ISEAGE Traffic Generator

PROJECT PLAN

Team 39
Client: Dr. Doug Jacobsen
Adviser: Dr. Julie Rursch
Matt Vanderwerf: Architect
Dustin Ryan-Roepsch: Quality Assurance
Josh Wallin: Requirements
Ethan Williams: Product Manager
sdmay19-39@iastate.edu
http://sdmay19-39.sd.ece.iastate.edu/

Revised: 09/28/2018 Version 1.0

Table of Contents

List of Figures

Include a **LIST** of all figures used. Be sure images throughout paper have same indexing.

*(example)*Figure 1: Proposed Design Diagram

List of Tables

ex. Table 1: Timeline of proposed work schedules for the Spring semester.

List of Symbols

List of Definitions

Please include any definitions and/or acronyms the readers would like to know.

CDC: Cyber Defense Competition

IDS: Intrusion Detection System

ISEAGE: Internet-Scale Event and Attack Generation Environment

VM: Virtual Machine

1 Introductory Material

1.1 Acknowledgement

We would like to thank Dr. Jacobsen for giving us the problem and helping us design the product so that it fits into the ISEAGE system. We would also like to thank Dr. Julie Rursch for helping us organize the project and determine the scope of each feature. Finally, thank you to ISEAGE who will be handling the integration of the product after it is completed as well as the rerouting of user responses to the product.

1.2 Problem Statement

The traffic generator will solve problems that are encountered at the cyber security competitions and classes at Iowa State. In competitions, the traffic going to the competitors is limited and done manually, making it easy for competitors to detect attacks by the red team. Similarly, in cyber security classes, techniques for guarding against attacks are neve demoed because there is no mechanism in place to simulate an attack. In addition to attacks on the targets, the tool also needs to generate normal-looking traffic to give a consistent, normal load to the target machine.

The traffic generator will be configured by a JSON file which will communicate to the program which targets should receive traffic identified by their address, what kind of traffic the target will receive (i.e. different types of attacks), and additional configurable flags for additional functionality. This file will be used by the system to create a separate task queue for each target. This task queue will be populated by an attack producer which will place the action in the appropriate queue for the target. From there, the consumer for each target will execute actions from the task queue generating the traffic. The tool will also go through a proxy before sending which will reassign the source address so that the attacks are not easily identifiable by the users who simply monitor the source addresses making requests. Return communication will be handled by the ISEAGE system.

1.3 Operating Environment

There are two main environments that the web traffic generator will be used in, the CDC (Cyber Defence Competition) and in cyber security classes at Iowa State. The product is purely software, and will run in a vmware vm in both cases. For the CDC, the software will be a vm inside of ISEAGE, while in the classroom it will either be a vm deployed on the students machine, or also in an ISEAGE environment.

1.4 Intended Users and Intended Uses (two paragraph +)

Our project will have two main intended end user groups: The Cyber Defense Competition (commonly referred to as the CDC), and classes at Iowa State that focus on networking and security.

During the Cyber Defense Competition the Red Team (hackers from industry) must penetrate Blue Team (students participating) systems and secure flags or perform other malicious acts. In order to better obscure the actions of the Red Team, realistic traffic must exist constantly on the network such as would be the case in a normal network. Otherwise, it becomes trivial for the Blue Team to identify the Red Team because most of the traffic can be assumed to be the Red Team. Our ISEAGE Traffic Generator is intended to continuously generate realistic internet traffic that would commonly exist on a production network. This means generating traffic from a wide array of protocols such as: SSH, IMAP, ICMP, HTTP, HTTPS, POP3, IMAP, etc. This will help to simulate a normal production environment network for use during the CDC.

Additionally, classes at Iowa State need realistic traffic within their lab environments for use in class. Not only must good traffic be generated as listed before, so must bad traffic. A good example of this is for use with intrusion detection systems (IDS). Currently there is no great way for classes to test IDSs in Iowa State labs because again, there is no realistic traffic on the network. By generating good traffic and also bad traffic, such as a brute force password fuzzer, there is better data to analyze using the IDS. Without realistic traffic it becomes hard for students to gain a full understanding of how an IDS works because the only traffic they see is the traffic generated from other students setting up their IDS.

1.5 Assumptions and Limitations

Assumptions

- The packets that we create with fake source addresses, will be rerouted back to us (our software does not do anything to make that happen, that needs to be handled by the environment)
- The scale of the traffic needed to generate will be small enough that a VM in an ISEAGE environment can handle it.

Limitations

- The ability to rewrite the source address of the generated traffic, to make the traffic appear like it came from several computers, will only work in an environment like ISEAGE.

