA NA .. 0"
[1] "35 NA NA .. 0"
[1] "36 NA NA .. 0"
[1] "37 NA NA .. 0"
[1] "38 NA NA .. 0"
[1] "39 NA NA .. 0"
[1] "40 NA NA .. 0"
[1] "41 NA NA .. 0"
[1] "42 NA NA .. 0"
[1] "43 NA NA .. 0"
```

```
[1] "44 NA NA .. 0"
[1] "45 NA NA .. 0"
[1] "46 NA NA .. 0"
[1] "47 NA NA .. 0"
[1] "48 NA NA .. 0"
[1] "49 NA NA .. 0"
[1] "50 NA NA .. 0"
[1] "51 NA NA .. 0"
[1] "52 NA NA .. 0"
[1] "53 NA NA .. 0"
[1] "54 NA NA .. 0"
[1] "55 NA NA .. 0"
[1] "56 NA NA .. 0"
[1] "57 NA NA .. 0"
[1] "58 NA NA .. 0"
[1] "59 NA NA .. 0"
[1] "60 NA NA .. 0"
[1] "61 NA NA .. 0"
[1] "62 NA NA .. 0"
[1] "63 NA NA .. 0"
[1] "64 NA NA .. 0"
[1] "65 NA NA .. 0"
[1] "66 NA NA .. 0"
[1] "67 NA NA .. 0"
[1] "68 NA NA .. 0"
[1] "69 NA NA .. 0"
[1] "70 NA NA .. 0"
[1] "71 NA NA .. 0"
[1] "72 NA NA .. 0"
[1] "73 NA NA .. 0"
[1] "74 NA NA .. 0"
[1] "75 NA NA .. 0"
[1] "76 NA NA .. 0"
[1] "77 NA NA .. 0"
[1] "78 NA NA .. 0"
[1] "79 NA NA .. 0"
[1] "80 NA NA .. 0"
[1] "total found ... 0"
```