Improving the Usage of Genetic-Algorithm Signature-Based Detection of Web Threats: A Targeted Approach

Thesis Proposal
By: Tyler Wilding
Tuesday November 1th, 2016
Supervisor: Dr. Salimur Chowdhury
Algoma University
Department of Mathematics and Computer Science

Introduction

As the internet more rapidly grows, so to it does the number of attacks we see on a daily basis; not to mention that as more and more different kinds of devices are connected to the internet more and more vulnerabilities are able to be exploited. As a result of this rapid growth problem, traditional prevention methods of even common web threats like XSS (Cross-Site-Scripting) and SQLi (SQL-Injection) can fall behind and leave systems vulnerable. In addition to various programming measures, tools are also used to maintain and scan logs on the web-server level to identify attackers and these tools typically employ a signature-based mechanism. These signatures are effective at finding the particular web threats that they were designed to detect but they are static; this is a problem as malicious users are constantly adapting their tactics but it is also a problem due to potential false-positive identification of genuine requests.

The proposed solution to this dilemma is to use a non-static signature generation approach using genetic algorithms.[2] By using this approach we are not only generating more signatures to combat the increasing amount of attacks but we are also generating signatures that are more accurate in their detection. Signature-based detection is used effectively in many areas of security such as malware detection, the signature itself is simply a representation of the original malicious incident. For example for a SQLi the signature may contain information related to whether or not a semi-colon was present in the attack, when a new potential-attack enters the system it can be scanned against this signature. However, static signature-based detection has been shown to fail as virus-creators have adapted their tactics to avoid these static signatures, [10-11] similar approaches will be used to circumvent the static nature of web threat detection as well. An evolutionary algorithm such as genetic algorithms are perfect candidates to solve this problem, they operate by modifying the original inputs and then testing the effectiveness of newly generated signatures, continuing until a specified goal is reached. Web threats come in many different varieties across all TCP communication layers, meaning that it is not practical to create a general approach to detect all web threats and it would be much more effective to optimize for each threat's characteristics.

It is clear from the literature that the static signature based approaches is showing its limitations and becoming a problem; in addition to the lack of effort placed on detecting application level attacks. Perhaps the reasoning for the lack of effort is because these attacks change so rapidly, or many can be prevented on the development side. However, even though many can be prevented through proper development, these attacks like all others are constantly adapting and changing code is often a slow process. It is also clear from the literature that [2], a very recent paper is essentially the first to introduce this concept of applying a genetic algorithm to generate entire new signatures to solve this problem however from reviewing this paper I see many open areas for development as well as providing a more comprehensive testing overview of the technique. To be more specific, the paper really only detected SQLi attacks due to their skewed sample sizes, and the final testing did not break down their detection percentage despite the fact that they claimed to detect 2 other types of attacks, XSS and RFI. In addition, I would hope to apply this technique to a non-application layer attack to see if it holds up in those areas as well. In my opinion it may work for some, but not all as this type of attack detection does not take into account previous or future statistics, it only is considered with mapping attacks to signatures.

This thesis will revolve around the use of specifically tailored genetic algorithms in order to generate a greater quantity and quality of detection signatures. As a result, it is necessary to understand the various characteristics of each web threat to craft the best signature representation model for each. Additionally, through and proper testing must be undertaken in order to determine how much of an approvement the signatures create. In order to accomplish this, I propose the use of the Python programming language to create a genetic algorithm that will generate and later test the effectiveness of these aforementioned signatures.