1.6 Expected End Product and Other Deliverables

The tool will be developed over the term of the next 2 academic semesters and wil lbe delivered to Doug Jacobsen for use in ISEAGE in the first week of May in 2019. The product will be handed off as a virtual machine encapsulating the docker image for the product. The source code will also be transferred to enable ISEAGE to extend and change the tool should their requirements change in the future.

The second item that will be transferred to ISEAGE in early May is extensive documentation in two parts: Integration/use and design. The integration/use documentation will detail how to integrate the product into existing environments as well as how to run the program to generate traffic for the desired targets correctly. The design documentation will detail the design and implementation of the product's source code. This is intended to give ISEAGE the ability to extend the product should their requirements change in the future.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

To produce a configurable piece of software that can be used during the CDC and cyber security classrooms to obfuscate the red and green team's location on the network, and provide cyber security students with interesting data to play with while learning about IDS' (intrusion detection systems).

2.2 Functional Requirements

List and explain the functional requirements of the project. This would include all the technical requirements you fulfil during your senior design project.

2.3 Constraints Considerations

List and explain the constraints and non-functional requirements of the project. This is where you would enlist non-technical requirements. This may still be a fundamental deliverable that your client needs at the end of the semester.

Discuss the **standard** protocols that you follow in your lab or for writing code. Are these approved by standard organizations like IEEE, ABET etc. Will any of your practices be considered unethical by such organizations? Discuss how standards are applicable to your project.

2.4 Previous Work And Literature

TRex

TRex is Cisco's traffic generator used for benchmarking and stress testing several different parts of a network stack, including DPI, NAT, Firewall, IPS, load balancers, and network caches.

DPDK

The Linux Foundations' Data Plane Development Kit is a set of libraries for fast packet processing / manipulation

2.5 Proposed Design

Discuss possible solutions and design alternatives.

2.6 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

Docker was chosen due to its strength in scalability. Docker containers can be easily spun up and moved around due to the high encapsulation of the software within the container.

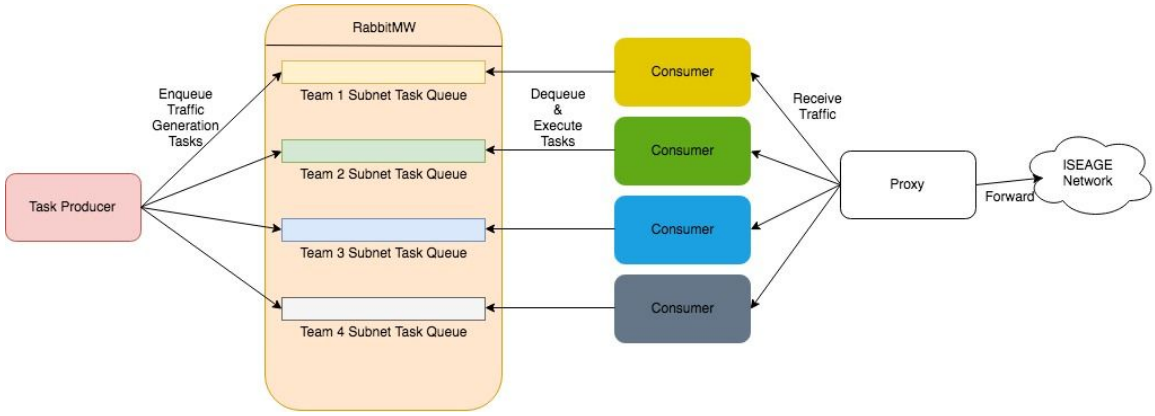
We chose to use a queuing system to enable us to horizontally scale our application. By using a queuing system we can easily change the number of producer or consumer nodes without any effect on the software itself. RabbitMQ was chosen because it's a highly scalable and available queueing system that supports multiple queues. It also is supported by our python library we are using for asynchronous tasks. We wanted the multiple queue support that RabbitMQ provides because it allows us to create a separate queue for each team we are generating traffic against.

2.7 Safety Considerations

Considering this is a software-only tool there are no significant safety considerations that need to be taken.

2.8 Task Approach

Describe any possible methods and/or solutions for approaching the project at hand. You may want to include diagrams such as flowcharts to, block diagrams, or other types to visualize these concepts.



2.9 Possible Risks And Risk Management

We are essentially making a scalable botnet. Most ISP's do "egress filtering" which means that traffic that has a source address that different from the ISP's net is automatically blocked, but if there is a situation in which people are able to bypass this restriction, then they would be able to DDOS a service very effectively, as it would be hard to ban the traffic since you can't just ban a range of IP addresses.

