Garrett Gonzalez-Rivas

Mr. Montgomery

HCVSD CSAEA III

August 21, 2021

# A Detailed Look Into the Operations of Establishing a Server

Some of the footnotes add another level of technical detail and complexity, while the rest are simply general supplemental text — with a much smaller number being semi-humorous meta-commentary, please do not hesitate to refer to them in the endnotes section.

# An Introduction and Overview into the Project

When this project was established, its purpose was to act as a substitute for the third year of the Computer Science and Applied Engineering Academy. For this purpose, a server was, first, to be set up and configured and then hardened against cyberattacks, exploitable vulnerabilities, and malicious activity. In order for the server to be "configured" it had to be running Windows Server, be able to be managed remotely, have at least two virtual machines hosted through Hyper-V, and host a website using Microsoft IIS Web Server.

For the report on this project, I broke the project into the ten following sections: Choosing the Right Operating System, Setting up the Hardware, Installing the Operating System, Setting up Windows Server with Udemy, More YouTube Tutorials, Creating a Website, Setting up the Penultimate Windows Server, Learning More About Internet Protocols, One Last Time, and Intranet vs Internet. This section, technically making the total report being eleven sections[1], is focussing on a synopsis of the project (hence its name "An Introduction and Overview into the Project") and overall what I learned from the project at a very high level. Being a synopsis of the project, the information in this section will be repeated throughout the entirety of the report, simply with significantly more detail.

The first section — Choosing the Right Operating System — is, as its name implies, the process in which I went through to determine what version of Windows Server I would run. The first two sections also highlight an aspect of how I work on project (and workloads in general) that I had not noticed as much in the past that I now know to make an effort to correct, however I will expand on this later. While choosing an operating system on consumer electronics is usually relatively simple (with the main choice being between Windows and Mac), this is not so in the world of enterprise-grade operating systems. In fact, I found that choosing a version of Windows Server was most akin to choosing a Linux distro (short for Linux distributions), in that they are all relatively similar and operate off of a small number of different kernels/architectures, with their intended application affecting their unique implementation.

Now with Linux distros, the differences are less significant and can be altered by downloading a new desktop environment or package manager, but with versions of Windows Server their differences cannot be changed and the nature of these differences are more in the capabilities, features, and size of the operating system. Furthermore, other than just editions of Windows Server, there are also years, and Microsoft still sells both Windows Server 2016 and Windows Server 2019[2]. Unless you are working with legacy software or server compatibility[3], there isn't much of a reason to go with an older year of Windows Server.

Within the years of Windows Server, there are "editions", each edition has a unique set of capabilities, with there being a higher number and more powerful features the higher level it is, but the more expensive the license is. The cost of the license was not going to affect me,

however the larger the operating system is (the more overhead), the slower it is, and so there is a happy medium between capabilities and overhead. For me that balance was found with Windows Server 2019 Standard Edition.

The second section — Setting Up the Hardware — covers the process in which I purchased the server, repaired the server (after a fashion), and finally some configuration of the hardware. When I was working on this section of the project, I initially spent just under three weeks researching different types of servers, tower vs rack mount servers, using old PCs as servers, etc. — but by the end of the three weeks I ended up going with a dell PowerEdge R710 that I was able to find on eBay for a relatively good price.

Unfortunately, when the server arrived, it arrived with several issues that I discovered upon trying to start the server. Due to the poor condition of the box, I thought that it was likely that some of the connections may have gotten loose, so I reseated many of the components and almost everything worked, I only had persistent problems with the iDRAC and the RAID controller. The iDRAC was being recognized by the system but was flagged as being inoperable, so I was able to get a replacement from the seller at no cost.

The RAID controller, on the other hand, had passed all the tests from the seller before being shipped out (and wasn't even being detected as a failed part in the diagnostics and hardware menus) so they refused to send out a replacement. Due to it being dead-on-arrival, I would have been covered by eBay's purchase protection. Before trying to make a claim (or

simply purchasing a new RAID controller), I decided it was worth the effort to completely dismantle everything connected to the RAID system and reassemble it. In fact with some of the components I reseated them several times rapidly (a trick I learned from my uncle who ran the IT department for several years; by rapidly seating and unseating the component, any corrosion or anything else on the contacts would be rubbed off and give a better chance of a connection). Doing so, I was able to get everything in the server fully operational.

The last thing I did was decide what kind of hard drives I would use. The first decision was SAS or SATA, and while SAS is more ideal for this application, I already had several SATA drives (I did order one SAS drive, however it arrived dead). I then had another decision: CMR or SMR drives. Because of the nature of the writes in a server, CMR drives would be the preferred option for everything but the backup drive, however the only 3.5” drives I had were SMR, and by an ironic twist of fate, all my 2.5” drives were CMR making the backup drive be a CMR HDD.

