ISEAGE 2.0

Design Document

sdmay_06
Doug Jacobson, Julie Rursch
Jacob Conn
Jacob Morrow - Scrum Master
Evan Hellman - Project Manager
Nicholas Krabbenhoft - Systems Engineer
Cameron Isbell - Software Engineer
Jonathan Schnell - Project Owner
https://sdmay22-06.sd.ece.iastate.edu/

Revised: 12/5/2021 version 3

Executive Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

RFC standards were used for the packet subsystems which will handle the IPV4/IPV6 packets.

IETF IPV6 standards

IPV4 standards

Agile Development with KanBan

Summary of Requirements

- ☐ Implement support for IPv6
- ☐ Implement support for Virtual IPs
- ☐ Refactor codebase to improve readability and maintainability
- ☐ Refactor codebase to increase portability for various modern unix systems
- ☐ Update the configuration file format to support the JSON format
- ☐ Delegate layer 2 (ARP/NDP) communications to operating system

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

CPRE 185, 230, 231, 331, 308, 288, 430/530

EE 285

ComS 227, 228, 309, 327, 352

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- ☐ ESXi Management
- ☐ Code Review
- ☐ IPv6 packet structure & networking
- ☐ Cyber Security challenge creation
- ☐ Large-scale C/C++ development

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

Tables

Figures

1 Team

1.1 Team Member

Jacob Morrow

Jon Schnell

Evan Hellman

Cameron Isbell

Nicholas Krabbenhoft

1.2 Required Skill Sets for Your Project

- Networking
- VMware
- C/C++ programming languages
- Software design/architecture
- Computer Engineering
- Cyber Security Engineering

1.3 Skill Sets covered by the Team

Cameron:

Skills: Most of my programming experience is in C/C++. I also know Java, Python, C#, and vhdl. I have some driver programming experience from CPRE 480.

Jon

Skills: networking, network security, linux, python, C, and java programming, ESXI, virtualization.

Nicholas

Skills: Strong skills in networking, C, Linux, and python. I also have experience in virtualization and running small lab environments.

Evan

Skills: Strong background in development especially in Agile environments. Experience programming in C/C++ and a variety of other languages. Well versed in software development and architecture techniques. Experience working with Windows and Linux-based machines as well as their network configurations. Knowledge of computer networking and network security.

Jacob

Skills: C programming, Java programming, very basic python programming, basic Linux programming

1.4 Project Management Style Adopted by the team

The team used an Agile development methodology for project management. More specifically, the team utilized a KanBan-style board to track and assign work.

1.5 Initial Project Management Roles

Jacob Morrow – Scrum Master

Jon Schnell - Project owner

Evan Hellman - Project Manager

Cameron Isbell- Software Engineer

Nicholas Krabbenhoft- Systems Engineer

2 Introduction

2.1 Problem Statement

What problem is your project trying to solve? Use non-technical jargon as much as possible.

We are creating a new ISEAGE to support IPv6 as well as virtual IP's and several different modifications to host a cyber defense competition in the Spring of 2022 for universities in Kosovo.

2.2 Requirements & Constraints

List all requirements for your project . This includes functional requirements (specification), resource requirements, qualitative aesthetics requirements, economic/market requirements, environmental requirements, UI requirements, and any others relevant to your project. When a requirement is also a quantitative constraint, either separate it into a list of constraints, or annotate at the end of requirement as "**(constraint)**". Other requirements can be a single list or can be broken out into multiple lists based on the category.

Implement IPV6 support to ISEAGE

- ISEAGE must be able to handle IPv6 as well as it handles IPv4
- Keep hardware requirements roughly the same so all current systems can be upgraded without requiring new hardware

Design an entry level CDC for students in Kosovo

2.3 Engineering Standards

What Engineering standards are likely to apply to your project? Some standards might be built into your requirements (Use 802.11 ac wifi standard) and many others might fall out of design. For each standard listed, also provide a brief justification.

IETF IPV6 is the primary standard as we have to be emulating a full network stack emulating a network running IPv6.

Standard API's used by ISEAGE.

2.4 Intended Users and Uses

Who benefits from the results of your project? Who cares that it exists? How will they use it? Enumerating as many "use cases" as possible also helps you make sure that your requirements are complete (each use case may give rise to its own set of requirements).

The main users will be researchers and competitors. ISEAGE is used to simulate internet scale attacks and defense computer situations. This means we will have 2 use cases. For students and competitions, this will need to be easy to use and configure. They will also require standard attacks to work well. For researchers, this project will be able to withstand high loads for studying DOS and

DDOS attacks. It also needs to stick to the real life implementations to allow new research to be developed that is applicable to the regular internet.

2 Project Plan

2.1 Project Management/Tracking Procedures

Which of agile, waterfall or waterfall+agile project management style are you adopting? Justify it with respect to the project goals.

Our team will be doing fully-agile development using KanBan. We believe that the agile development process is fitting for our project because we have many different features that can be implemented concurrently. It will give our team the flexibility to handle these features as we need to and also ensure that our project work remains manageable despite load changes with other school work.

What will your group use to track progress throughout the course of this and the next semester. This could include Git, Github, Trello, Slack or any other tools helpful in project management.

We will be using agile/scrum because we feel it is most conducive to our production schedule.

We will be using GitLab, Click-Up and Discord to help with project management.

2.2 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. This step might be useful even if you adopt agile methodology. If you are agile, you can also provide a linear progression of completed requirements aligned with your sprints for the entire project.

