Wireless Security Lab & Open BTS

Design Document

This document outlines our plan for building configuring a remotely accessible lab for experimenting with wireless security and interfacing it with Open BTS.

Senior Design, Team 13-14:

Xiaofei Niu
Thuong Tran
Matt Mallet
Yuqi Wang
Chris Van Oort
Muhammad Tahsinur Rahman Khan

Client/Advisor:

Dr. George Amariucai

Table of Contents

1) Project Overview

The purpose of this project is to provide students enrolled in Computer Engineering 537: Wireless Network Security at Iowa State University with an environment in which they can carry out various experiments involving different wireless communication protocols. Our goal is to build a sandbox style environment capable of supporting four simultaneous users with the tools and hardware needed to carry out a multitude of different experiments, and make this environment accessible from anywhere in the world to support the needs of both on- and off-campus users. The rest of this document deals with the design and configuration of the environment. It is broken down into three main sections: system design, detailed implementation, and testing.

2) High-Level System Design

The main objective of the project is to provide a remotely accessible laboratory environment where students can carry out wireless security experiments. Any wireless security experiment would ideally require at least three interactive nodes: two transceiver nodes communicating information wirelessly with each other and an attacker node which attempts to target the wireless communication.

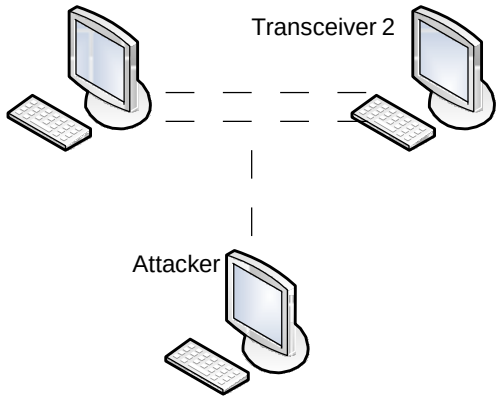


Figure 1: Basic three-node attack scenario

An infrastructure setup may additionally require a coordination center like an access point in a Wi-Fi network or a base station in a GSM network. In this case the two transceivers will communicate with each other through the coordination center or a single transceiver will communicate with the external network through the coordination center.

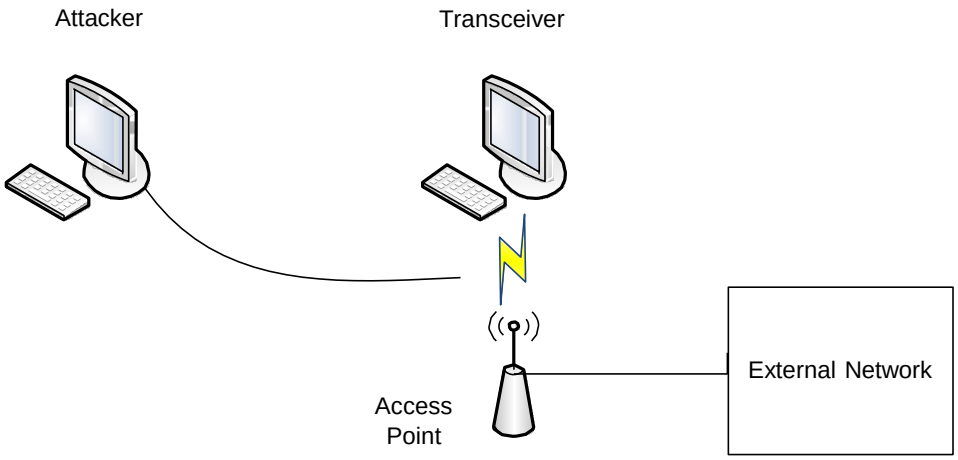


Figure 2: Two-node infrastructure-based attack scenario

The experimenter will ideally require at least minimal remote access to two transceiver nodes or a single transceiver node and an access point to set up a viable experiment and the attacker node to carry out this experiment. Many wireless security experiments require direct access to the wireless interface to bypass certain protocols and carry out an attack. It is also the goal of this project to build an environment where an experimenter can remotely carry out many different types of wireless security experiments without limitation to hardware access. Thus an experimenter will require full access to at least two different machines , each with its own wireless interface and running remote desktop software, and may additionally also require access to a third machine or an access point.

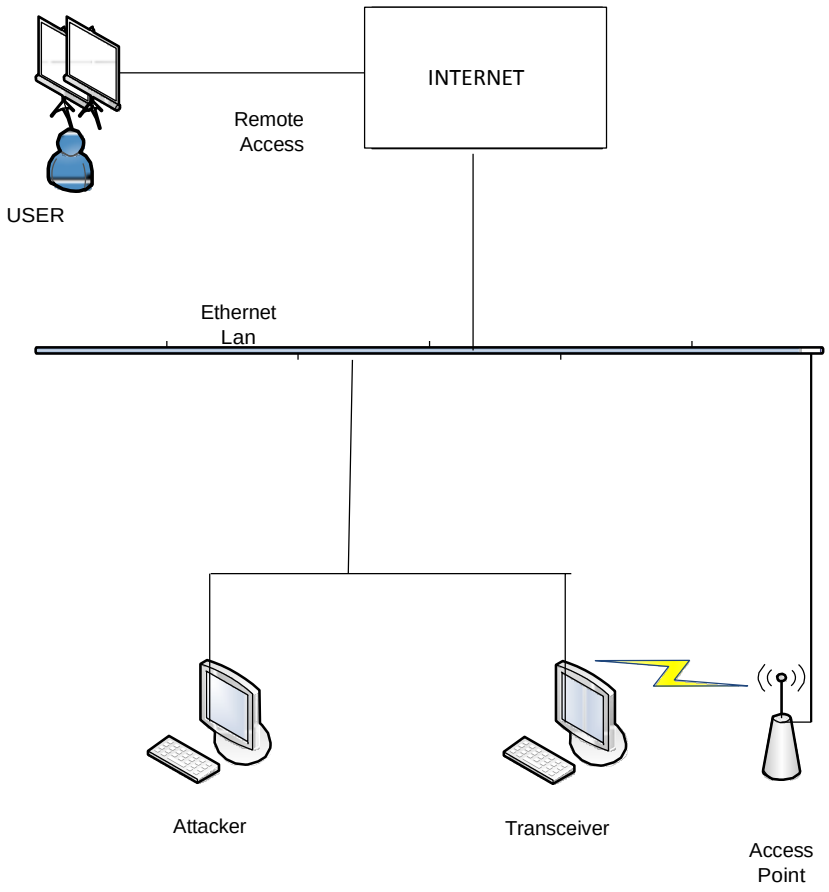


Figure 3: Remote access laboratory single-user overview

Another important part of the project is to provide multiple user access, such that many users can access the lab hardware remotely at the same time. After much debate and analysis, the client and the team have come to the conclusion that we will support up to 'four' users at the same time. As each user will require at least two machines and the lab will support four users at the same time, the lab will need at least eight remotely accessible machines. To setup such a hardware platform we have decided to use virtual machine servers which can support multiple virtual machines at the same time. This will not only help bring down the cost but will also help with setup and administration.

*A detailed description of the VM setup is provided in the Detailed Design section of this document.