The third section — Installing the Operating System — covers how I got the ISO, how I got the license, and my choice in how to configure the management of the server. The actual step-by-step installation and settings chosen are all present in the section “Setting up the Penultimate Windows Server”. I was able to obtain the ISO and license key for Windows Server 2019 Standard Edition through Microsoft Azure’s Student License (under the software section of the educational resources). Once I downloaded the ISO and created an installation media using the program Rufus, I booted the server to the installation wizard and installed Windows Server.

Now that I had Windows Server installed, I created a strong admin password, and configured the management (local and remote) of the server. For the remote management there were four options: RDP, WAC, RSAT, and SSH. RDP has many security vulnerabilities, and WAC is not a standalone management option (design to run in tandem and supplement RSAT). RSAT, unfortunately, only runs on a Windows computer, and while I have a computer running Windows 10 it is broken (partially operational), so I went with the only option I had: SSH. After installing and configuring the SSH client OpenSSH on the server — I chose open SSH because it is one of the two clients I have used in the past — and did a small amount of 'management'[4] on Command Line through the connection.

The fourth section — Setting up Windows Server with Udemy — is when I realized that I needed to do more research into how Windows Server actually functions and is applied, so I got a Udemy course. I have always worked with a natural trial-and-error approach, and while it may not always be the fastest way to get to the solution, it means that I almost certainly will have gained a better understanding of how everything works. This time, however, I had to come to terms with the differences in between trial-and-error and trying to solve a problem blindfolded.

With that realization, I purchased the aforementioned Udemy course on how to use Windows Server 2019. In order to get ready for the Udemy course, I did my first wipe and reinstallation of a fresh copy of Windows Server. I would do this several more times, but this time I made an error that invalidated my license key for Windows Server 2019 Standard Edition

and Azure only gives you one. I considered switching to Datacenter Edition, however the inactivated Windows Server 2019 Standard Edition has nearly all of the features (compared to its activated counterpart) and its main limitation is that it will only run for ninety days, so I stayed with it.

Despite it being inactivated, I had a fresh install of Windows Server, and I followed along step by step with the Udemy course. Ultimately, this led me to setting up things such as AD DS and DNS roles (and several more) on my server, and while this did not help me with my project, I did learn how they work and how networks and server clusters are managed within enterprise environments. What the Udemy course did help me with, was that it taught me how to navigate Windows Server, where everything was, and how to control everything. If I ever need to, I even know how to set up both remote and local print services (port 139).

The fifth section — More YouTube Tutorials — was likely the shortest section, however without it I still would have been unlikely to have been able to complete the project. Now that I knew how to use Windows Server, I still needed to get a better understanding of how to configure the different roles and features in Windows Server. Unfortunately, the Udemy course had rather poor instruction in that category, so I searched for videos detailing how to set up each individual role and ended up putting them all together in my penultimate and ultimate builds of the server.

The sixth section — Creating a Website — takes a slightly different direction. After having spent weeks researching through documentation, forums, YouTube, and more, I needed to take a break from that and program something. I have created several websites in the past and it is something I quite enjoy, so I decided that I wanted to create as much of the website as I could by hand within the time frame and without compromising on the aesthetics or functionality of the website. To do this, I started with a basic template for Bootstrap Studio, added in some more elements, and then exported it to my IDE even though everything was still in Lorem ipsum and all the images and icons were placeholders.

Opening this up in my IDE of choice, CodeRunner 3[5], and opening a preview of the website in Safari, I started making edits to the website. I first started by replacing the filler images (I put a new folder of my photos in the assets folder and just specified a new path in the index.html file), the icons, and the links. It was easy to substitute out the filler images and icons, however rather than just pasting in the href for the links, I also added to the anchor element. I added a line of code to ensure that the link opens in a new tab, runs as a new process, and cannot access either the refer header or the window.opener property. While this is not a major security enhancement, it is a relatively simple one that is easily implemented, and is certainly a best practice.

The final task in creating the website was to replace all the Lorem ipsum. This only totaled to being five medium sized paragraphs, about a dozen short blurbs, and filling in some personal information. Overall in creating the website, I spent between two and two and a half

weeks creating it, however during that time I didn't let the server lay dormant, and continued doing research during the creation of the website and after about two weeks of working on the website I began doing some of the work that I describe in the next section. Working on both the server and website simultaneously, however, slowed me down and it took me nearly another week to export a final version of the website to the server.

The seventh section — Setting up the Penultimate Server — was originally supposed to be known my a different name: "Setting up the Server for the Last Time", unfortunately its name had to be changed for reasons I will get into shortly. In order to begin this section of the project, I reinitialized the RAID array and reinstalled Windows Server. Once I had Windows Server installed and a strong password created for the admin account, I started the actual configuration of the OS.

In configuring the server, there were options for both security and the roles that would run on the server, for general server function I set the server name, set the correct timezone, and installed and configured Windows updates. Next I set up the iDRAC which is another method to manage the server remotely, however I never ended up using it as I configured SSH instead. I then configured the remote management, networking, and the Windows Firewall. Next I installed the roles: Hyper-V, IIS, and Remote Access, and their accompanying Features.