Aim & Objectives

It is my aim through this research to develop a simple tool that can scan web log data to produce a set of signatures to be used in the identification of more web threats and provide comprehensive testing of the technique. I believe that by identifying the strongest detection metrics for each individual web threat, rather than a general approach for all, that much greater and accurate detection as well as testing can be delivered. I plan to accomplish this initiative through the following objectives:

- Review literature concerning:
  - Genetic Algorithm Creation and Optimization
  - Signature-based Detection of Web Threats
  - Specific Web Threat Information
- Develop Genetic Algorithm Tool using the Python library DEAP
  - Begin simple by focusing on just one type of attack with a simple signature
  - Identify and expand signature metrics
  - Expand to other web threats after testing.
- Develop scripts using python to scrape web requests from web logs.
- Develop or use scripts to generate large amounts of test data
- Develop scripts using python to test the individual accuracy of detection for each web threat with new, unseen data compared to other conventional signature detection approach.
  - Plot results using R

Research Questions

- How accurate is this approach at detecting each individual type of web threat?
- What are the metrics that should be used to detect the various web threats?
- Is this approach as effective at detecting web threats on other layers such as a DDoS (Denial-of-Service) attack, as it is at detecting application level attacks such as SQLi. **time permitting**

Literature Review

Initially, it would make the most sense to get a general idea of what are some of the web threats as well as what are considered to be the more serious ones. [1] Provides us with such, starting off the shortcomings of conventional firewall approaches. They mention how a significant portion of attacks on both the network and application layer bypass the safeguards of a conventional firewall approach. Clearly there is a problem with detecting these threats and as stated in [1], conventional methods are starting to show their limitations and weaknesses in this aspect. [1] also discusses the various web threats and what layers they are applicable to and categorizes them into three spectrums: application attacks, distributed denial of service attacks, and network attacks. There exists some overlap between these spectrums as well, for example a socket or RPC attack lies on the edge of being an application attack and a DDoS attack. This is not just a problem in web-threats either, it is also a problem with conventional malware detection which uses signature based approaches, virus creators are starting to make viruses that modify their own code before spreading making each virus unique and a single signature cannot detect it easily or reliably. [10-11]

Getting back to the main topic at hand, [2] begins by discussing the various approaches and problems with signature based detection. Similar to [1], [2] mentions how a slight deviation from the static set of signatures can cause the current methods of detection to fail and they propose using a genetic algorithm to continuously expand and evolve the set of signatures. [2] Also discusses how there is very little effort to develop signature-based intrusion detections for web-application level threats, technologies such as Snort or Bro are more specialized for network intrusions instead. Of the detection systems that do detect application level attacks rely on regular expressions for signatures. [2] shows that there were many signature based related however, this is the first to apply a mutation focused genetic algorithm approach. This is a similar approach to [3], where they generated variations on network-layer exploits to attempt to have better chances of detecting variations on the attack, however their intentions are more so for doing black-box testing of a signature based detection tool, [4] is another paper with a similar reasoning. [5] Goes indepth about the detection and prevention of XSS attacks, focused on checking to see if code is vulnerable. While this is not directly related to detecting the event of the attack like this thesis will focus on, it provides some good background on the metrics that can be used to construct an XSS signature.

Another aspect of this thesis that must be considered in the literature is the overall design and implementation of a genetic algorithm. I have found several papers [6-8] which each discuss various aspects that can be exploited for performance or optimizing the genetic algorithm. Likewise, these papers would be valuable tools for explaining the concept of what a genetic algorithm is and how they operate. Genetic algorithm are a search-based software engineering technique as well, and a very highly referenced and influential paper, [9] will also help in this aspect.

To summarize, it is clear from the literature that the static signature based approaches is showing its limitations and becoming a problem; in addition to the lack of effort placed on detecting application level attacks. Perhaps the reasoning for the lack of effort is because these attacks change so rapidly, or many can be prevented on the development side. However, even though many can be prevented through proper development, these attacks like all others are constantly adapting and changing code is often a slow process. It is also clear from the literature that [2], a very recent paper is essentially the first to introduce this concept of applying a genetic algorithm to generate entire new signatures to solve this problem however from reviewing this paper I see many open areas for development as well as providing a more comprehensive testing overview of the technique. To be more specific, the paper really only detected SQLi attacks due to their skewed sample sizes, and the final testing did not break down their detection percentage despite the fact that they claimed to detect 2 other types of attacks, XSS and RFI. In addition, I would hope to apply this technique to a non-application layer attack to see if it holds up in those areas as well. In my opinion it may work for some, but not all as this type of attack detection does not take into account previous or future statistics, it only is considered with mapping attacks to signatures.