Valid users will be able to access the lab environment remotely over the public internet from anywhere. They will first pass the Iowa State network firewall to connect to a local Iowa State VPN server using valid credentials. They will then pass the lab environment network firewall to connect to the lab environment using remote desktop and valid credentials. The lab environment will house the virtual servers and the wireless hardware to carry out wireless security experiments for Wi-Fi, Bluetooth and optionally GSM and RFID.

3) Functional Decomposition

The system as a whole will attempt to provide a laboratory environment for the students such that they can carry out many different types of wireless security experiments related to Wi-Fi, Bluetooth, GSM, RFID and/or ZigBee. This section attempts to divide the whole project into several parts, whereby each part plays a functional role to satisfy the requirements.

3.1) Hardware Architecture:

This part of the project involves the setup and the configuration of the VM servers and the wireless hardware. The VM servers will need to be set up such that each VM can independently support its own Wi-Fi, Bluetooth, and/or other wireless adapters. The wireless hardware will include all necessary devices such as routers, switches, antennas, radios, spectrum analyzers and wireless cameras.

3.2) Software Architecture

This part of the project involves the setup and configuration of the Operating Systems, wireless tools, user accounts, remote desktop software and any other custom software to support wireless attacks and hardware integration. The lab will also include a web interface to provide administrative information and network monitoring for the users.

3.3) Wireless Experiments

This part of the project involves the configuration and implementation of security critical wireless experiments for the following technologies

Wi-Fi	3-4 experiments
Bluetooth	1-2 experiments
GSM	1-2 experiments (optional)
RFID/ZigBee	1-2 experiments (optional)

3.4) Project Documentation

Project documentation will include:

Project Plan: This document will define functional and non-functional requirements for the project and provide a timeline.

Design Document: This document will provide a detailed design for the overall project and for each individual part of the project.

Testing Report: This document will list test results for any preliminary testing and final testing at the end of the project.

Final Report: This document will provide a final analysis of the project after completion and suggest future development or action.

Administrator Documentation: This document will provide a reference guide for the instructor and the teaching assistant to administer and manage the systems.

Student Documentation: This document will provide detailed instructions and tool descriptions for wireless experiments for the students.

4) Design Tradeoffs

Physical Machines vs. Virtual Machines

Installing a large number of physical machines can be very expensive and the machines themselves will have to be configured separately. A virtual machine server can be a very effective solution, especially when a large number of machines need to be set up and accessed remotely. These virtual machines can also be configured and administered easily using a common console. They also provide the added benefit that they can be replaced with an existing image if something goes wrong or the machine gets corrupted.

Single Adapter vs. Multiple Adapters

A single wireless adapter can potentially be used by multiple users connected remotely for only a very few select specific experiments. Many wireless security experiments require direct access to the wireless interface to bypass certain protocols and carry out an attack. Thus, each wireless adapter can be accessible by only one user at one point of time. Also, although the virtual machines can share a single Ethernet adapter, they cannot share a single wireless adapter. Thus each virtual machine will need its own separate wireless adapter.

Single Router vs. Multiple Routers

A single router can be used to set up different wireless networks with different names and different encryption techniques. However each network run by a single router needs to run on the same channel and running multiple networks can also overload the router. Thus a single router can support only one jamming experiment and multiple routers will be needed to support multiple users.

PCI Adapters vs. USB Adapters

PCI adapters have been available in the market for a long time now and have better support and troubleshooting for attack experiments. However, USB adapters are starting to be used more and more extensively nowadays and offer the added advantage that they can be connected to the VM server using extensible USB cords such that they can be spaced apart to limit interaction and interference between the radio waves. Additionally, server machines often do not come with a very large number of PCI slots and USB adapters may be necessary to support multiple adapters for the multiple VM's.

Windows OS vs. Linux OS

Windows OS has arguably better support for remote access and is the more widely used operating system. However, Linux OS offers better flexibility, has better support for wireless tools, can be easier to configure and is an open source solution. Windows users can take some time to get used to the Linux environment; however, Linux offers much better support for wireless tools and can therefore be considered the better option for carrying out wireless security experiments.

5) Low-Level System Architecture

The full-system architecture consists of three distinct but related sub-architectures: hardware, software, and network. The hardware architecture consists of all physical components necessary to properly implement the requirements of the project (e.g. computing equipment, radios, network interconnect hardware, and any custom-built hardware deemed necessary). The software architecture consists of all non-physical components of the project (e.g. virtualization environment, operating systems, exploit tools, administration services, user scripts, and device firmware for any modified or custom device in use). The network architecture contains some components of both hardware and software architecture, but abstracts these components to deliver a node-centric view of inter-device communication paths and protocols.

5.1) Hardware Architecture

The hardware architecture was designed with the intent of employing commodity hardware in creative ways in order to fulfill the unique needs of the project in a cost-effective way. Attempts were made to consolidate hardware wherever possible to trim costs and ease administration. The end result is robust yet easy to use and expandable should future need arise.

The backbone of the project consists of two commodities x86-based servers. These will serve as the sole compute nodes for all project-related services and users. The servers will be sufficiently powerful such that no student's use of the environment will noticeably impact its usability for any other student, nor will the necessary backend services disrupt or be disrupted by student usage. Due to materials on hand, a single server based on Intel's Xeon 5500 series processor technology and architecture was chosen as the initial test bed for this project. Xeon 5500 is a maturing series (which leads to lower costs), but still utilizes Intel's current-generation Nehalem core for an excellent power to cost ratio. Paired with triple- channel DDR-1333 and a mainboard supporting advanced Intel Virtualization Technologies, this is a very capable architecture for a multi-user, multi-service environment. The full technical specifications for the test server are listed below; however, it should be noted that the final implementation will utilize two machines of possibly lower performance, in order to spread the I/O load.

5.1.1) Base Testbed Specifications:

Processor:	Intel Xeon 5504, 4x2.0GHz, two CPUs
Mainboard:	Supermicro X8DTL-3F X58-chipset board, with IPMI
Memory:	12GB ECC DDR3-1333, 6x2GB
Storage:	Western Digital RE3, 500GB, two drives, JBOD

In addition to the core systems, peripherals are necessary in order to communicate with the wireless networks under test. The Xeon 5500 series architecture provides support for Intel Virtualization Technologies for Directed I/O (VT-d), which allows virtual machines to have direct access to the PCI bus. This allows for the ability to access PCI- or PCI Express-connected Wi-Fi cards as virtualized hardware, which would not be possible on a platform which does not support VT-d. Using Atheros PCI-based interfaces enables the utilization of the very well-supported open-source driver package ath5k, which by

virtue of being open may be able to be modified to generate novel attacks. Ultimately, however, the team has chosen to utilize USB-based dongles from Ralink, which has a less-supported but similarly open-source driver, and eliminates the need for a platform supporting VT-d or a motherboard with more than four PCI or PCI Express slots. Banking on the use of largely independent host controllers (motherboard controllers supplemented by PCI Express-based add-on card controllers), the USB devices should experience no more noticeable latency than the PCI solution, and provide a comparably priced, more convenient alternative.

In addition to base computing resources, several other hardware devices are necessary to complete the lab environment. A number of wireless access points are required in order for the devices to speak to one another and for many of the attacks to be properly carried out. Some method of analyzing the signals present in the laboratory – whether spectrum or waveform – was requested by the client as a means of better understanding the attacks taking place. Finally, hardware for any protocol being studied other than Wi-Fi – such as Bluetooth – would also need to be made available.