2.10 Project Proposed Milestones and Evaluation Criteria

Limited Support for Simple Protocols

Our first milestone is a very simple prototype, that will test our configuration file format, and be able to support http traffic through WGET, and SSH traffic.

This version of the software will be able to specify a list of "targets" and the protocol / flags to use on that target.

Note that source address rewriting is not considered in this milestone.

Source Address spoofing

The next milestone is support for source address spoofing. This is a large hurdle to get over as it's a very esoteric thing to do, with little existing support materials found online. For example, a lot of tools allow for rewriting http traffic, but we want to rewrite tcp traffic as a whole.

Wide support for various protocols.

After a limited set of protocols is implemented, the next step is just to expand the types of traffic the generator can produce as much as possible before the end of the semester.

2.11 Project Tracking Procedures

Our team has created a Trello board for our project, and are following an agile process. Every task that needs to be completed for the project will have a ticket on trello, and every ticket will be assigned to a team member. Once a ticket is completed, it will be moved to the done section of the board. During our weekly team meetings, we will triage the remaining tickets, and start over again.

2.12 Expected Results and Validation

Upon project completion, our team will have a tool to test at both CDC competitions and within computer engineering classrooms. Sufficient documentation will also be created to ensure the possibility of future work and extensions, should the clients desire it.

Validation will consist of basic testing within both a mock ISEAGE environment, as well as in actual competition settings. By monitoring traffic for a series of use cases, contained within configuration files for the final tool, we will confirm the conformance of our tool's output to its specification. This will then be reviewed by the client to ensure traceability from outcomes to high level requirements and client requests.

2.13 Test Plan

Provide a functional test plan for the present project version

1. Develop a sequence of configuration files increasing in complexity from basic, normal traffic (i.e. non-attack) to full, expected load in a competition/classroom environment (i.e. mixed attack/non-attack traffic)
2. Generate traffic for each configuration file within a realistic testing environment (ISERINK)
3. Monitor generated traffic from both a network (high) level and a user (low) level to ensure that results conform to specifications
4. Capture appropriate statistics for tool output and provide to client for review and feedback

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

• A realistic, well-planned schedule is an essential component of every well-planned project

• Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity

• A detailed schedule is needed as a part of the plan:

– Start with a Gantt chart showing the tasks and associated subtasks versus the proposed project calendar. The Gantt chart shall be referenced and summarized in the text.

– Annotate the Gantt chart with when each project deliverable will be delivered

• Completely compatible with a Agile development cycle if that's your thing

How would you plan for the project to be completed in two semesters. Represent with appropriate charts and tables or other means.

Make sure to include at least a couple paragraphs discussing the timeline and why it is being proposed. Include details that distinguish between design details for present project version and later stages of project.

3.2 Feasibility Assessment

Upon reviewing existing libraries for tool development, as well as consulting multiple times with clients, we believe that this project has been sufficiently narrowed in scope for completion within the given timeframe. Client goals and team expertise have been sufficiently clarified for us to make a clear assessment of our ability to apply our skill sets appropriately and complete the technical work on schedule.

3.3 Personnel Effort Requirements

Task	Description	Developers required	Estimated hours/ developer	Total task time (hours)
Develop configuration file	Determine structure of configuration file so that all the required data can be represented as simply as possible for use by ISEAGE	4	2	8
Establish tools	Create example applications proving that main requirements of the tool can be accomplished with the use of the tools	4	4	16
Build producer for HTTP	Build out the producer to create the request action that will be executed by the correct consumer for the target. This step includes setting up RabbitMQ	2	10	20
Build consumer for HTTP	Build out the consumer to execute any request that comes to it from it's specified RabbitMQ queue without source masking	2	10	20

3.4 Other Resource Requirements

We do not intend to use any other resources.

3.5 Financial Requirements

We do not expect to have any financial costs for this project.

4 Closure Materials

4.1 Conclusion

Before building this product, the CDC as well as cyber security classes at Iowa State lacked realistic traffic to go to competitors or students. This results in easy identification of the attacking team from the team generating normal traffic in the cyber defense competitions. In classes, this results in techniques and tools which are shown to detect or thwart attacks never being demoed or shown working in the real world.

The traffic generator which will be integrated over summer 2019 will make it easy for ISEAGE to set up traffic and attacks going to specified targets through a JSON configuration file. This tool will provide both simple and complex attacks to automatically test the competitors in the CDC and to demonstrate tools and techniques in action for students taking cyber security courses. Additionally, both sets of future users will also get normal traffic to both give noise to the attack requests as well as ensure that their services run under adequately realistic traffic.

4.2 References

4.3 Appendices