Tasks:

1. Compile
   a. Create freeBSD virtual machine(s) as needed to run ISEFlow
      i. Create VM
      ii. Install all dependencies
   b. Compile ISEAGE 1.0 source code and run ISEFlow on the configured VM(s)
   c. Familiarize codebase
      i. Analyze codebase structure and build process
      ii. Analyze codebase deployment process and environment
      iii. Gain insights by running ISEFlow and comparing to codebase actual functionality to the codebase itself
2. Refactor
   a. Add detailed comments to each function while understanding how code works
   b. Add modularity to codebase
3. Add IPv6 Functionality
   a. Add IPv6 functionality for greater flexibility
   b. Move virtual IP from ISEAGE to kernel
   c. Add easy configuration functionality
4. Kosovo CDC

a. Plan Scenario
    i. Plan and document a realistic CDC scenario
    ii. Plan and document the scenario's network
        1. Design devices with realistic OSes, network stacks, protocols, and applications
        2. Implement designed devices (OS Images)
        3. Carefully document all of the changes to each device
    iii. Create scenario document including network diagrams, detailed device info, user info, and any other important details
b. Implement Vulnerabilities
    i. Identify, choose, and document realistic attack types, including IPv6 attacks
    ii. Implement chosen vulnerabilities in various devices
    iii. Create vulnerable application or configuration
c. Test devices and network
    i. Create a test instance of a blue team network
    ii. Test functionality of each machine and their applications
    iii. Perform a penetration test against the network to confirm vulnerabilities are functional

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

What are some key milestones in your proposed project? It may be helpful to develop these milestones for each task and subtask from 2.2. How do you measure progress on a given task? These metrics, preferably quantifiable, should be developed for each task. The milestones should be stated in terms of these metrics: Machine learning algorithm XYZ will classify with 80% accuracy; the pattern recognition logic on FPGA will recognize a pattern every 1 ms (at 1K patterns/sec throughput). ML accuracy target might go up to 90% from 80%.

In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprints).

Milestones:

1. Compile and run ISEAGE 1.0 (ISEFlow)
2. Layer 2 and IPv4 network stack functionality is refactored into the kernel
3. Layer 2 and IPv4 network functionality has 10% lower latency than ISEAGE 1.0
4. IPv6 networking is fully functional in ISEAGE 2.0
5. Virtual IPs are fully functional in ISEAGE 2.0
6. Kosovo CDC has at least 5 vulnerable devices
7. Kosovo CDC has at least 4 vulnerabilities of various types per device

2.4 Project Timeline/Schedule

• 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 (that you developed in 2.2) 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

• Project schedule/Gantt chart can be adapted to Agile or Waterfall development model. For agile, a sprint schedule with specific technical milestones/requirements/targets will work.

Gantt Chart

https://docs.google.com/spreadsheets/d/1FSsm8D_l9mMmGJyryjlCKoV9MdThMxHkp1tOJKV7is8/edit?usp=sharing

Table 1. Gantt chart

2.5 Risks And Risk Management/Mitigation

Consider for each task what risks exist (certain performance targets may not be met; certain tool may not work as expected) and assign an educated guess of probability for that risk. For any risk factor with a probability exceeding 0.5, develop a risk mitigation plan. Can you eliminate that task and add another task or set of tasks that might cost more? Can you buy something off-the-shelf from the market to achieve that functionality? Can you try an alternative tool, technology, algorithm, or board?

Agile projects can associate risks and risk mitigation with each sprint.

Risks:

1. ISEAGE 2.0 will not be functioning for Kosovo CDC ~%66
    a. Use ISAGE 1.0 and reduce boxes we create
2. Layer 2 communications cannot fully be moved to the kernel. ~35% chance
3. ISEAGE hardware unprepared for testing/CDC ~%4

2.6 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in the total number of person-hours required to perform the task.

All numbers in hours

Task:	Cameron	Evan	Jon	Jacob	Nicholas
Compile	2	5	2	2	10
Refactor & understand code	30	30	30	30	30
Move layer 2 to kernel	10	20	0	10	0
Add configuration functionality	10	0	10	7	0
IPv6 Functionality	5	10	5	5	25
Kosovo CDC	12	12	15	12	15

Table 2. Individuals Task Breakdown

Task	Tens of hours to do total
manage client design requirements	1
study the existing source code	15
plan requirements	1
locally compile code	2
Project Definition and Planning	2.3
build an environment on ISEAGE 1.0	7
Implement new features	9

Move layer 2 to kernel	4
implement IPV6	5
Add configuration functionality	2.7
Plan Scenario	1
Plan and document a realistic CDC scenario	.5
Plan and document the scenario’s network	2
Design devices with realistic OSes, network stacks, protocols, and applications	3
Implement designed devices (OS Images)	2
Document all of the changes to each device	1.5
Create scenario document including network diagrams, detailed device info, user info, and any other important details	2
Implement Vulnerabilities	5

Table 3. Task Time Breakdown

2.7 Other Resource Requirements

Identify the other resources aside from financial (such as parts and materials) required to complete the project.

1. Hardware to system test ISEAGE 2.0 on
2. Additional hardware to run ISEAGE 2.0 on for the competition

3 Design

3.1 Design Context

3.1.1 Broader Context

Describe the broader context in which your design problem is situated. What communities are you designing for? What communities are affected by your design? What societal needs does your project address?

List relevant considerations related to your project in each of the following areas:

Public health, safety, and welfare

How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)

Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, increasing/reducing job opportunities

Global, cultural, and social

How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.

Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices

Environmental

What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.

Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials

Economic

What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.

Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization

- Public health/Safety/Welfare - For the universities, this project has to be able to replicate meaningful cyber security scenarios to help educate the students in a manner that is both engaging and challenging.
- Global/Cultural/Social -When designing our scenarios as well as implementing the ISEAGE 2.0 prototype, we have to consider the laws and culture of cyberspace that Kosovo may have.
- Environmental - The biggest environmental impact that this project will have will be the electricity used by the servers running the testbed and the general impact of the creation of the servers used to run the system. To reduce these impacts, we will try to keep the compute requirements of the system low to ensure that the testbed can be run on older computers in order to reduce the manufacturing impacts or allow it to be run on modern computers with relatively lower power consumption costs. Because we are giving the system out to other universities and researchers, we have little control over the computers they run it on or how the electricity is generated.
- Economic - Our project is a free to use research product so there are few economic concerns that we need to deal with. Ideally this project will allow people to learn cyber security for a reduced price at other universities because they will be able to host their own test bed environment rather than pay for some other system.