The choice of hardware type and interface, and equipment multiplicity, has been carefully weighed between price, performance, user experience, and implementation practicality. The following summarizes the pros and cons of several implementation scenarios for Wi-Fi hardware; similar tradeoffs exist with other interface options (such as Bluetooth).

5.2) Wi-Fi Hardware Scenarios:

Scenario 1: 1/1/1 Bridged

Transmitters:	1 Wi-Fi
Access Points:	1
Attack Nodes:	1 Wi-Fi (shared via Ethernet bridge)
Simultaneous users:	Many
Implementation:	One transmitter communicates over one access point. Attack node is accessible to users over a virtual Ethernet connection. Users send packets to virtual attack node, which relays them to the radio.
Pros:	Requires only two wireless interfaces and an access point, so cost is low. No worry about multi-channel interference. Assuming a sufficiently powerful host, would support every user simultaneously. Few privileges needed on the machine for use.
Cons:	Many attacks require driver-level access to the wireless interface, which is not accessible over a virtual Ethernet bridge, therefore this method severely limits the types of attacks which are possible. Users could write attacks which require unfair amounts of radio time, take down the network, or otherwise negatively impact other users' experiences.

Scenario 2: 1/1/1 Shared

Transmitters:	1 Wi-Fi
Access Points:	1
Attack Nodes:	1 Wi-Fi (shared)
Simultaneous users:	1 or Many
Implementation:	One transmitter communicates over one access point. Users simultaneously log on to attack node directly and generate attacks on local hardware.
Pros:	Requires only two wireless interfaces and an access point, so cost is low. No worry about multi-channel interference. Assuming a sufficiently powerful host, would support every user simultaneously. Local hardware allows driver-level access, so most attacks are feasible.
Cons:	Limited support for driver-level access by multiple simultaneous users; device locking by OS may allow only one user access to hardware at a time. If multiple users are allowed, users could write attacks that would require unfair amounts of radio time, take down the network, or otherwise negatively impact other users' experiences.

Scenario 3: 1/1/4 Independent

Transmitters:	1 Wi-Fi
Access Points:	1
Attack Nodes:	4 Wi-Fi
Simultaneous users:	4
Implementation:	One transmitter communicates over one access point. Four machines with radios are available for student use, one student per machine, first come first served.
Pros:	Requires five wireless interfaces and an access point; cost is fairly low. No worry about multi-channel interference. Potentially lowers the requirement for host power due to lower user count. Local hardware on single-user system

	guarantees unrestricted access to radio.
Cons:	Limited to four simultaneous users; others must wait for slots to open. A single target network means users can negatively impact other users' environment experience, which may be manifest in unpredictable results or unreliable environment operation.

Scenario 4: 4/4/4 Independent

Transmitters:	4 Wi-Fi
Access Points:	4
Attack Nodes:	4 Wi-Fi
Simultaneous users:	4
Implementation:	Four transmitters communicate over four access points. Four machines with radios are available for student use, one student per machine, first come first served.
Pros:	Each user may run any attack on the environment, including denial-of-service or jamming attacks, with little risk of disrupting other users. Local hardware on single-user system guarantees unrestricted access to radio.
Cons:	Limited to four simultaneous users; others must wait for slots to open. This volume of USB devices requires a hub or controller card. Cost is comparatively high. Interference between four independent networks may be a problem.

Given these choices, and the relatively low cost of the variable hardware being considered, it was ultimately decided to pursue a plan of independent networks with four transmitters, four access points and four malicious radios. This choice guarantees the greatest number of possible attacks and the best reliability for all users.

In terms of physical hardware used, the radios will be chosen according to greatest chipset compatibility with attack generation; testing is currently ongoing to determine the relative performance of several retail cards in this regard. As previously mentioned, Ralink chipset devices are being most closely inspected, but the precise family has yet to be decided.

For access points, the team will be utilizing consumer-grade routers. As these will not be used for file transfer or other massive-throughput applications, they will be chosen based on low cost and compatibility with the DD-WRT custom firmware package. Currently, four D-Link DIR-601 routers form the access point array for the environment.

In order to support attacks on Bluetooth equipment, five economy Bluetooth to USB adapters will be added to the machine, in the configuration of two transmitters, one receiver, and two attack adapters. One or more USB hubs or PCI/e to USB controller cards will be necessary to accommodate these and the Wi-Fi adapters.

To add depth to the project, some other unique hardware will be made available. A Wi-Fi camera will be added to the lab to allow students the ability to attempt attacks against a practical target. In addition, a USB-based spectrum analyzer tool will be placed in the lab and will be accessible to students to allow the visualization of the radio environment. This will either be an off-the-shelf unit, or a tool developed on top of the TI eZ430-RF2500 development platform.

It is also desirable to be able to see waveforms of some of the radio activity to analyze network modulation schemes. An attempt will be made to construct a hardware interface to capture network traffic, convert to baseband, and display the result to the user, although the details of the implementation of such a device are in a very early stage of development.

Should the client decide that another protocol is also to be supported (e.g. RFID, GSM, etc), these implementations will require hardware as well, which likely will consist simply of additional USB devices.

A full-system hardware diagram is illustrated below.

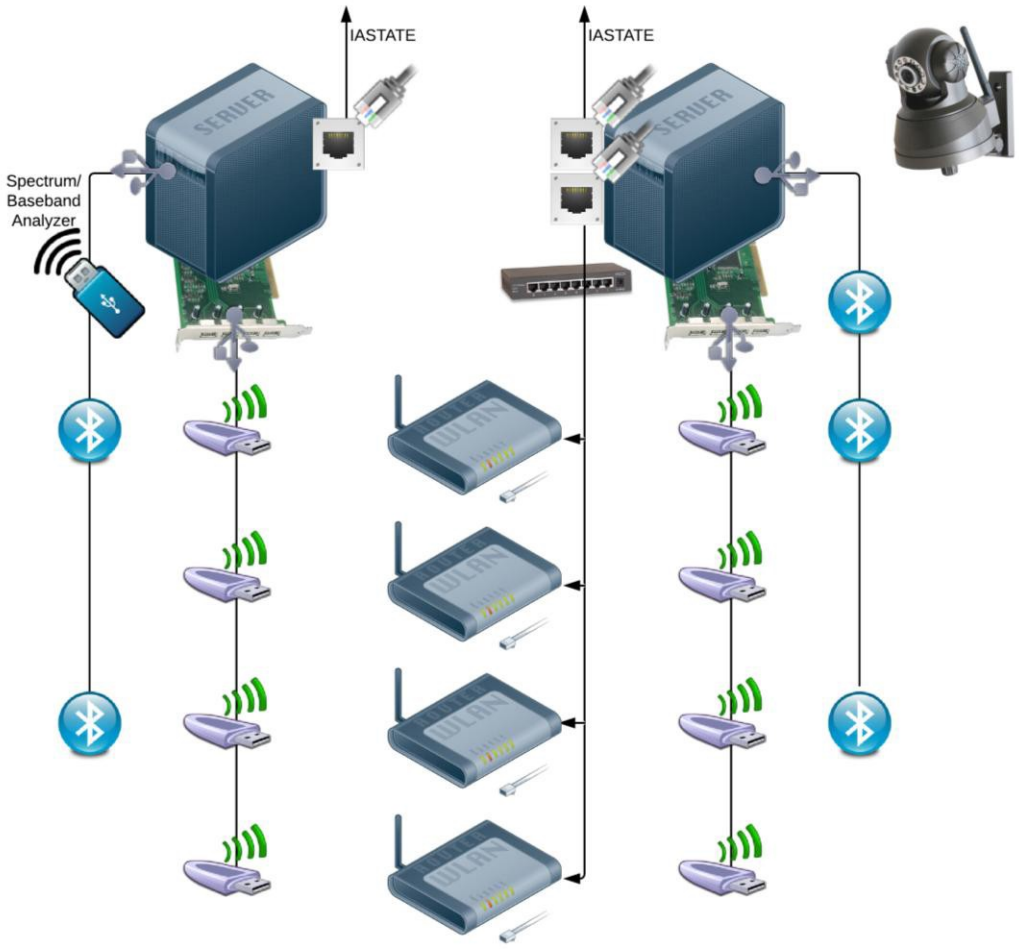


Figure 5: Full-system hardware architecture (servers depicted as complete units for simplicity)