Opening the Hyper-V manager I created two Gen. 2 virtual machines and allocated each 16GB of dynamic memory and 511GB of storage. Each VM was initialized with a Windows 10

Pro for Workstations installer ISO mounted  and I booted each VM from this and installed Windows. I then configured each VM with a shared Microsoft Account (which also used a very secure password), configured the VMs to use relatively secure passphrases and installed all the updates for the VMs. I then attempted to configure Microsoft IIS Web Server, however I soon realized that I needed to learn more about how internet protocols worked before I would get IIS fully functional, which is what the next section is.

The eighth section — Learning More About Internet Protocols — just like its name suggests and as I said before: is about how the different internet protocols work, interact, and how they affect deployment of a web server (IIS web server, specifically). The marker for what separated this section in the project from the section before it is that I reached out and got help for the first time; I reached out to my uncle. Unfortunately, he didn't remember much about this topic (it is not the area of computer science he teaches), however he had learned about it and worked with it in a limited capacity before, so he was able to point me in the right direction and guide my following research.

For me, this was probably my favorite section, I really enjoyed learning more about how the internet and routers work and how computers communicate with one another. I had a very fundamental understanding of how the TCP/IP system worked, however it was all at a very high level and completely theoretical. My research was split into three categories: LAN/WAN, DNS, and SSL.

When a computer is sending information within the LAN, they first need to know what IP network their address is on and the IP network the designation's address is on. This is accomplished by the computer evaluating both their subnet mask and and the destination's subnet mask. Because they match (they are on the same IP network), the computer realizes there is no need to forward the request to the NAT router (default gateway). The computer uses an ARP to identify the MAC address of the destination, and then sends the data through a series of TCP packets.

When a computer is sending data though the WAN, however, the process gets slightly more complicated. The computer does the same evaluation of the subnet masks, but since because they don't match, the computer must forward the data though the NAT router. This data is encapsulated into a series of TCP segments, which are then encapsulated as TCP packets, and then encapsulated in frames. These segments include the source and destination ports, the packets include the source and destination IPs, and the frame includes the source MAC address and the MAC address of the NAT router's internal NIT.

Once the NAT router receives the data, it logs that when data comes back to it on that source port (from the TCP segment), to forward it within the LAN to the computer with the IP from the TCP packet. The NAT router then re-encapsulates the TCP segments with new TCP packets using its publicly accessible IP address as the source address (but the same destination IP address), and then encapsulates these new packets in a frame with the NAT routers external interface's MAC address as the source and the MAC address of a server within the ISP. Within

the ISP's routing system, the TCP packets will be re-encapsulated in new frames, however the packets themselves will not be modified in transit.

The NAT router knows where to return the incoming data from the WAN because when the router receives packets with the router's external IP address and those packets contain segments with destination ports that match the source ports the NAT router received (and logged) from a computer on its LAN, it knows to forward those packets to that computer. However, this means that if a computer had not forwarded packets though the NAT router on a particular port, the router would simply drop any incoming packets with that destination port because it would not know where to forward it — that is where port forwarding comes in. Port forwarding is a way to make a rule that whenever the router receives data on a particular port, to always forward it to an IP address, which was in my case ports 80 and 443 on my server.

As for how people were to know to send these HTTP and HTTPS requests to my router's external IP, that is where DNS comes in. DNS is basically a phonebook for the internet, and by registering a domain, and creating DNS records, I can direct anyone who types that domain into a browser to my router (which then forwards it to my server). However, because ISPs use DHCP to assign the external IP to routers on their network, you have to update these DNS records every time it changes, or use a DDNS client to update them automatically, which is what I did.

The final aspect was making this connection secure, and while DNSSEC and HSTS (both of which were implemented after HTTPS/SSL) can improve the security of a website, the most

important thing is using the TLS protocol, or as it is often referred to: SSL. SSL/TLS is a protocol for securely encrypting internet traffic and works by using asymmetric encryption in the TLS handshake to generate a one-time-use session key which is then used with symmetric encryption to encrypt the data being exchanged by the computers. TLS also uses MAC (different then MAC addresses — remarkably good naming) to ensure that the data both came from the correct web server, and was not altered in transit. In order to use the SSL/TLS protocol, you need an SSL certificate, which I acquired in the next section.

This section, the ninth section, has the triumphant title — One Last Time — however it is only the ninth section for a reason. This sections begins with the penultimate build of the server, however for a reason that is still not fully known to me (I believe it was a drive failure although I am not completely sure), upon a restart of the server for installing updates, the server booted to an error saying my installation of Windows was damaged. To make things worse, I had neglected to configure Windows Backup Services.