Rationale

One of the reasons why I chose this topic is because out of all of the things that I enjoy developing, automating tasks is my most favourite. Conventional signature based approaches typically require manual identification if the original source is malicious, then a signature can be made from it; or manually creating some kind of pattern matching string. With this approach, this step is removed and the algorithm creates new signatures that detect new attacks into the future and that interests me a lot. Another reason is that security is just in general is an interesting topic, things are constantly changing and improvements feel like they could make a substantial difference immediately.   Additionally, I saw flaws with this new study that is very interesting but are too large for me to ignore and I think that I can present the concept much better. Lastly, I feel like this topic and what I have planned is very reasonable to fit within the short time-frame of the thesis and that I can achieve my goals.

I believe that my thesis, if the results are positive like I expect them to be, will be a better proof of concept for this approach to signature-based detection. Just like how virus security companies are being forced to develop other methods other than just static signature based detection, I believe that web-security companies will as well; and this technique of using genetic algorithms may be part of that future. Additionally, the web is becoming more and more widely used, with the Internet of Things growing and becoming more of a security concern everyday; better protection and detection tools must be put in place to defend it.

Scope

I intend this thesis to cover the majority of the background and rationale for each web threat I venture to detect with this approach. For each web threat I would detail what I discover is the best metrics to detect it as well as the reasons why. This will also include individual test data for the training of each web threat's signature set, which will be used at the end of the work to test against new data and compare performance. This individual test data will also most likely include the different variables that can be manipulated in the genetic algorithm such as the selection percentage and crossover rate. To summarize I intend for the thesis to cover every significant detail and have comprehensive test data to show for each web threat. The one main area that my thesis may fall short is just on the amount of web threats I can detect. I am aiming for at least two different web threats, but ideally I would like to have more especially from different TCP layers to identify the answer to my third research question. Additionally, the results from different layers for this genetic algorithm technique may not be amazing compared to application layer web threats as other techniques may be better suited, however that would be an interesting result and conclusion to include in my thesis either way so I would not consider it shortcoming in my thesis but rather in the approach.

Methods

Materials

For programming languages, I can see myself making use of Python3, R, and potentially some BASH. Python will be used for doing the majority of the work in the project, including all of the parsing of web-server logs, and using a library called DEAP as a framework to make a fast genetic algorithm. Additionally, Python may find use in interacting with the website itself to generate test data, otherwise there are many automated scripts for pen-testing that would do the job, one that I've identified for SQLi is called sqlmap. R will be used for making graphs for the testing phases, and BASH may be used to interact with the server to run the tests automatically.