The final goal of this project is to implement a functional GSM OpenBTS (base transreceiver station) network on the ISU network, using USRPs (universal software defined radio peripherels) from National Instruments. The project needs to implement a small-scale implementation of the OpenBTS (single USRP). The main functional requirement is that implementation of that the interfacing with the USRPs is done via LabVIEW.

Firstly, we need to understand what are the functionalities of USRPs, what is SDR (software define radio), what is BTS (base transceiver system) and how can we use our resources in order to implement an OpenBTS.

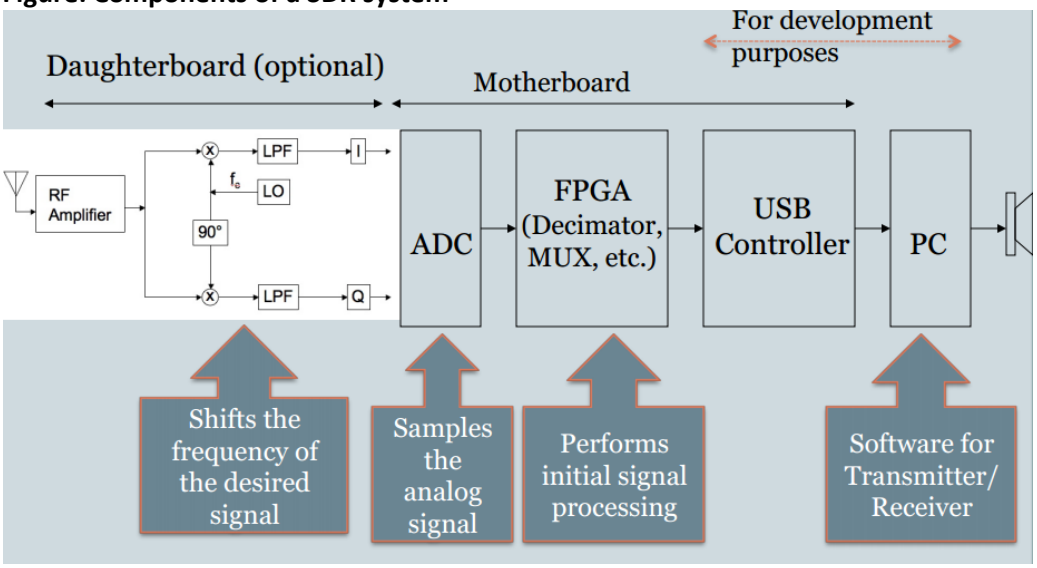
Software Defined Radio (SDR)

A software-defined radio system, or SDR, is new technology for implementing radio communications systems. Components that have been typically implemented in hardware are instead implemented by means of software on a personal computer or embedded system. Implementations of radio communications in software have lots of advantages over traditional hardware implementation.

Some of the advantages are:

- ✓ Makes communications systems reconfigurable (adapting to new standards), keeping the hardware configuration the same.
- ✓ Upgradable. Modifiable. Whereas, traditional hardware have fixed design and can only provide limited implementation of filters.
- ✓ Flexible enough to avoid the limited spectrum assumptions.
- ✓ Ability to implement cognitive/smart radio.

Figure: Components of a SDR system



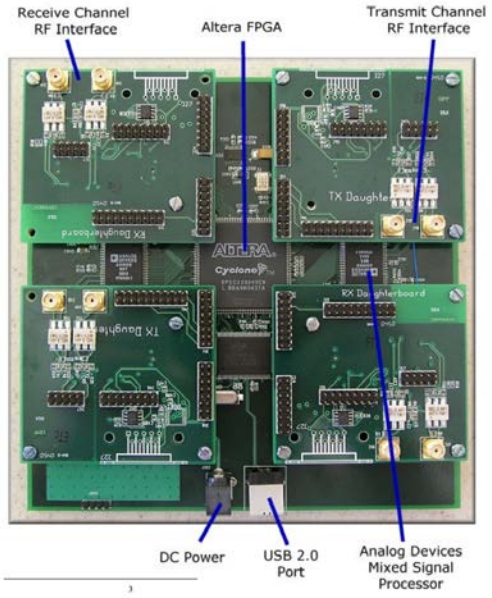
USRPs:

Universal Software Radio Peripheral products are computer hosted software radios. They can be connected to a host PC through a high-speed USB or Gigabit Ethernet Link and can turn the standard host PC into wireless prototyping platform. USRPs are commonly used with the GNU Radio/LabView software suite to create complex software-defined radio systems.

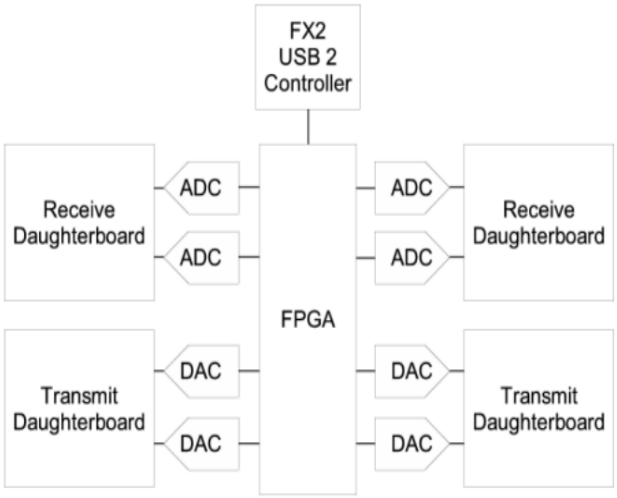
The USRP product family includes a variety of models that use a similar architecture. A motherboard provides the following subsystems: clock generation and synchronization, FPGA, ADCs, DACs, host processor interface, and power regulation. These are the basic components that are required for baseband processing of signals. A modular front-end, called a daughterboard, is used for analog operations such as up/down-conversion, filtering, and other signal conditioning. This modularity permits the USRP to serve applications that operate between DC and 6 GHz.

Figures:

(1) Snapshot of an USRP



(2) Schematic diagram of the hardware layout:



BTS (base transceiver station)

A base transceiver station (BTS) is a piece of equipment that facilitates wireless communication between user equipment (UE) and a network. UEs are devices like mobile phones (handsets), WLL phones, computers with internet connectivity, WiFi and WiMAX devices and others. The network can be that of any of the wireless communication technologies like GSM, CDMA, Wireless local loop, WAN, WiFi, WiMAX, etc.