After trying several times to recover the server[6], I ended up cutting my losses, replacing the unrecognized drive, and initializing yet another new virtual disk. This unfortunate accident happened at about 10pm, and I ended up staying up — with the aid of enough caffeine to kill a horse[7] — until 2pm the next day completely recreating all my work up until the point I was at with the penultimate server. I firmly believe that, if I didn't have to wait for everything to initialize, install, download, load, and anything else along those lines, I could have had the server completely rebuilt in an hour

Some of the changes that I implemented when configuring this new rebuild of the server, however, were tantamount in getting IIS to function as intended. They could have, however, been relatively easily implemented on the previous complete build of the server. Another thing I made sure to configure, was Windows Backup Services. From here, I simply had to configure IIS and implement what I had learned about regarding SSL certificates and internet security.

The first changes I implemented were making the binding protocol HTTPS and enabling and configuring HSTS. Now to activate HTTPS, I had to have an SSL certificate. I unfortunately had to change both my DDNS client, and domain provider to be able to do this, and it wasn't until my second SSL certificate that I got it to work, however I was able to install the certificate to the server and use the SSL/TLS protocol. I then configured DDNS with my new domain client, created DNS records, and configured the security settings within my domain provider. Doing all this, I was able to type my domain into safari and access my website using the SSL/TLS protocol.

This now leaves only the tenth and final section — Intranet vs Internet — which got its name due to a harsh reality about the testing environment I had been using. All the tests I had done up until that point regarding connectivity, HTTP rerouting, the SSL/TLS protocol, and more had been done on the intranet, or the LAN. This section's beginning is when, for the first time, I discovered that my web server was not accessible though the internet, or the WAN.

Discovering this I applied one of my standard debugging processes: eliminate everything that can't be the problem. In order to not be on the same LAN but still be able to troubleshoot from my desk next to the server, I used my iPhone and a bluetooth tether to my Mac. I tried connecting from multiple browsers, I then checked the firewall on the server (although because I could connect through ports 443 and 80 on the LAN I didn't believe it to be the problem). I also tried bypassing my DDNS and domain provider and trying to connect using my router's public IP (and even tried specifying what ports to hit). Next, I tried disabling HSTS and connecting through HTTP, and finally I then tried connecting through a series of different ports (alternates for HTTP and HTTPS as well as unusual ports like port 24).

Unfortunately, none of these attempts bore fruit. I had a suspicion that it was a firewall on my router, however extensive searching, researching, and going through the router's controls proved that my router did not have a user-configureable firewall. Unfortunately, my mother was not comfortable with me putting OpenWRT or another alternate OS on our router and I was not able to get another router to use temporarily for this project. I then went to scanning with BlackArch for reconnaissance and enumeration. I was unable to discover anything particularly useful with this approach either. Even as I am writing this report, I am trying new things, however the more things I try the less I expect something to work.

While I was unable to get the web server to be fully functional with its intention to be accessible over the internet, I ultimately found this project to be one of the best learning experiences I have ever had. I am leaving of this project with many lessons, not only regarding

the subject of the project itself, but also how to better overcome failures, manage workloads, and more.

In this project, there were many set backs and down-right failures, and I had to start over configuring the server many times. If I hadn't learned to be able to move past these and learn from them rather than focussing on the failure itself, I would never have been able to come as far in the project as I have. I also had to be able to curb my perfectionist nature and go with "good enough" to be able to keep the project going. When I was working on the website, I had to accept that it was not the focus of the project and I couldn't sink all my time into it. Due to this, I had to leave it with some of the default icons, I had to comment out entire sections I and comment out the Google Maps frame I was (unsuccessfully) tinkering with.

In past projects, in the Academy and in everything else, I would have spent countless hours working on aspects of a project that had little-to-no impact on the actual project itself because I believed that everything had to be 'perfect'. However, I have learned to be able to work past that, at least to some extent, with this project. This was, however not my biggest realization from the project. As for that, it was learning about management of my time and my workload.

I didn't realize this at the time, but in the early stages of the project, (mainly in the first two sections — but also in the fourth section) I was spending too much time going down a rabbit hole and researching while not actually doing anything to further the project itself. One of the

biggest things I gained from this project was how my actions and my approach to the work changed. Looking back, I see this same behavior in nearly every large project I have done in the past. In the beginning I spend nearly all my time researching, planning, preparing, and doing whatever other prep work that can be done but I produce almost no progress on the project itself. In the timeline of the project, I spend nearly the same amount of time researching the operating system and hardware (and how the hardware worked despite that not being directly related to the project) as I did doing all the work in the fourth, fifth, sixth, and seventh sections combined[8].

I have realized that this is not an effective way to actually work on a project, and despite having a slight trip down the proverbial rabbit hole in the fourth section, I was able to keep myself productive for the remainder of the project. I was also able to apply these same new-found skills to other projects I was working on at the time mainly my precalculus class, summer work, and the completion of my Eagle project. I was able to utilize these skills I had learned from these early stages of my server project to get them completed in a much more efficient manner. In fact, without applying the lessons from this project it is possible, and maybe even probable, that I would not have been able to complete them at all.