3.1.2 User Needs

List each of your user groups. For each user group, list a needs statement in the form of:

User group needs (a way to) do something (i.e., a task to accomplish, a practice to implement, a way to be) because some insight or detail about the user group.

There are 3 main groups of users we are designing this for:

CDC Events

- Needs to be able to deploy large number of replicated machines with different IP addresses to each of the competitor teams in order to run the competitions
- Must be able to easily connect external computer to the system to allow red teamers to bring their own hardware to the competitions
- Easy monitoring of the traffic in order to insure rules of the competition are being followed and to allow research on the attacks to be conducted

Classrooms

- In the classroom we need to be able to easily configure and manage the routing configuration of ISEFlow
- In the classroom we need an Advanced way to monitor clients and traffic to view things such as ARP tables.
- A system for spoofing normal internet traffic and attacks such as a distributed denial of service.

Researchers

- Easy to add and remove machines and reconfigure the network architecture in order to allow rapid changes to experiments and research
- In depth logging to provide real world data from the experiments.
- Easily transferable configuration files to allow others to replicate the experiments.

3.1.3 Prior Work/Solutions

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available. Thus, provide a list of pros and cons of your target solution compared to all other related products/systems.

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

ISEAGE 2.0 is the improved version of ISEAGE 1.0. ISEAGE 1.0 is extremely outdated (more than 15 years old) and thus a modernized ISEAGE 2.0 needs to be completed to supersede it and bring the ISEAGE project up to date. Our team was given a baseline ISEAGE 2.0 system that was designed and implemented by a graduate student a few years ago. However, this baseline (or 'original') ISEAGE 2.0 system is very incomplete and poorly documented. This original code was never formally tested and there is no documentation for how to run it. There are many features that are not fully implemented and our team was supplied with a long list of "notes" regarding which features were functional, tested at all, or not implemented.

After diving into the 'original' ISEAGE 2.0 code, we found a number of discrepancies between the architectural diagrams for the code and the actual implementation. We also found it nearly impossible to actually compile and run the code on any FreeBSD system it was designed for, let alone any newer systems. As a result, our team opted to redesign the entire ISEAGE 2.0 system using some approaches utilized in the old code and some entirely new ones that will allow us to make the system meet all of the requirements. These improvements are broad and are detailed further in design section.

3.1.4 Technical Complexity

Provide evidence that your project is of sufficient technical complexity. Use the following metric or argue for one of your own. Justify your statements (e.g., list the components/subsystems and describe the applicable scientific, mathematical, or engineering principles)

The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles –AND–

The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

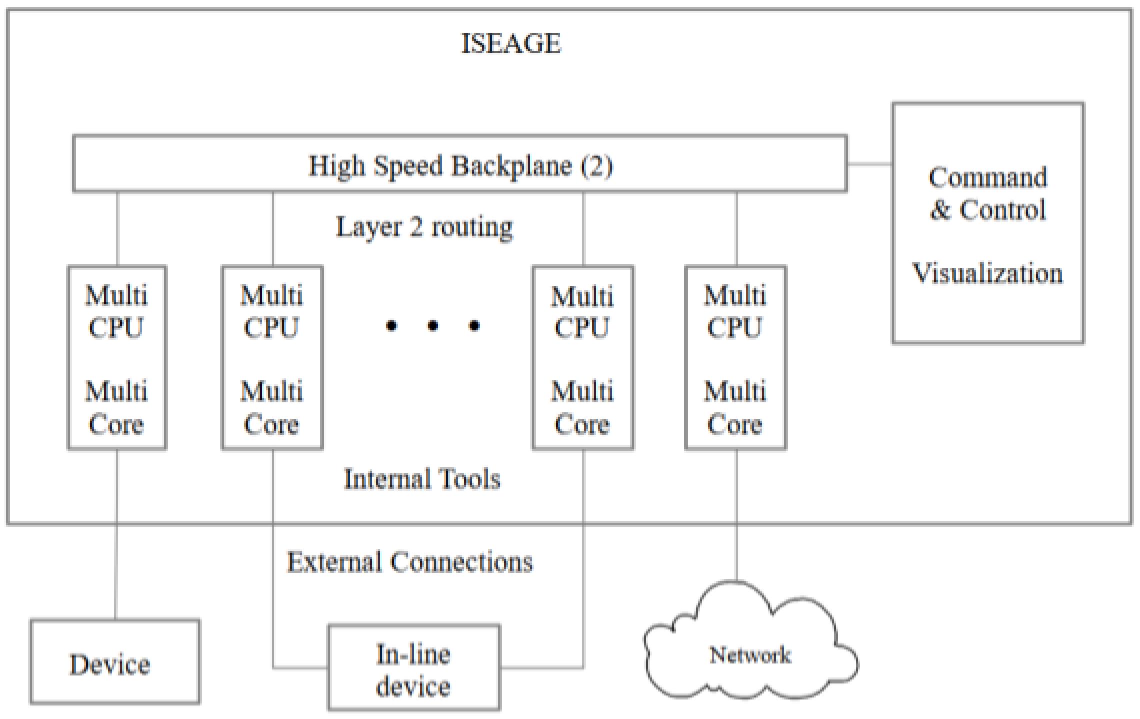

Fig 1. Current ISEAGE layout [2]

This project clearly consists of multiple complex subsystems in both the code and the different machines that interact with each other to run the testbed environment. There are 5 different subsystems in the code involving tools, packets, control, comm, and admin. Each of these subsystems must follow software engineering design principles for construction and modularity of code. In particular the packet subsystems must exactly match the RFC standards for IPv4 & IPv6 packets. We will also need to be able to handle ethernet frame packets according to the RFC standards.

Implementing the new IPv6 and control UX will be massive improvements on the current system and be significantly challenging to implement. The new IPv6 has significantly different packet formats which will need to be implemented from scratch. The current code must also be refactored to run on modern dependencies, operating systems and in accordance with modern standards. This should all be a significant challenge.