BTS works by regularly sending beacon signal in its coverage range, registration the mobile station in its coverage and as soon as the mobile station invokes service a free channel is assigned to it. MS (mobile station) sends its voice or data signal to BTS and BTS sends it to BSC (Base Station Controller) and BSC sends it to MSC (Mobile Switching Center)and MSC connects to the other side Mobile Station/PSTN phone/ or connects to SMSC (Short Message Service Center)if the service is for SMS or SGSN (Serving GPRS support node) for internet service. Thus BTS is the first contact for connection or release of a mobile service.

A BTS in general has the following parts: Transceiver (TRX), Power amplifier (PA), Combiner, Duplexer, Antenna, Alarm extension system, Control function and Baseband receiver unit (BBxx).

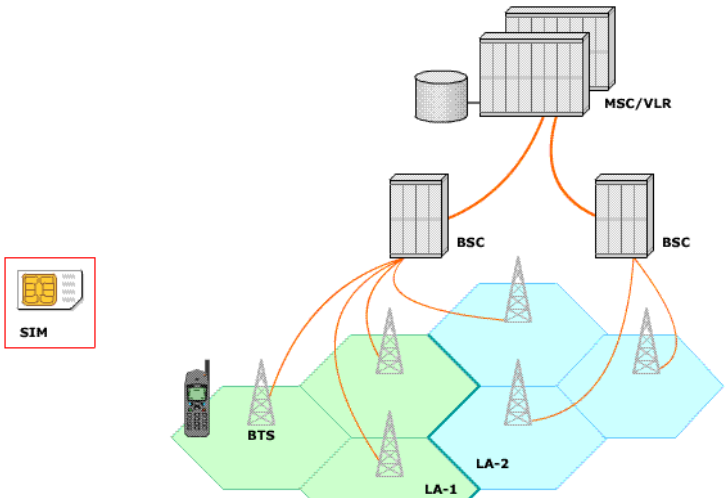


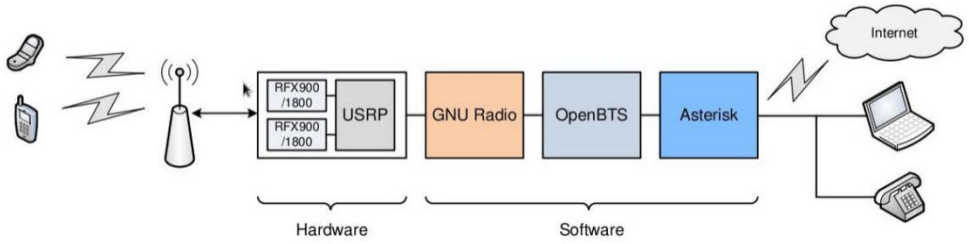
Figure: Simplified diagram of cellphone network structure

OpenBTS (Open Base Transceiver Station)

OpenBTS (Open Base Transceiver Station) is a software-based GSM access point, allowing standard GSM-compatible mobile phones to be used as SIP (Session Initiation Protocol) endpoints in Voice over IP (VOIP) networks.

OpenBTS replaces the conventional GSM operator core network infrastructure from layer 3 upwards. Instead of relying on external base station controllers for radio resource management, OpenBTS units perform this function internally. Instead of forwarding call traffic through to an operator's mobile switching center, OpenBTS delivers calls via SIP to a VOIP soft switch or PBX (private branch exchange).

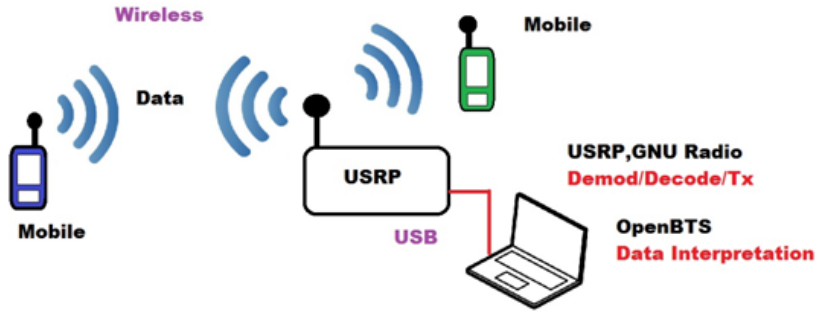
Figure: One implementation of OpenBTS



Our project:

What we will be trying to implement in our senior design is to program the USRP using LabVIEW/ GNU radio and make it act like an OpenBTS. That way we can cause our cellphones to connect using the OpenBTS network, instead of connecting to the cellular network and make Skype calls.

Figure: Below we have tried to illustrate how we might go about implementing on what is required



5.2) Software Architecture

The software required to implement all features of this project is a mixture of operating systems, open-source utilities, custom shell scripts, web applications, and embedded firmware. Generally speaking, the operating systems and utilities are unmodified off-the-shelf products, while all of the supporting software is written by the team.

At the root of all software systems is the hypervisor. The hypervisor is installed as an operating system on the physical host, and emulates physical hardware to allow other operating systems to run inside containers ("virtual machines") within it. The software chosen for this project is VMware vSphere Hypervisor (formerly VMware ESXi), a mature, full-featured product which is licensed from VMware for no cost. It was chosen for its feature set, ease of management via vSphere Client, and a moderately strong team background in its use on top of the hypervisor, two sets of virtual machines are installed, one set per physical host. The first set is an array of machines running the BackTrack distribution of Linux, a security-centric distribution that comes pre-loaded with many of the tools necessary to run experiments on the lab environment. One BackTrack machine is created for each student that will be using the environment, to enable the student to create custom scripts, modify system files, and otherwise configure the system as much as they deem necessary to execute any particular attack, without affecting any other user. The second set is an array of transmitter nodes, again one per student. These will run the Arch lightweight Linux distribution. A few additional machines will be configured in addition to the basic student-accessible machines. One additional machine will be installed alongside the transmitter nodes to serve as the receiver for any transmitter action which requires an endpoint, such as file transfers or Bluetooth communications, and one will be installed to act as a firewall and port blocker between the transmitter machines and the outside world to ensure correct traffic routing between interfaces. One machine will be installed on each host which will configure the host upon startup (this will be covered in greater detail shortly). A final machine will be installed alongside the attack nodes which will serve as the administration node for the environment, hosting the web interface and user documents, and running all necessary back-end processes.

Each machine that is designed to be single-user remotely accessed (all attack and transmitter nodes) will run NoMachine NX Server, which handles remote session serving. Alternately, users may log on via SSH using a terminal emulator if they do not require a GUI. The administration node will be given only SSH access. SSH access will also be enabled on the hypervisor.

The two host configuration nodes are necessary to provide a workaround to peculiarities in the hypervisor. Upon cold boot, the host wipes the local file system, including the configuration for all system processes and the SSH key cache. As these are necessary to perform certain actions on the hypervisor which are essential to the correct operation of the environment, provisions must be made to ensure that if the server loses power, it can recover upon power-up without further intervention from the system administrator. The configuration nodes must store the hypervisor's root password in plaintext in order to be able to execute the necessary commands, so each of these machines is scripted to power on upon cold boot of the host, execute the necessary configuration scripts, and then shut itself off to provide a very narrow window operation to deter exploitation.