Now this may seem strange to be talking about being productive, not procrastinating, not wasting time, and not waiting until the last minute to start working since I am writing this report in the middle of August, however there are a few reasons as to why I am, and it isn't procrastination. This is, in part, due to the amount of work and effort I put into the project (and report). From the time I started working on the project in earnest until now, I have only taken

breaks from when I was physically unable to work (which happened several times from changes in meds leaving me unable to get out of bed for sometimes up to days at a time) or when I was focussed completely on another project (during finals for my summer precalculus class I didn't make much progress on the server project for nearly four days). It was also, in part, due to the outcome of the project.

Since I reached the conclusion of this project, about two weeks before this, I continued to try everything that I could think of (or that I was able to find online) and delayed in writing up this report until I was certain that I would be unable to make the website fully functional and accessible from the WAN. After it was nearly a week into August, I decided that the only thing I could do was to write up a proper and full report of the project. Since I started writing this report, I have had several more attempts to remedy the server, however, as of now, none have been successful. In my latest attempt, I installed Windows Server 2019 Essentials Edition on an old computer and completed mostly the same set-up steps for the web server. I had the idea that because the OS running on my server wasn't activated, it might be interfering with it hosting a website. This turned out not to be the source of the problem, however I now know one more thing the problem isn't.

# Choosing the Right Operating System

The first step was to create an installation media for windows 10 server. Luckily, with the Microsoft Azure Student Subscription license, you can access activation codes and download links for Windows Server 2019 (and other supported years of Windows Server) Standard Edition, Data Center Edition, and Essentials Edition; it also includes another server operating System: Microsoft Hyper-V Server 2019. When deciding which operating system to use I researched the differences between them and ultimately decided on going with Windows Server 2019 Standard Edition.

The biggest difference is the server type: the difference between Windows Server and Microsoft Hyper-V Server. [9] Windows Sever is designed to handle the entire infrastructure of a small business or to work in a so-called “cluster” of servers to handle larger infrastructures and provide redundancy. A server running in a standalone ( also called an “isolated”) configuration or a cluster configuration can have several roles. A “Role” is a set of programs that, once installed, initialized, and configured, allow the server to perform a certain task such as hosting a Domain Name System (DNS) Server or a Dynamic Host Configuration Protocol (DHCP) Server.

Microsoft Hyper-V Server, on the other hand, is essentially a Windows Server Core that’s been stripped down with the Hyper-V role preinstalled. Being based off of Windows Server Core, it runs completely through the command line (no GUI) unless you manage it through

Windows Admin Center on an Administrator Workstation. Despite it being a dedicated Hyper-V server that is designed to be managed through the console or remotely, you have full access to all Hyper-V and server management programs, such as Failover Cluster Manager and Remote Desktop Virtualization Host. There are, however, some benefits to this approach. Mainly, there is less overhead and is not only more efficient with computer resources for the OS but it also runs the virtual machines (VMs) more efficiently.

Because I needed to have other roles running on the server, mainly Internet Information Services (IIS), I have to run a version of Windows Server. There are three years of Windows Server that will be able to accomplish all the requirements satisfactorily (Windows Server 2012 R2, 2016, and 2019) and each of these years have three editions (Windows Server Essentials, Standard, and Datacenter). In order to clear up some possible confusion, there are three different terms: “years”, “versions”, and “editions”.

Every few years Microsoft releases either an update (denoted by an “R2” next to the year) to an existing year of Windows Server, or a completely new Windows Server, I have mostly seen these referred to as the “years” of Windows Server. The “versions” are used to refer to the different OS types within Windows Server (ie. core, nano, etc.). Finally there are the “editions” of Windows Server, these are how Microsoft denotes the different capabilities and features of a year of Windows Server (ie. essentials, standard, etc.).

Unless you are using legacy software that requires older generations of Windows Server, there is not much of a reason to use an older year of Windows Server. Even if you are, Windows Server 2019 is able to run at what is called a "functional level" of Windows Server all the way back to 2004. Windows Server 2008 and 2008 R2 unfortunately have reached the end of mainstream support and become serious security liabilities. Similarly, in about two years[10], Windows Server 2012 and 2012 R2 will reach the end of mainstream support and start introducing vulnerabilities as they lack patches for holes in security.

The reason Windows Server 2012 R2 is considered a viable operating system for this project but Windows Server 2012 is not, is because while there is little difference in the interface, was a were significant overhaul in Hyper-V (significant to this project) as well as improvements to Storage Spaces and Active Directory (not significant to this project[11]). The two final options, for the years of Windows Server, are Windows Server 2016 and Windows Server 2019. There are several updates and features that were both added to and removed from Windows Server 2016, as well as security changes and updates to create Windows Server 2019.

The main feature changes (that affect this project) are that Windows Server 2019 now offers shielded virtual machines in Hyper-V [12], VMConnect , Windows Defender Advanced Threat Protection, and integrated network security upgrades[13]. All in all, these security benefits, along with the ability to run shielded VMs[14], makes running Windows Server 2019 the best option.