3.2 Design Exploration

3.2.1 Design Decisions

List key design decisions (at least three) that you have made or will need to make in relation to your proposed solution. These can include, but are not limited to, materials, subsystems, physical components, sensors/chips/devices, physical layout, features, etc.

Four major design decisions we still need to make are the following:

1. How to properly move the network stack down to the operating system level.
2. How to implement a better way to configure ip tables.
3. How to implement IP spoofing for simulating DDOS attacks
4. How to better monitor the traffic traversing ISEFlow.

3.2.2 Ideation

For one design decision, describe how you ideated or identified potential options (e.g., lotus blossom technique). List at least five options that you considered

To give an idea of our design process here is an example of how we worked through one decision.

To properly implement moving the network stack down to the operating system, we have the following five ideas:

1. We could alter the existing code's TCP dump algorithm.
2. We could move IPv4 & IPv6 down to the operating system level and not implement IPv6.
3. We could implement a new IPv6 implementation of our own into the code.
4. We could find an existing, proprietary solution to move IPv6 down.
5. We could implement both IPv4 and IPv6 into a new solution to be built into the code.

3.2.3 Decision-Making and Trade-Off

Demonstrate the process you used to identify the pros and cons or trade-offs between each of your ideated options. You may wish you include a weighted decision matrix or other relevant tool. Describe the option you chose and why you chose it.

We chose to examine the pros and cons by measuring how close we could get the implementation to our client's standards vs how realistic those implementations are. We've based these off both our skills as programmers and the needs of our clients. Going off of our 5 ideations, implementing both IPv4 and IPv6 into the newly built code would be able to satisfy our clients needs as well as challenging us as a group, since past students failed to do this in past assignments.

3.3 Proposed Design

Discuss what you have done so far – what have you tried/implemented/tested?

As of now, we have a reimplementation of the libdispatch dependency using pthreads to allow the code to run on modern systems. We have also made significant improvements and reductions to the code to remove dead code and improve readability. We have also acquired a set of servers to provide a testing environment for our code changes and are ready to begin the full implementation next semester.

3.3.1 Design Visual and Description

Include a visual depiction of your current design. Different visual types may be relevant to different types of projects. You may include: a block diagram of individual components or subsystems and their interconnections, a circuit diagram, a sketch of physical components and their operation, etc.