The administration node will act as the backbone for the operation of the environment. It acts as the bridge between the user's requests and the hypervisor's actions. The administration node runs an Apache web server, a MySQL database server and a PHP interpreter engine which hosts an information and control portal for the environment. The portal is the first step in a user's access of the environment. After user authentication via PHP sessions, the user can perform a number of actions, such as power on or off virtual machines, launch the NX remote access plugin, reload a virtual disk from backup (if a machine change should happen to have caused it to fail to boot), examine system load and hardware usage, view and message users currently using lab resources, view lab-level data such as spectrum, and more. The portal also hosts lab documentation and tutorials for the provided tools in the form of a comprehensive wiki. In order to accomplish hypervisor-level actions such as powering on or off machines, the portal executes console SSH commands via PHP.

A number of tools will be written for the hypervisor that facilitate remote user interaction as well as simplification of administration. The web server requests actions by calling these tools over a SSH connection to the server. Tools for user creation, deletion and modification for the portal, machine power state control, disk copy and virtual machine initialization, and all other necessary actions will be written as shell scripts and stored in the hypervisor's file-system.

An additional set of scripts will be written for execution on boot by each attack machine. These scripts will set environment variables for the environment such that users can write scripts that may be run on any of the four attack sets without modification (e.g. instead of hard-coding machine specifics, these may be called from environment variables which are updated by the machine on start-up).

On the transmit machines, scripts will be written to automate the sending and receiving of data. This will negate the need for the student to use the machine directly (although they will still be given access in order to do so), and add a level of repeatability and conformity to the environment. The machines will be scripted to log in and out of several common websites, transfer data to and from a remote machine, and perform other everyday user actions.

An overview of the backend software is shown below.

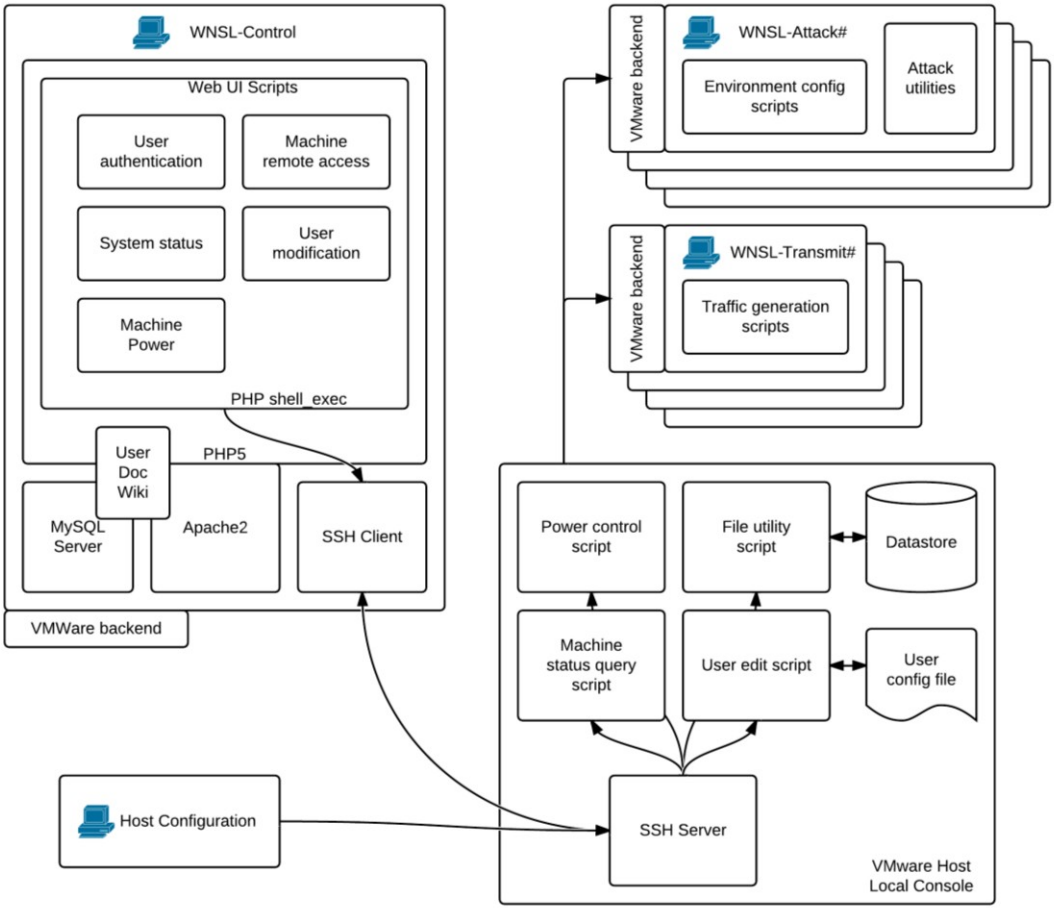


Figure 6: Overview of backend environment (duplicated host details omitted)

5.3) Network Architecture

The network architecture of the environment is designed to allow communication between machines where required, while segmenting separate network blocks and also allowing access to all machines from the outside world. This is accomplished using a combination of virtual and physical networking devices.

Each host is connected to the Iowa State network and thereby the Internet via a physical Ethernet interface. The physical Ethernet is connected to a virtual switch on the host, which in turn connects to all virtual machines. In the case of the attack nodes, this Ethernet interface is unrestricted and allows traffic on all ports; on the transmitter nodes, the Ethernet interface is restricted to only the ports required to contact the machine via SSH and NX via the firewall machine. Each machine also is connected to a physical USB WiFi device and by restricting traffic to SSH and NX on the transmitter, all HTTP and other traffic is forced across the wireless network for possible interception.

The array of routers is connected via a switch to a second physical Ethernet interface on the host. This is run through the firewall and eventually connects to the outside world. All administrative machines must also be given access to the Iowa State network in order to operate correctly, although they may be given access via the firewall machine with a lenient security profile implemented for some added security.

A simplified diagram of the network topology is depicted below.