The next decision is choosing which of the three editions of Windows Server 2019 is best suited for this project. There are three basic editions of Windows Server, and they are Windows Server: Essentials, Standard, and Datacenter Edition. In addition there are Windows Server Core and Nano Windows Server. Server Core and Nano Server are both headless options of Windows Server (no mouse, monitor, or keyboard), and both have many benefits over their desktop counterparts[15].

Windows Server Core[16] is able to host Hyper-V VMs, run infrastructure workloads (including file servers, domain controllers, AD DS Servers, and DNS servers), and most roles that can be run by its desktop countertops. Windows Server Core is also considered more secure than its desktop counterparts because it takes more technical skill to navigate and lacks vulnerabilities affiliated with the Graphical User Interface (GUI) and the extra features installed to replicate a more windows-like environment. In addition to this, the fewer number of vulnerabilities means that security patches are smaller[17] and there are fewer of them that are necessary, which results in less downtime for the server.

Windows Nano Server, however is a more complicated system. It was originally promoted as a successor to Server Core, claiming that it was a smaller, more secure, more efficient headless version of windows server that could still run all the roles of Windows Server Core, handle all the infrastructure workloads, and run Hyper-V VMs. In fact, it was designed to be the ideal environment for running IIS Web Server, hosting containers, or Hyper-V VMs. Unfortunately, it was later given a complete overhaul, removing all infrastructure capabilities

(including IIS and Hyper-V) and Microsoft suggested that Windows Server Core should be used for these roles instead.

These changes to Windows Nano Server were explained in a blog post by Microsoft titled, “Delivering continuous innovation with Windows Server" saying, “Based on that feedback, we are making an important change to Nano Server. This next release will focus on making Nano Server the very best container image possible. From these changes, customers will now see the Nano Server images shrink in size by more than 50 percent, further decreasing startup times and improving container density.” Despite these changes, if you go through Microsoft’s documentation you can still find a large amount of outdated information pertaining to the original purpose and deployment of Windows Nano Server.

There is one other dedicated headless Windows Server option, and that is Windows 10 IoT Core. Windows 10 IoT Core uses an amalgamation of the Windows 10 common architecture and the Windows Server Core architecture. While both Windows 10 and Windows Server 2019 share both a kernel and many of the same base programs, they have a different architecture, and this has caused many people to consider Windows 10 IoT Core a lightweight build of Windows 10 rather than a headless server despite Microsoft including it in their server documentation.

While the ideal operating system for this project would have been Windows Server Core, the the requirement to manage it through another Windows PC or solely through the command line ultimately eliminated it. This left the three editions of Windows Server.

Windows Server 2019 Essentials Edition was designed specifically for small businesses, however it was the most easily eliminated because it does not allow any Windows Server Containers with Hyper-V and does not have functionality for Windows Server Containers without Hyper-V isolation.

The more involved choice was between the two final options, which are also the most similar of the choices[18]. Windows Server 2019 Standard Edition and Datacenter Edition have many things in common, they have the same number of possible Internet Authentication Service (IAS) connections, Routing and Remote Access Service (RRAS) connections, Server Message Block (SMB) connections, and Remote Desktop Services (RDS) connections. However none of these were particularly significant to this project. The differences that could be significant to the project are:

- Windows Server 2019 Standard Edition can only run two Guest VMs, while Windows Server 2019 Datacenter Edition can run a theoretically unlimited number;
- Windows Server 2019 Standard Edition can only run two shielded VMs (depending on CALs licensing), while Windows Server 2019 Datacenter Edition can run a theoretically unlimited number;
- Windows Server 2019 Standard Edition can not operate software-defined networking, while Windows Server 2019 Datacenter Edition can;

- Windows Server 2019 Standard Edition can only operate as guest (for hearted activation) if hosted on  Windows Server 2019 Datacenter Edition, while Windows Server 2019 Datacenter Edition can operate as either a host or a guest.

While these differences clearly indicate that Windows Server 2019 Datacenter Edition is the more feature-rich of the two, there isn't necessarily a "better" operating system, only an operating system better suited for a particular role. Due to this project operating on a limited scale, the larger package size and more feature-loaded Windows Server 2019 Datacenter Edition operating system is not needed. While I don't have to worry about licensing costs, the faster restart and load times on Windows Server 2019 Standard Edition along with its smaller package size is why I ultimately decided to use it for this project.

# Setting up the Hardware

Once the operating system was chosen, the next thing to establish is the hardware that the operating system will run on, and to set up that hardware. Something to note is that while working on this report and writing up all the information on the server after the project was 'done' I used PowerShell to view the system information and had it export the print out to a txt file which is included at the end of the endnotes of this report.

The process to reach what hardware the server would run on is slightly less thorough than the decision in what operating system the server itself will be, because it was mainly dependent on two factors: cost and availability. The server had to be relatively cheap[19] and be available to purchase immediately. After doing some searching, I found a seller on eBay, who had relatively good prices and good customer feedback. This is where I ended up purchasing a Dell PowerEdge R710.