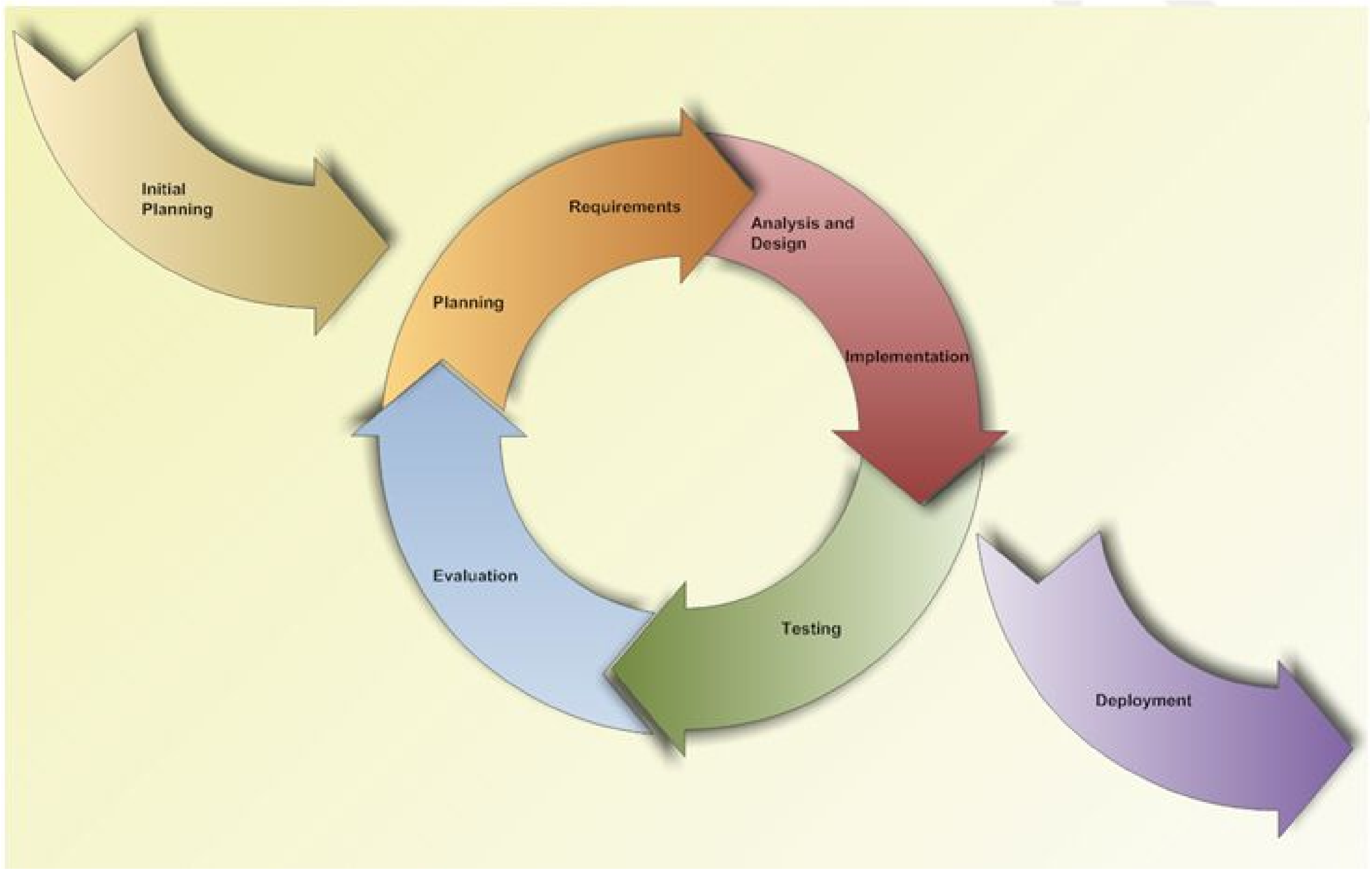
In terms of software, as previously mentioned I will be using various penetration testing tools such as sqlmap to generate test data, however in order to do so I may need to run my own small test website. In which case I will most likely use a popular framework such as Wordpress to facilitate a front-end to send requests to the server. I am debating on the usage of a database for the storage of signatures and test data, if over the course of the work in the thesis I feel as it would make things easier to be inside a database rather than a flat-file, then I will switch to using MySQL or Postgres.

Lastly, the only hardware I may need is the server to run the heavy tests, as well as the web server to generate the test data. In which case, I will opt to choose cloud-based infrastructure such as Amazon AWS or DigitalOcean.