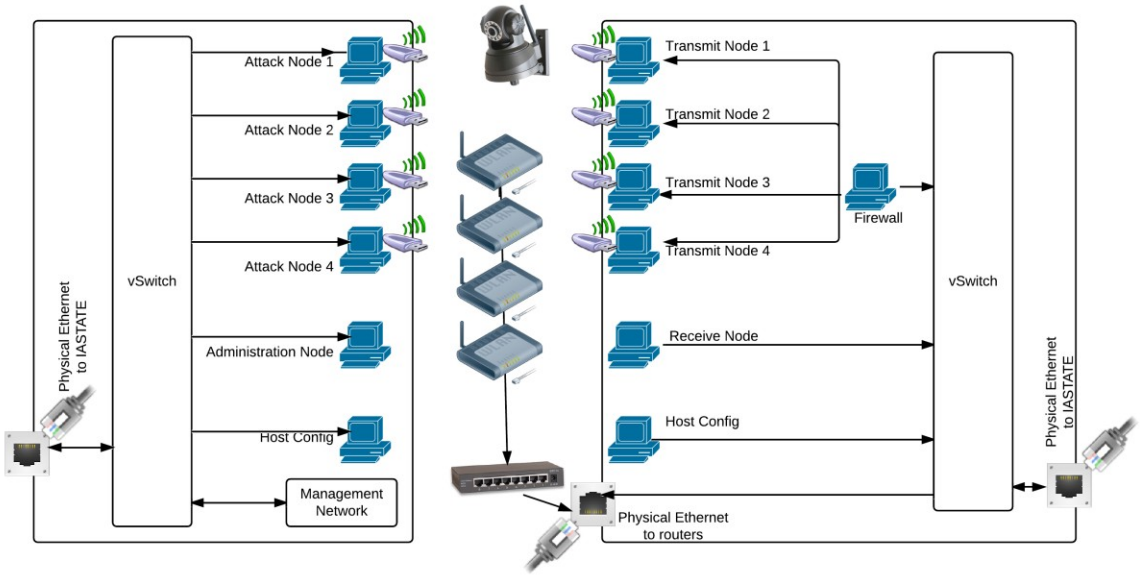


Figure 7: Full-system network topology

6) System Module Design

As much of the system is assembled from off-the-shelf components in standard configurations, extensive clarification is not necessary. As far as installation of physical components is concerned, brackets for secure wall, ceiling, or shelf mounting (depending on client specification) will be provided for radio components. The server itself will be installed either in a rack-mountable enclosure or a modular ATX case, again depending on final installation location and client request.

The virtual machines will be configured with a standard set of hardware using VMware's suggested settings for the guest OS, with the addition of wireless networking devices as necessary for the particular machine being designed.

The backend scripting framework utilizes standard Linux shell commands and VMware's own command-line tools to perform several hypervisor-specific functions. These commands are run on the hypervisor's console via SSH requests from the control node, which are generated using commands embedded in the PHP web interface to the environment and executed via PHP's shell_exec function. This function runs under privileges of the webserver user, but because the functions are relayed over SSH, the scripts are in fact run with the permissions of the remote host. A privileged user is created on the hypervisor to host these commands.

To handle authentication, the webserver user's RSA key is stored on the hypervisor. This prevents the server for asking for the privileged user password on every script execution.

The script architecture is designed to provide several important functions. First, it allows machines to be powered on and off. This is required to accommodate all users of the environment. With only four available radios, only four user machines can be connected to radios and powered on at any given time. The assignment of users to machines can be done in two ways; either four virtual machines can be created and all users can log on to these machines directly, or one virtual machine per user can be created and the radios dynamically assigned as users log on. The former scenario is much less complex, but users may be able to modify the environment in ways that affect other users' results. The latter requires more machines and some backend scripting, but allows each user a unique environment, which they can edit, at will, without affecting other users. This latter approach is what is attempted in this project.

When a user requests to use a machine, the scripts first determine which radios are currently in use. If there are none available, the user is not allowed to log on, or may log on without radios (for retrieving files or other offline use); if one or more is available, the user is allowed to choose a radio set to use. It patches the radio's configuration information (USB device address) into the virtual machine's configuration file and then powers on the machine. After the machine has booted, the user is redirected to a Java launcher for the NX client, and they can begin to use the environment.

It is conceivable that users will switch radios from logon to logon. Boot scripts on each virtual machine will attempt to patch this by setting the interface MAC addresses and establishing environment variables to replace key radio-specific data. In this way, users' scripts can be written to run on any machine. Given the option to choose a radio set, users may alternately elect to wait until a specific radio is available in order to work on their scripts if problems should arise from changing hardware.

Because users will be given low-level access to the operating system, it will be possible for the user to damage their machine. The web interface will give the user the option to restore the virtual disk image from backup, using VMware's command-line disk cloning tools.

Performance and environment data will be presented to the user, as a means of displaying the current status of the radios (e.g. in use versus available) as well as environment conditions such as local radio traffic spectrum.

A set of scripts written in Python will run continuously on all transmitter machines. These scripts will continually run experiments such as logging in and out of common sites such as Gmail, Facebook and Twitter, transferring files, and connecting and disconnecting from the access point. These actions are performed in hopes of generating live, capturable data for lab users to experiment with. Some specific actions will be able to be requested by the user, such as switching encryption mechanisms, and the user will be given access to the machine to generate custom data, but for the most part the transmitter nodes will act without user intervention for simplicity.

A number of scripts will also be made available to the system administrator. These scripts will be responsible for initializing the lab for a set of users; user creation, modification and deletion; virtual disk and machine configuration; and other tasks. These may be run via direct SSH to the hypervisor, but will generally not be made available on the web interface for security.

7) User Interface Design

The backend of the site may actually run the environment behind the scenes, but the user interface is no less important. The tools and scripts will be wrapped into an attractive and easy to use website. The interface will be simple buttons and color-coded displays, and will be heavily documented. User authentication will be provided by PHP sessions in order to secure a user's own machine from use by others.

In addition to interfaces to the various script actions, the website will contain documentation for the tools and utilities available to the user as a part of the BackTrack design suite. This documentation may reference public websites on the tools, or when these are not available will consist of team-written documentation. The user documentation model envisioned for the lab (large numbers of small articles with independent topics) ideally fits organization by a wiki. A MediaWiki installation will be used to host the user documentation, forming an easily-updatable and convenient-to-access knowledge database. Administration documentation will instead be provided via a primarily offline delivery mechanism for security.

An important part of the project is the creation and walkthrough of common attack scenarios to carry out within the environment. These scenarios will tend to exemplify the use of one or a number of related tools, and will serve as tutorial examples that may be built upon by the user to create new and interesting experiments.

8) Testing

Given the number of different components in this project, individual components as well as the system as a whole will need to be extensively tested. To ease this process we have broken the testing down into a few distinct parts. First and foremost, the hardware architecture will need to be evaluated to ensure the platform will be able to support four lab users at any given time. Given our current design, this means each physical machine will have to support at least four virtual machines, with the possibility of one or two more for administrative purposes. After the platform has undergone adequate testing we will move on to testing the configuration of individual virtual machine as well as the wireless hardware. Once the configuration has been validated we can move on to testing individual components of the sandbox environment. For each tool presented in the sandbox we must provide a 'proof of concept' to show the lab users how to use the tools and hardware provided for them. After all of the previous tests have been proven to work we will move on to testing the system as a whole.

8.1) System Architecture

Each of the virtual machines will need to meet, at least, the minimum system requirements for the given operating system. This first check will help insure an enjoyable user experience. Next we plan to run different benchmarks on the individual virtual machines. Among these benchmarks are

- Xbench – Xserver benchmark
- CacheBench – Measures bandwidth of the memory subsystem (CPU cache and RAM).
- UnixBench – High-level Linux benchmarking suite

We can then compare the results to the results published for each operating system. This will allow us to compare the performance of the virtual machines with non-virtual versions of the same operating system.

8.2) Component Testing

This set of tests will deal with the basic configuration of the lab environment. We will test that individual components of the lab – wireless access points, virtual machines, wireless cameras, spectrum analyzers, and so on – will be able to communicate with each other properly. To do this well will have to manually test the components and analyze the network traffic. We can do this with a tool called AirTraf. AirTraf is one of the first 802.11 network analyzers and can be used to ensure proper communication using the 802.11 protocol. In addition to testing that the hardware is communicating correctly we will also have to test that our custom administration scripts are working correctly. These administrative scripts will be responsible for the booting of virtual machines by the lab user, presenting the lab user with what hardware resources are currently available and patching configuration files to ensure that any virtual machine will be able to use any of the wireless hardware. Testing must ensure that these scripts are working correctly for the lab to be functional.