The Dell PowerEdge R710 is a 2U rack-mountable server, with my particular configuration having the following I/O:

- 4 gigabit ethernet jacks
- 6 3.5" SAS/SATA drive bays
- 1 SATA II DVD Rom slot
- 4 USB 2 Type A ports
- 2 VGA ports

- 1 DVI-D port
- 1 IDRAC port (with 1 SDXC card slot for firmware updates)

The PERC6/i[20] Serial Attached SCSI (SAS) Redundant Array of Independent Disks (RAID) Controller is connected to the server through a PCIe x8 slot. Connected to the server through the PERC6/i are two Serial Advanced Technology Attachment (SATA) Hard Disk Drives (HDDs).

Drive 0 (PD0) in the RAID array is a Seagate BarraCuda Compute (ST2000DM008) internal 3.5” HDD with a capacity of 2TB, a spin-rate of 5400 RPM, a cache of 128MB[21], and an overall transfer rate of 6GB/s.

Drive 1 (PD1) in the RAID array is a Seagate BarraCuda 7200.12 (ST31000528AS) internal 3.5” HDD with a capacity of 1TB, a spin-rate of 7200 Revolutions per Minute (RPM), a cache of 32MB, and an overall transfer rate of 3.0Gb/s.

There are two major downsides to both of these drives: they are both SATA drives, and they are both SMR Drives. SAS drives are faster, more reliable, more efficient, and designed for reading and writing data during continuous computer use. In fact, they were designed for server use. I ended up going with SATA drives, however, because I already had them and they worked (or so I thought). I did purchase a relatively cheap (used) 1 TB SAS HDD and it was advertised as fully functional, however when I received it in the mail, it was completely dysfunctional, so I continued with the two SATA drives I already had.

SMR, or Shingled Magnetic Recording, drives has one benefit: storage density; despite this, CMR, or Conventional Magnetic Recording, drives are significantly preferable in this scenario.

SMR works by writing data partially overlapping other data, and operates under the knowledge that the write head is larger and less precise than the read head. This results that if the write head over writes half of the imprint of each bit written, the read head is delicate enough to still determine each bit. While this theoretically leaves all the data readable, if the writing operation does not go ‘optimally’ the corrupted lanes of data may have to be rewritten again and again until it has been successfully recorded. This process of having to rewrite some lanes of data mean that the transfer rates decrease significantly. Data caching is meant to offset this.

There are usually two types of flash on SMR drives (not including the standard controller cache): on-disk disk temporary cache and on-disk long term cache. CMR drives use a technology called Perpendicular Magnetic Recording (PMR) to write their data onto the disks, and this same PMR technology is what is used for the on-disk cache on SMR drives. While not any faster than a standard CMR drive, this transfer rate is significantly faster than writing directly to an SMR drive without flash.

The SMR drive writes to the on-disk temporary cache, and the controller immediately starts writing that to the SMR permanent storage. These on disk caches, however, are rather small on the outermost tracks of the disks, and if the cache is filled before the controller can transfer the data to the inner SMR tracks, the cache must be overwritten or emptied[22] before new data can be written to the SMR permanent storage.

In addition to overwrite corruption issues, there is another aspect to the SMR system that slows down the write speeds. In the SMR system, data can only be written sequentially, so they partially overlap each other — just like their namesake: roofing shingles. However, this means that to change any bit (a 1 or a 0 — a single bit of binary data) in the middle of a track, the entire track has to be rewritten in sequential order.

To prevent having to rewrite the entire disk, the SMR system breaks its storage into blocks called, “zones”. This reduces the size of the rewritten data to the size of a single zone (usually 256MiB, or 268.435 MB, but this can differ depending on the drive). Whenever a bit needs to be updated in any of these zones, the entire zone has to be read, written into the controller’s cache (different than the on-drive cache — explained in the endnotes), and then updated in the controllers cache, before it can then be rewritten into the zone.

The same operation in a CMR system would be to overwrite the bit that needs to be updated with the correct bit. It takes that single step and makes it four, in fact, it is even more steps than that, because it neglects the multiple extra movements of the reading head and the

writing head. This is called “Write Amplification” and flash memory suffers from the same inefficiency (except for Single-Level Cell, or SLC, flash).

SMR drives should never be used in a RAID array[23]. RAID arrays are a high-random-access-and-update application, meaning the response time of a SMR drive will be pretty much the least optimal scenario. SMR systems shine in one scenario for consumers or stand alone servers — and for that scenario I am using a CMR drive (this will be explained later) — and that is backups. Being completely sequential, not being particularly time sensitive, and not part of a RAID array, a SMR drive offers you a more efficient, cheaper, and more storage dense drive that is ideal for that application.