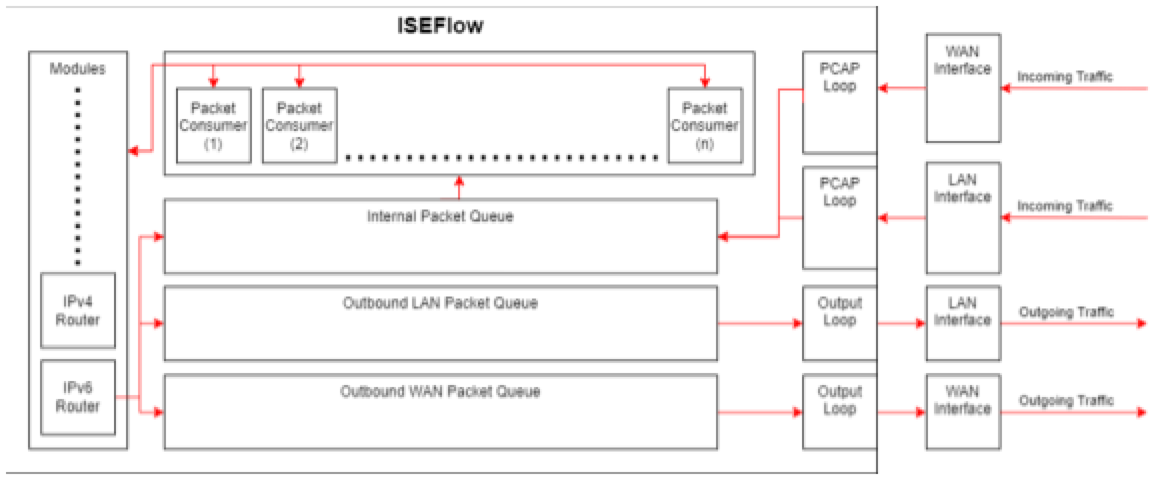

Fig 2. Refactored Implementation

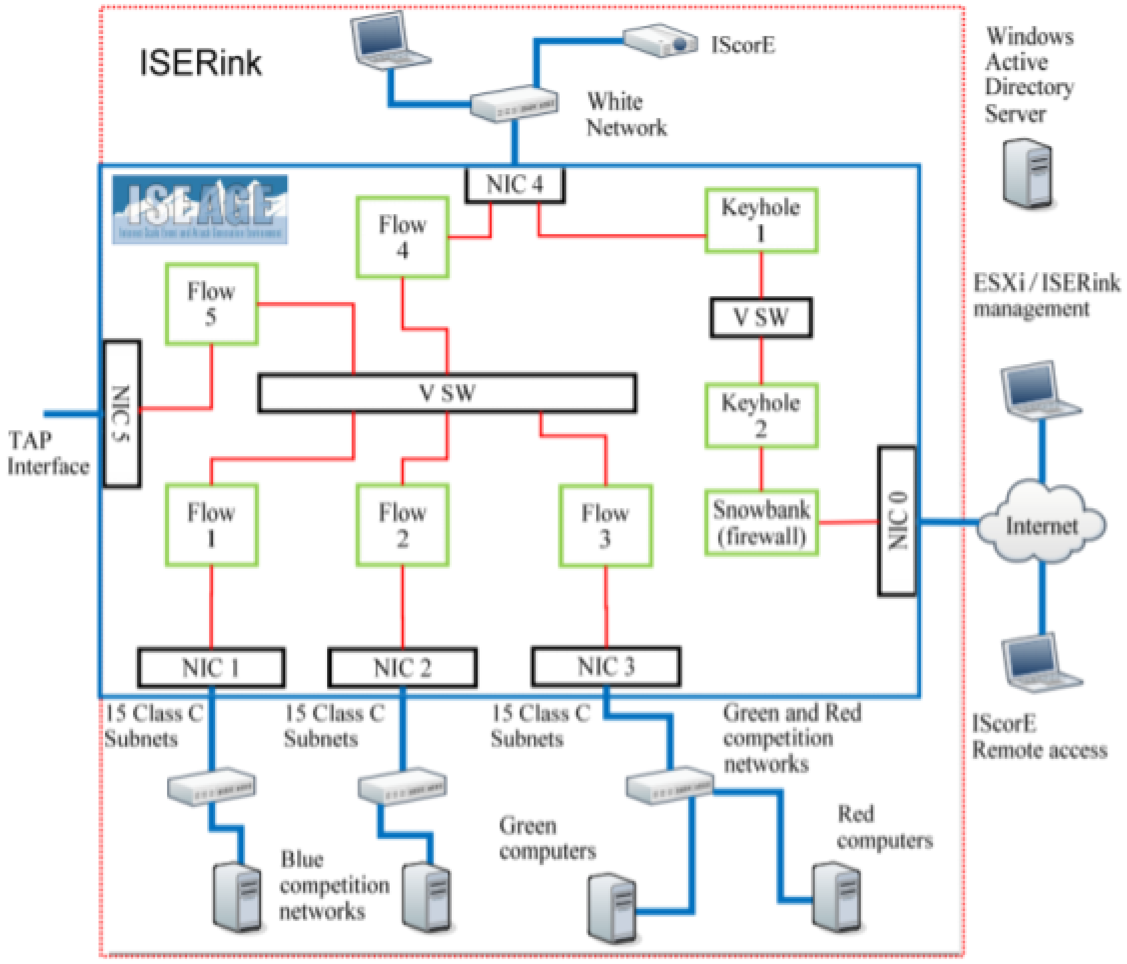

Fig. 3 Detailed ISEAGE overview [3]

Describe your current design, referencing the visual. This design description should be in sufficient detail that another team of engineers can look through it and implement it.

At ISEAGE's core there is a layer 2 virtual switch which connects a number of "flow's" to the network. This Is where theISEflow code Runs. These devices, virtual machines in our case act as routers giving us a layer 3 network to play with. These routers should be highly configurable supporting virtual IP's and acting as default gateways for up to 15 class c networks on the inside interface of the router. (depicted in blue above). This network can connect to the internet through Keyhole 1 and keyhole2 using HTTP, HTTPS, and FTP.

3.3.2 Functionality

Describe how your design is intended to operate in its user and/or real-world context. This description can be supplemented by a visual, such as a timeline, storyboard, or sketch.

How well does the current design satisfy functional and non-functional requirements?

Our implementation is intended to be used in the Kosovo Cyber-Defense competition. It will be interacted with by competitors in the competition and by systems engineers configuring ISEAGE. Currently, ISEAGE 1.0 is functional enough to be used in competitions. Our goal is to bring ISEAGE 1.0 into the future on an updated operating system, refactor some functionality to the operating system, and add support for IPV6.

3.3.3 Areas of Concern and Development

Based on your current design, what are your primary concerns for delivering a product/system that addresses requirements and meets user and client needs?

What are your immediate plans for developing the solution to address those concerns? What questions do you have for clients, TAs, and faculty advisers?

Our primary concern is the IPv6 implementation. IPv6 is a vital component that will be required for ISEAGE 2.0, but we are unsure on how to properly implement it. Our immediate plans are to study IPv6 and learn more about its implementation and to study the code for its implementation of IPv4. For our clients,, we are planning to ask about implementing our own solution for both IPv4 and IPv6.

3.4 Technology Considerations

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

All code we've received and will develop runs on UNIX, specifically FreeBSD. For that reason, we've run into issues with the age of the code and the operating system the code runs on. In particular, we've run into issues with certain libraries and had to completely replace them. On one hand, FreeBSD is a widely supported operating system which works well for the project. On the other hand, we don't know what libraries will be out of date in the future.

Discuss possible solutions and design alternatives

For now, we can use very standard libraries, such as pthreads, to mitigate this issue. A possible design alternative is to use a linux distribution instead. In general, linux is a more modern type of operating system and is likely to be supported for many years to come. However, distributions of UNIX like FreeBSD are popular enough that we don't believe they will be going anywhere anytime soon.

3.5 Design Analysis

– Did your proposed design from 3.3 work? Why or why not?

As of now we have not been able to have the full implementation of our project run since the old codebase still had issues when we got it and took us longer than expected to make the changes necessary for it to be run on modern systems.

– What are your observations, thoughts, and ideas to modify or iterate over the design?

We had originally planned to implement IPv6 in a similar way to how IPv4 was implemented. However as we read through the code we decided that using a series of modules to handle the packets would be the way to go. We feel like this will allow easier additions to the application in future

3.6 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

Please see section 5 for the complete design plan followed by implementation details.

4 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopting test strategy and instruments. In this overarching introduction, given an overview of the testing strategy. Emphasize any unique challenges to testing for your system/design.

4.1 Unit Testing

What units are being tested? How? Tools?

We will be testing several methods and functions over time to ensure they are not only able to compile, but able to function separately and without difficulty. These methods will include but not be limited to; isemaker config files, dataqueue.c, ifconfig.c and ip.c.

What we will be doing to test these code pieces is to isolate them from the rest of the codebase as best we can, reveal dependencies of certain functions, create test cases to test functionality and bug testing.

Tools that could be used to test these code pieces can be NUnit, CuTest, CUnit, AceUnit, and autounit.

4.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

ISEAGE has two major interfaces, the inside interface used to communicate from board to board and an outside interface used to house subnets. Based on our current understanding of the codebase, the code used to handle these interfaces primarily resides in our 'comm' folder. To test these interfaces, we could use both a unit testing application, such as CUnit, to check its functionality. Two examples of such unit tests could be checking that a packet sent from one board is received by another and we could test that passthrough works by sending in a packet and checking that the output is exactly the same as the input.