8.3) Proof of Concept

Once all the previous tests have been completed we will move on to testing the individual components provided in the sandbox environment. Each tool we plan to provide to the user will have to be documented as well as proven to work. These 'proof of concept' tests will also function as a tutorial on how to use the tools to execute different attacks inside the lab environment. Once we have completed

these tests on a single machine we will have to verify that they do not interfere with other users who are trying to conduct their own experiments.

8.4) Overall System

We plan to test the overall system by having the Spring 2013 class of Computer Engineering 537 perform a few of the proof of concept demonstrations as well as play around with the system and see what they can do to break it. This test will be crucial to evaluating the system we have designed. We hope to have the system operational sufficiently long before the end of the semester in order to allow adequate testing time.

9) Functional Requirements

9.1) OpenBTS & USRP Requirements

The system shall implement a functioning OpenBTS sub-system using USRP's as communication hardware.

Fit Criteria: USRP's run on the GSM network and are the hardware available for with OpenBTS to communicate with.

Rationale: OpenBTS will be provide the communication software to give function to USRP's.

The system shall act as a wireless base station for the purpose of injecting, receiving, and sending wireless data.

Fit Criteria: The purpose of the lab is to non-destructively interact with wireless traffic.

Rationale: The basic ways to interact with data are create/rename/update/delete – as a wireless base station students will be able to entertain all possibilities.

Between OpenBTS and USRP's, Labview shall be used to transfer data and interact as an interstitiary application.

Fit Criteria: OpenBTS doesn't directly interface with the USRP's.

Rationale: In order to create a wireless base station the hardware must have a line of communication with the software.

The existing web control interface shall be updated to reflect control of the new USRP & OpenBTS functionality.

Fit Criteria: Users control the system through a web interface.

Rationale: In order to utilize our updates and changes, users will need to have an updated and functional interface to do so.

Ubuntu virtual machines will be created to interface with the USRP's and run the OpenBTS system.

Fit Criteria: Users are interacting with the system entirely remotely.

Rationale: OpenBTS is software designed to run on *nix systems; LabView is also compatible with *nix systems.

9.2) Existing System Requirements

The existing system consisting of backtrack linux, webserver, and attack clients shall be resurrected and made functional again.

Fit Criteria: The existing system maintains much functionality that is useful to build upon to further the project goal of student lab experiments.

Rationale: The existing system is already built and simply needs to be reconfigured for new hardware.

The existing system shall be made current to any security patches.

Fit Criteria: The existing system was made a couple of years ago and deserves security attention.

Rationale: Many security patches occur every year and there are bound to be some we need to implement.

The existing system shall be remotely accessible through VMWare vSphere Clients and VMWare workstations.

Fit Criteria: This is a remote access, wireless lab for students.

Rationale: Students will be unable to gain physical access to the server / peripherals and must be able to still interact with the lab.

10) Non-Functional Requirements

10.1) Documentation

Updated documentation shall be created and merged with existing documentation where possible.

Fit Criteria: The system is to be used for students for a number of years.

Rationale: To maintain longevity, the system must be well documented to have a hope of being well maintained.

Updated system passwords and account access shall be enacted.

Fit Criteria: Passwords are regularly compromised.

Rationale: System administrators and maintainers shouldn't have to worry about the insecurity of passwords or have to hunt for them in many places.

10.2) Legal

Design actions shall be taken to prevent end-user students from breaking the law by any means of illicit activity.

Fit Criteria: System must be legal to operate.

Rationale: If a student breaks the law unknowingly using a lab from Iowa State, the University is but in a compromising legal position.

System shall conform to any and all operational and environmental requirements and regulations.

Fit Criteria: System must be legal to operate.

Rationale: System must conform to all regulations imposed by governing bodies.

10.3) User Experience

Overall system user experience shall not be learning prohibitive.

Fit Criteria: The user interface must be intuitive to use.

Rationale: If end-users are unable to use the software the system functionality is nullified.

11) Risk and Mitigation

11.1) Risk: No access to web server

Mitigation: We can request members of the previous group to send us the web server materials, and create a virtual machine image from it, so that this problem is never encountered again by us, or subsequent groups.

11.2) Risk: USRP/OpenBTS has unknown level of compatibility with a virtualized environment

Mitigation: Depending on levels of compatibility, we could limit the number of virtual machines that have access to the devices, or in the extreme case, run this portion of the project on a standalone server.

11.3) Risk: Serious ethical and legal boundaries involved in the project

Mitigation: Carefully explore and define these boundaries. Make them absolutely clear, and possibly limit what software can do to avoid crossing these boundaries.

12) Maintainability

The goal of the Wireless Computer Security Lab is to provide the students of the Computer Engineering 537 class a practical environment for experimenting with wireless security. Since this class is taken by distance education students, the lab must be setup with special considerations that a normal computer lab may not consider. One consideration is how the students will access the lab as they will not have physical access to the lab equipment. Another consideration is the physical environment the lab will reside in. Things like temperature, space and wireless signal interference are critical to the operation of the lab. Finally, due to the type of experiments, the students will require a fairly high level of control over the equipment. The lab must be built with this in mind.

Since some experiments will require the student to have direct control over different equipment, there is a concern over the possibility of the student breaking the lab environment that has been set up for them. This concern makes the lab a prime candidate for virtualization. For example if a student were to delete an important configuration file while editing it the system may no longer work the way it was intended to or work at all. However since the machine is a virtual image, the system administrator would be able to back up the machine to a previously working state very easily. Besides making the lab easy to administrate, virtualization also makes the lab more cost effective. Instead of having one physical computer running one lab session for one student, the lab will be able to run multiple sessions on one physical machine for multiple students. Now that the amount of physical equipment needed is significantly less, the amount of space needed to run the lab is also significantly less.

By choosing to run the lab in a virtual environment as well as having the lab only remotely accessible to students, the amount of space required to set up and run the lab is small enough not to warrant entire room. It has been decided that, since the amount of space required is relatively small, the lab will be set up and run in the Nuclear Engineering Building along will other machines of the same type. This raises the concern that with so many computers so close together that the temperature may excide the level of safe operation. This has yet to be seen and probably will not be testable until the weather becomes significantly warmer. However, it still remains a concern. Another concern that arises from having so many computers so close together is with so many wireless signals on the airwaves there may be a considerable amount of interference. To counter this when the machine is set up in Nuclear Engineering an audit of wireless traffic must be done. Once this information has been analyzed the correct configuration should allow the lab to co-exist with the rest of the computers.

13) Conclusion

Although exact specifications for how the tests should be judged have not been agreed upon yet we have laid out an extensive plan for what needs to be tested and how we plan to execute those tests. We are confident that once the system is able to pass each phase of testing that the end product will be fully functional. We are also confident that the choices we have made in the design of this system are the best choices in terms of functionality per dollar. Every decision was made with the idea of creating the most versatile lab environment within a reasonable budget. Using USB devices allows us to keep costs down and consolidate more VMs to each host. Using open source software allows us flexibility in choosing an operating system and costs us nothing. Opting for a virtual environment allows for greater system uptime along with keeping overall hardware costs down. Although we are confident with our decisions we are leaving previously discussed options open. To the best of our research, we could not find any lab environment like the one we were asked to design. With this in mind it is likely there could be unforeseen obstacles as we move toward implementation.