Development Methodology

My plan to develop the needed software for this thesis is essentially described in my objectives but to reiterate. First I need to develop the basic parser from web request format into the bitstrings that the genetic algorithm can take as input. Each web threat would require slightly different parsing rules. Next, I need to develop the actual genetic algorithm that will use a subset of the signatures to train, and test it against the others using a heuristic to evaluate performance. Once this is complete, I need to generate a large amount of test data to train the algorithm better, as well as a large amount of un-seen test data for which the algorithm can be compared to a conventional approach. These basic steps would have be undertaken for each web threat I decide to include in my thesis. For this reason, an iterative approach is clearly going to be the best way to go, I decided to choose the Iterative Development Model. I also liked other models such as Waterfall with Subprojects and Iterative Waterfall, but the problem with these models is that integration is saved until the end of the development phase and I need continuous constant integration and feedback to ensure my tools are working. An iterative development model allows to me break up my project into these small phases but still continuously integrate into the system.

In terms of adding other software engineering practices such as Unit Testing, it would be a good idea for some of the smaller components but in regards to the main genetic algorithm, it is essentially a unit test of itself. That is, the algorithm is constantly checking if the result is a false positive or not, the software is not supposed to be always correct so that could make unit testing rather challenging.



In the iterative development model showed above, we can see how there is the initial planning phase. This phase is largely covered by this proposal, what my end goal is, what I'm going to use to accomplish it, etc. When we get into the main circle, we can see a natural staged approach, I will use the example of detecting SQLi attacks to illustrate how this methodology would work. The first step would be to make the parser to take the data from the web-server logs into a format usable by the genetic algorithm and this is essentially the planning and requirements phases. The analysis and design phases would involve figuring out what information is pertinent from the web-server request for detecting if it is an SQLi or not, as well as trimming away the unneeded lines from the files. I would then implement the program and verify that the output is correct, this would be a good component to use unit testing. After evaluating that everything is ok, we would move onto the next phase detailed above, and a similar process would take place. Additionally, some changes might need to be made to the parser, so we would repeat the process. Eventually we would deploy the entire system and run it through its paces for the final test data.

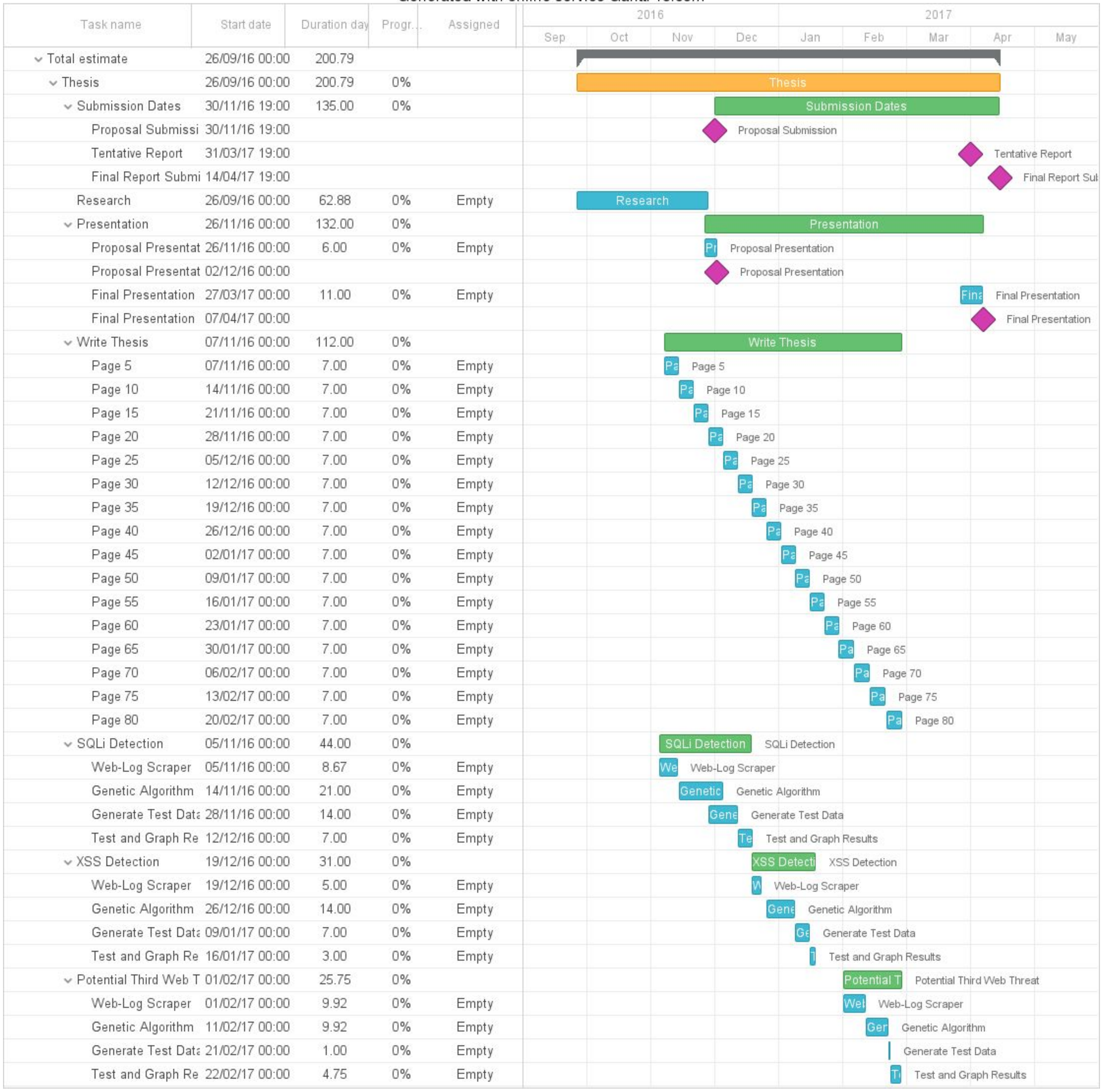
Data will be stored in either a text-file or in a database. Test data will be largely collected through the use of "attacking" a test website and parsing the web-server logs to get the HTTP GET and POST requests. These are simple strings that contain the request that I can parse and trim up to be used in the genetic algorithm. In the real tests there will be attacks and non-attacks to test for false-positive predictions so this information will also have to be stored alongside the attack strings, whether or not it is an attack or not. The final results will be analyzed with either the signatures detecting the attack or not, and this can be easily graphed using R.