4.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

Our most critical integration path has to be the queue used to process packets. It is the portion of ISEAGE that handles ISEAGE's primary purpose, to be an internet testbed for cyber security. To test our queue, we will make unit tests that check if all queues are being read from properly, that packets from the queue are being routed where they need to go, and that each queue is receiving the proper information associated with it.

4.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

Our system level components are components that are now being handled by the operating system instead of our code. Some of our network stack will eventually be moved down to the operating system level. When this happens, we will test for functionality by using unit tests where the output from the OS to the rest of the system would be received. We would be checking against our expected values from the current system. Performing these tests would demonstrate our handler's functionality as well as its limitations. In short, we would be testing both the internal and external interfaces as well as the components in-between and the operating system's handling of the network stack.

4.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure do not break? Is it driven by requirements? Tools?

By implementing past test cases for the new additions, we can ensure that the changes made do not affect the functionality of the project. We need to ensure that the IPv6 implementation does not crash or cause breaks within the software so it can at least run properly or at all.

Though there aren't any specific applications to do these tests, there are two strategies we can follow in Agile in order to keep track of progress made. Sprint Level Regression can be done when every new function is added, a new test case can be taken from the test suite and test this functionality. End-to-End testing includes all test cases at once and tests the complete product.

4.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

We will show them the tests we ran for IPv6 that prove it's functional within ISEAGE.

We will most likely involve our clients by running through a virtual scenario that can test our project's requirements fully. The biggest test that will be taking place is the Kosovo CDC. This will be a full test of the entire system to make sure every part of it is fully functional and can stand up to the stress of a competition. Ideally this will also test all of the edge cases and provide any bugs that will need to be fixed in future versions that were not found in our unit testing.

4.7 Security Testing (if applicable)

If we are able to get dedicated servers set up for the testing and running of this code, a fuzzing application should be implemented to find any security issues or general crashes that are in the code that we missed in source code reviews.

4.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

The results of our testing should be a text output from our unit tests that will show what, if any, of our tests failed. This will allow us to isolate bugs and fix them. Once unit testing has revealed no new bugs, we will put the system into our fuzzing framework, assuming we have servers. The fuzzing will reveal the minimum input required to crash the program. This will allow us to find more edge case bugs that we will be able to fix and create more unit tests for. After all of this, the system should ideally not have any more bugs and will be ready for the Kosovo CDC. However, if there are more bugs the CDC should provide a real life use case that should reveal them.

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3. If your project has inseparable activities between design and implementation, you can list them either in the Design section or this section.

Our work on iseage can be split into 2 broad categories. First we are refactoring the existing code base by updating out of date dependencies such as libdispatch, improving code reusability, and removing dead code. This changes very little of the structure of the overarching code base and is happening as we read through and implement the other requirements and is relatively self explanatory therefore we won't go into significant detail about its implementation.

The second and significantly more consequential change we are implementing is the original plan of having a series of modules that each packet would travel through that was not implemented.

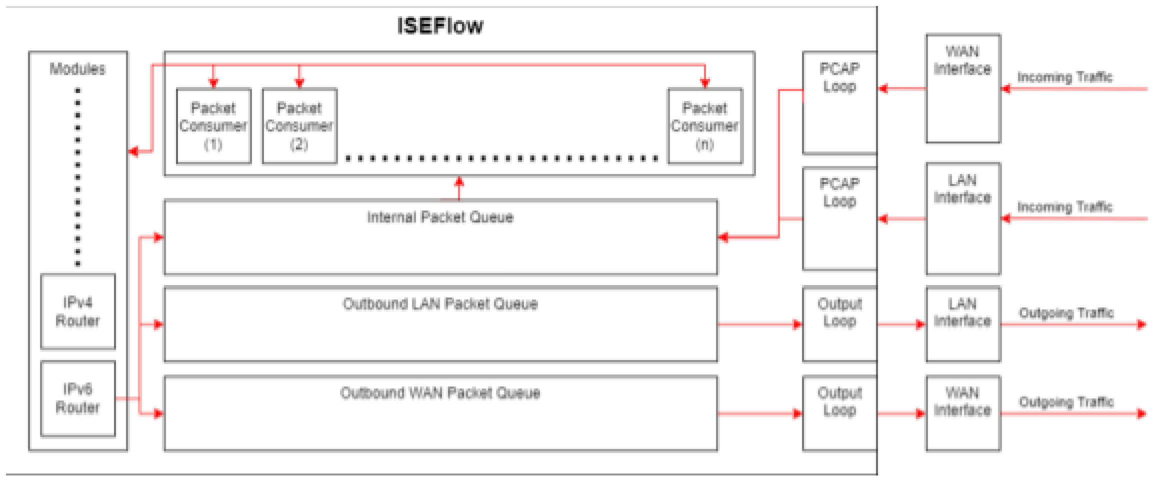

Fig 1. Refactored Implementation

The flow of the packets through ISEFlow is as follows. First they enter through the computers WAN interfaces and are captured by the program. Once captured, they are dropped onto the internal ethernet capture queue. Once they reach the front of the queue, a thread picks up the packet and checks it against any modules in the ethernet model list. If any of the modules matches to the packet, it modifies the packet and continues to be checked by the other modules.

One of 3 things will result from these modules.

1. A module will complete everything needed for the packet, create a response and send that to the outbound queue.
2. A module will see that it needs to be routed to some other queue within the system. For example, if it is an IP packet that needs to be routed it will be sent to the layer 3 queue.
3. No module will match to the packet and it will be dropped.

If the packet is sent to the layer 3 queue, a similar thing will happen where the packet will be filtered through all of the layer 3 modules which will modify the packet appropriately. Which will result in the packet being sent through the layer 3 queue multiple times until it is routed out of the system to the correct destination or the TTL reaches 0 and the packet is dropped

This will allow easy additions of modules which is how we will add the IPv6 support and other requirements.

5.1 Implementation Notes

5.1.1 Thread Pool

To implement a thread pool, we've created two classes: a semaphore class and a thread pool class. Our thread pool is a self contained system that can add jobs, and be permanently shut down. Jobs are run through a "consumer loop". This container loop is used to check when a job is in the queue and perform the top job.

5.1.2 Queue

We plan to replace the current data queue system with a newer version implemented using C++. Currently, there is a data queue with a head and tail attribute. This data queue is used to handle packet throughput from one board to another. We plan a similar implementation as an object with head and tail attributes.

5.1.2 IPV6

We plan to add support for IPV6 into the codebase after it has been refactored because we will have a good understanding of the code at this point and we should be able to add an IPV6 module to handle packets identified by this new version.

6 Professionalism

This discussion is with respect to the paper titled " Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

6.1 Areas of Responsibility

Pick one of IEEE, ACM, or SE code of ethics. Add a column to Table 1 from the paper corresponding to the society-specific code of ethics selected above. State how it addresses each of the areas of seven professional responsibilities in the table. Briefly describe each entry added to the table in your own words. How does the IEEE, ACM, or SE code of ethics differ from the NSPE version for each area?

For this project, we have decided on the ACM code of ethics.

Work Competence: Work should only be performed in areas of competence. Lack of expertise must be disclosed to the employer. Must maintain excellence through upgrading technical knowledge.

Financial Responsibility: Do not engage in deceptive financial practices such as bribery or other malpractices.

Communication Honesty: Be honest of qualifications and expertise, be forthright of circumstances that may lead to conflicts that could undermine success.

Health & Wellbeing: Leadership should consider the physical safety and psychological well-being of all workers.

Property Ownership: Avoid harm of one's property, whether it be physical or data. Professionals should follow best ethical practices with good reason unless told otherwise.

Sustainability: Professionals should promote environmental sustainability both locally and globally.

Social Responsibility: Procedures and attitudes should be towards the quality and welfare of society, reducing harm to the public, serving public interest and fulfilling social responsibilities.

6.2 Project Specific Professional Responsibility Areas

For each of the professional responsibility area in Table 1, discuss whether it applies in your project's professional context. Why yes or why not? How well is your team performing (High, Medium, Low, N/A) in each of the seven areas of professional responsibility, again in the context of your project. Justify.

Work Competence: We have only been performing work that we are confident in our expertise in. On the few occasions we have had doubts about our abilities we have discussed our problems with the clients to ensure they understand where the project is and so that we do not mislead them.

Financial Responsibility: This is a research project that will not be charged for which certainly puts us in the High category for this project.

Communication Honesty: We have maintained open communication and honest communication with our clients about roadblocks and challenges which puts us at a High.

Health & Wellbeing: There are no major or unusual health risks associated with this project. All work is done virtually and in an isolated environment so no harm should come of this project besides the carpal tunnel and other common maladies common to excessive computer work which are outside of our ability to prevent. This means that this section is not applicable for us.

Property Ownership: All of the code has been written by current and former ISU employees and students without a clear statement of what license this is operating under. Because all code written by students is owned by those students, proper legal distribution of this code base may require the permission of every author. Currently this probably puts the project on the Low end and we need to find an appropriate License to operate under.

Sustainability: This project requires relatively few compute resources and can easily be run on older hardware allowing old systems to be reused. However, this does not decrease the impact of other projects so that would put us at a Medium.

Social Responsibility: This project will be a learning tool to help new people and researchers in the field. Allowing more people into the field will have a massive benefit to society as a whole because there is a massive need for more cyber security professionals which clearly places us at a High in this category.

6.3 Most Applicable Professional Responsibility Area

For our project, our team believes the most significant area of professional responsibility is social responsibility. It is paramount that we report only the truth to our clients to both keep them informed and to make sure the future of ISEAGE is well documented. We believe that we've followed this well with regular updates, meetings, and voicing our concerns directly to our clients.

7 Closing Material

7.1 Discussion

Discuss the main results of your project – for a product discuss if the requirements are met, for experiments oriented project – what are the results of the experiment, if you were validating a hypothesis – did it work?

This project will update ISEAGE to run on modern systems and fully implement the module system into the program to allow for easy expansion. This will allow more classes, competitions, and researchers to make use of the system. Ideally this will result in a better understanding of cyber attacks and more cyber security professional.

7.2 Conclusion

Summarize the work you have done so far. Briefly reiterate your goals. Then, reiterate the best plan of action (or solution) to achieving your goals. What constrained you from achieving these goals (if something did)? What could be done differently in a future design/implementation iteration to achieve these goals?

The vast majority of the work so far has been understanding how the old code runs and interacts with itself and determining the best way to implement new functionality. We have been able to update the code as we read through it to remove unused code and improve readability. We have also reimplemented the libdispatch dependency to use pthreads now because libdispatch has been obsolete for more than a year.

We still need to restructure the code to contain modules and then implement the IPv6 and layer 2 modules. This will be created by sending each packet through a list of ethernet modules and layer 3 repeatedly until it is routed out to an external computer or it's TTL goes to zero. Once this is complete we will need to create several CDC boxes to allow us to provide a live fire test for the product.

Our current largest challenges the way packets are captured. All packets still reach the operating system which means our current implementation will not be able to prevent the OS from responding to the layer 2 packets or alter them before they reach the OS.

Our other problem is that the current format of the packets held in the code does not contain any layer 2 information. This will be needed to detect where in our fake routers the packet is. It will also be difficult to add this as the mk_map command compiles the configuration file down to raw binary objects to be used in the program so changing anything may break the configuration setup.

Next semester we will continue to update the code and (re)implement the needed functionality for IPv6.

7.3 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

[1]

"Internet-Scale Event and Attack Generation Environment • Electrical and Computer Engineering • Iowa State University," *Iastate.edu*, 2018. https://www.ece.iastate.edu/research/centers-institutes-and-laboratories/internet-scale-event-and-attack-generation-environment/.