Test Plan

Testing Method	**Objectives**	**Modules to be Tested**	**Potential Risks**	**Testing Period**
Unit Testing	Testing of individual modules of the overall system, ensuring output and functionality is correct.	Modules and parsers taking input from the web-logs. Initial genetic algorithm	Genetic algorithm cannot be completely tested as input will be different at the end.	All throughout development
Integration Testing	Making sure that all modules that make up a single web threat work together and then all web threats together.	All the modules	Continuous integration could potentially cause problems as some portions fall behind.	All throughout.
Stress Testing	Seeing how the system performs under load.	Parser and genetic algorithm	Test data is not significant enough.	End.
QA Testing	Seeing if the system meets requirements and is working properly.	All components.	Extra time, but essentially part of the thesis anyway.	Frequently throughout.

Timetable

Generated with online service GanttPro.com



Task name	Start date	Duration day	Progr...	Assigned
Total estimate	26/09/16 00:00	200.79		
Thesis	26/09/16 00:00	200.79	0%	
Submission Dates	30/11/16 19:00	135.00	0%	
Proposal Submissi	30/11/16 19:00			
Tentative Report	31/03/17 19:00			
Final Report Submi	14/04/17 19:00			
Research	26/09/16 00:00	62.88	0%	Empty
Presentation	26/11/16 00:00	132.00	0%	
Proposal Presentat	26/11/16 00:00	6.00	0%	Empty
Proposal Presentat	02/12/16 00:00			
Final Presentation	27/03/17 00:00	11.00	0%	Empty
Final Presentation	07/04/17 00:00			
Write Thesis	07/11/16 00:00	112.00	0%	
Page 5	07/11/16 00:00	7.00	0%	Empty
Page 10	14/11/16 00:00	7.00	0%	Empty
Page 15	21/11/16 00:00	7.00	0%	Empty
Page 20	28/11/16 00:00	7.00	0%	Empty
Page 25	05/12/16 00:00	7.00	0%	Empty
Page 30	12/12/16 00:00	7.00	0%	Empty
Page 35	19/12/16 00:00	7.00	0%	Empty
Page 40	26/12/16 00:00	7.00	0%	Empty
Page 45	02/01/17 00:00	7.00	0%	Empty
Page 50	09/01/17 00:00	7.00	0%	Empty
Page 55	16/01/17 00:00	7.00	0%	Empty
Page 60	23/01/17 00:00	7.00	0%	Empty
Page 65	30/01/17 00:00	7.00	0%	Empty
Page 70	06/02/17 00:00	7.00	0%	Empty
Page 75	13/02/17 00:00	7.00	0%	Empty
Page 80	20/02/17 00:00	7.00	0%	Empty
SQLi Detection	05/11/16 00:00	44.00	0%	
Web-Log Scraper	05/11/16 00:00	8.67	0%	Empty
Genetic Algorithm	14/11/16 00:00	21.00	0%	Empty
Generate Test Data	28/11/16 00:00	14.00	0%	Empty
Test and Graph Re	12/12/16 00:00	7.00	0%	Empty
XSS Detection	19/12/16 00:00	31.00	0%	
Web-Log Scraper	19/12/16 00:00	5.00	0%	Empty
Genetic Algorithm	26/12/16 00:00	14.00	0%	Empty
Generate Test Data	09/01/17 00:00	7.00	0%	Empty
Test and Graph Re	16/01/17 00:00	3.00	0%	Empty
Potential Third Web T	01/02/17 00:00	25.75	0%	
Web-Log Scraper	01/02/17 00:00	9.92	0%	Empty
Genetic Algorithm	11/02/17 00:00	9.92	0%	Empty
Generate Test Data	21/02/17 00:00	1.00	0%	Empty
Test and Graph Re	22/02/17 00:00	4.75	0%	Empty

References

[1] L. MacVittie and D. Holmes, “The New Data Center Firewall Paradigm,” *F5 Networks, Inc., Seattle*, 2012.

[2] R. Bronte, H. Shahriar, and H. M. Haddad, “A Signature-Based Intrusion Detection System for Web Applications based on Genetic Algorithm,” 2016, pp. 32–39.

[3] G. Vigna, W. Robertson, and D. Balzarotti, “Testing network-based intrusion detection signatures using mutant exploits,” 2004, p. 21.

[4] F. Massicotte and Y. Labiche, “On the Verification and Validation of Signature-Based, Network Intrusion Detection Systems,” 2012, pp. 61–70.

[5] M. K. Gupta, M. C. Govil, G. Singh, and P. Sharma, “XSSDM: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications,” 2015, pp. 2010–2015.

[6] N. Stark, G. F. Minetti, and C. Salto, “A new strategy for adapting the mutation probability in genetic algorithms,” in *XVIII Congreso Argentino de Ciencias de la Computación*, 2012.

[7] D. Whitley, T. Starkweather, and C. Bogart, “Genetic algorithms and neural networks: Optimizing connections and connectivity,” *Parallel computing*, vol. 14, no. 3, pp. 347–361, 1990.

[8] J. J. Grefenstette, “Optimization of control parameters for genetic algorithms,” *IEEE Transactions on systems, man, and cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[9] M. Harman, S. A. Mansouri, and Y. Zhang, “Search-based software engineering: Trends, techniques and applications,” *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–61, Nov. 2012.

[10] A. Lakhotia, A. Kapoor, and E. U. Kumar, “Are metamorphic viruses really invincible,” *Virus Bulletin*, vol. 12, p. 57, 2004.

[11] D. Lin and M. Stamp, “Hunting for undetectable metamorphic viruses,” *Journal in Computer Virology*, vol. 7, no. 3, pp. 201–214, Aug. 2011.

Appendices