[2]

Iowa State University, *ISEAGE Testbed Overview*. Iowa State University, 2012.

[3]

Iowa State University, *ISEAGE Design, installation, and configuration document*, vol. 1. Iowa State University, 2012.

[4]

A. Shiranzaei and R. Z. Khan, "Internet protocol versions — A review," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 397–401.

[5]

J. May, "Upgrades to ISEFlow: Offloading ARP and supporting IPv6," Pdf, Iowa State University, 2019.

7.4 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.

7.4.1 Team Contract

Team Members:

1) Jacob Morrow________________________________ 2) _Evan Hellman_______________________________

3) _Jonathon Schnell_______________________________ 4) _Cameron Isbell_______________________________

5) _Nicholas Krabbenhoft_______________________________ 6) ________________________________

7) ________________________________ 8) ________________________________

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

Regular team meetings are expected to take place over Discord on Wednesdays at 3:30pm.

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-

mail, phone, app, face-to-face):

Discord will be used for updates between team members and email will be used to communicate with the instructor.

3. Decision-making policy (e.g., consensus, majority vote):

While it depends on the decision, most decisions will likely be made by consensus. If no consensus is possible we will use a majority vote.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be

shared/archived):

Meeting minutes and other records will be stored in discord in their own channel.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

All individuals are expected to attend meetings unless prior notification otherwise or extenuating circumstances arise.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Each assignment should be completed by the due date. All team members should contribute in some way to each assignment. Responsibility for a failure to meet deadlines rests with all team members.

3. Expected level of communication with other team members:

Team members are expected to voice any concerns they have, whether it is about scheduling, the project, etc. over discord or email to other group members. Each member should check the discord daily for updates.

4. Expected level of commitment to team decisions and tasks:

Team members should commit to team consensus to complete project goals. If a task is unclear, team members should contact all other members to clear confusion.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction,

individual component design, testing, etc.):

a. Project owner: Jon
    i. Manages client interaction and design requirement changes, main point of contact for client interactions.
b. Project manager: Evan
    i. Manages team resources a scheduling
    ii. Makes big picture decisions about the direction of project
    iii. Manage a product backlog to ensure a steady and manageable flow of work
c. Scrum master: Jacob
    i. Manages the development process
    ii. Does anything required to ensure the team can work efficiently
d. Lead Software Engineer: Cameron
    i. Main point of contact for codebase
    ii. Manages design requirements and code structure
e. Lead Systems engineer: Nicholas
    i. Manages systems development
    ii. Maintain systems and deploys code or CICD

2. Strategies for supporting and guiding the work of all team members:

To support and guide our team members, we plan to hold regular team meetings, to use the discord to communicate regularly with each other, and to divide work among team members according to each's expertise.

3. Strategies for recognizing the contributions of all team members:

By using discord and git's contributions tab, we will be able to measure each team member's contribution.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

a. Cameron:
    i. Skills: Most of my programming experience is in C/C++. I also know Java, Python, C#, and vhdl. I have some driver programming experience from CPRE 480.
    ii. Expertise: Programming.
    iii. Perspective: As a computer engineering student, I am best suited at looking at the lower level code and developing on that.
b. Jon
    i. Skills: networking, network security, linux, python, C, and java programming, ESXI, virtualization.
    ii. expertise: virtualization and networking
    iii. perspective: Security centered, thinking about security not just as a career but also a hobby. I am a TA for 230/231 so I find educating people about security enjoyable. Run an ESXI homelab in freetime.
c. Nicholas
    i. Skills: Strong skills in networking, C, Linux, and python. I also have experience in virtualization and running small lab environments.
    ii. expertise : Low level embedded code and networking.
    iii. Perspective: I bring a embedded and low level security perspective to the
d. Evan
    i. Skills: Strong background in development especially in Agile environments. Experience programming in C and a variety of other languages. Experience working with Windows and Linux-based machines as well as their network configurations. Knowledge of computer networking and network security.
    ii. Expertise: Development techniques, software architecture, high-level design, software implementation
    iii. Perspective: As a software engineer who has industry experience in development, I have knowledge of how the industry streamlines the design and implementation of software systems. I also have experience managing software development projects and bring a more high-level understanding of the entire process.
e. Jacob
    i. Skills: C programming, Java programming, very basic python programming, basic Linux programming
    ii. Expertise: Basic language programming.

iii. Perspective: I like to poke through old security software and test their limits in some of my off time. I picked security because I felt I'd be able to help more people this way.

2. Strategies for encouraging and support contributions and ideas from all team members:

- Use happy reaction emotes and positive reinforcements in discord when someone posts about a milestone or creative idea.
- Promoting idea sharing and creativity in our regular scrum meetings.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will

a team member inform the team that the team environment is obstructing their

opportunity or ability to contribute?)

- Reach out to fellow team members if they are becoming frustrated or distant from the group.
- Maintain mutual respect for fellow team members and an open floor where everyone can share their ideas and other team members.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

Complete the client requirements by April of 2022.

2. Strategies for planning and assigning individual and team work:

Clickup will be used to create, assign, and manage Scrum stories. This will allow us to accurately assign and track work as it is being done and allow individual developers to focus on development rather than planning and other overhead.

3. Strategies for keeping on task:

Weekly Scrum meetings will ensure members are held accountable for the tasks they agreed to work on for each sprint as well as provide a forum for discussion or modification of stories as conditions change. This will help the team react to change effectively and keep us on task.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

We plan to first talk directly with the team member and to notify the TA.

2. What will your team do if the infractions continue?

The instructor will be notified and removal may have to be implemented.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) _Jacob Morrow____ DATE _12/05/2021_________________

2) ___Nicholas Krabbenhoft____ DATE _12/05/2021_________________

3) __Evan Hellman____ DATE _12/05/2021_________________

4) __Cameron Isbell____ DATE _12/05/2021_________________

5) __Jonathon Schnell____ DATE _12/05/2021_________________

6) ____ DATE __________________

7) ____ DATE __________________

8) ____ DATE __________________