Based on all of this, I should clearly have used the SMR drives for the backup drive of the server and used other CMR drives for the raid array that the server runs on. And while I own four 1TB CMR HDDs and four SSDs (however, one is NVMe so is inapplicable), all seven are in a 2.5” form factor and, while there are configurations of the Dell PowerEdge R710 that have eight 2.5” drive bays, my server configuration was six 3.5” hard drives rendering them useless. I was able to use one of these CMR HDDs by using a USB-to-SATA connector to connect a 2.5” HDD to the rear of the server and use it as the off-server backup drive. I had to adapt and work within the limitations of the server that I was able to get within the timeframe, and unfortunately, the only servers with 2.5” drive bays were either the wrong type of server, or were much more expensive.

The server is powered by two 870 Watt hot-swappable switch-mode power supplies. Both of these power supplies are plugged into a surge protector rated to 2,880 Joules, shared only by the monitor that is used by the server.

The server is configured with 64GB (8x8) of 2Rx4 rated Hynix ECC[24] 72-bit Buffered DDR3[25] SDRAM[26] RDIMMs[27] (HMT31GR7BFR4C-H9 D7 AB) with a base clock speed of 1333MHZ (at 1.35V) and a memory size of 8GB. Each Memory module has a heat spreader and thermal pads (in the place of thermal paste) on each memory cell. In addition to this, the air channelling insert diverts some of the airflow across the heat spreader on the RDIMMs.

The RDIMMs are configured to run at 1.35v with DDR3L (low voltage operation of the RDIMMs). While the memory modules support both 1.5v and 1.35v operation, it is not recommended to use higher operation voltages for increasing the clock speed above its specified value. However, I had no need to overclock the RDIMM modules, so I left them at the recommended voltage (1.35v) and at their base clock speed (1333MHZ / PC3 10600).

The server is configured with two Intel Xeon X5550 Nehalem processors (BX80602X5550), and Intel no longer has the data sheet for the X5550 available to the public, however going off its spec-sheet and the bios of the server this is what I have determined about the CPUs. It has a server vertical segment, meaning it is designed and tailored for a server use case, including ECC support, dual socket support, and additional management features.

The Intel Xeon X5550 Nehalem processors were built on the outdated 45nm architecture[28], which results in it being less efficient, have a higher Thermal Design Power (TDP), and have fewer cores in the same die-size. This outdated architecture gives this relatively under powered CPU (compared to other server CPUs) a TDP of 95 watts. The CPU is then seated in a LGA 1366 socket and (after the application of thermal compound)[29] a passive cooler was tensioned over the CPU holding it in place and making firm contact between the IHS and heat base.

The cooler that was used has both an aluminum heat base and aluminum heat fins. Each of the two coolers themselves are completely passive with sixty two heat fins, but three of the five server fans are directed across the coolers with a plastic air channel. Each of the fans (Sanyo San Ace 60 — 9gv0612p1m041) has a flow rate of 69CFM (cubic feet per minute) however, I was unable to get an accurate number for the max RPM of the fans.

When the server first arrived, It booted, however it was not reading the RAID controller, the IDRAC, or the front IO (including the DVD ROM). I was able to resolve the issue with the DVD ROM and front IO by simply reseating the connectors, and it is my assumption that they must have been loosened during shipping. The IDRAC and RAID controller, however, were a more persistent problem, and after reseating the RAID PCIE card and the IDRAC, I contacted the seller.

For the IDRAC, I was able to conclude that something must be wrong with it and the seller shipped a replacement at no cost. The RAID controller, however was a continuing issue. In a last ditch effort before either purchasing a new RAID controller (seller would not provide replacement because they said it passed all their tests before they sent it out) or returning the server and purchasing a new one, I completely disassembled the RAID system on the server. That included the PCIE riser card, the RAID controller card, the channel cabling, the front splitter, the front drive panel, and even the backup battery for the RAID controller card.

After using a can of air to clean every connection, I reassembled the server and booted into Dell Unified Server Configurator (Dell USC) (separate from the bios, however being able to do most of the roles of the BIOS and including several other features[30]) and ran a diagnostic on the RAID controller. It detected the controller — which it wasn't before — but found no "connected drives". This concerned me at first, however after looking into it, I learned that the Dell USC can only detect the virtual drives of the RAID controller and not the physical drives themselves.

The next step I took was to reboot the server and enter the RAID controller configuration menu. Rather than getting an error message, I was presented with the menu saying there were two 'foreign' physical drives that were unusable. After working with it (In true programmer fashion[31], I spent hours working with trial and error and exploring the problem myself rather than spending 20 minutes reading documentation) I was able to figure out how everything works and how to create the needed virtual drive.

From here I was able to configure a RAID 0 array virtual disk (called VD0) and initialized this drive. The first time I did this I did something called a deep wipe and total initialization which writes over the entire drive turning every bit to 0 and then initializes it in the array. However, because both drives were SMR drives, this process took nearly nine hours, despite it being a solely sequential write, making it the ideal write scenario for SMR drives (other than the write being greater than the CMR cache).

# Installing the Operating System

Now that I had chosen an operating system to run, as well as having acquired and configured the hardware it would run on, the next step was to actually get the operating system itself. A new copy of Windows Server 2019 Standard edition is approximately $450 dollars[32], however I had no intention of adding that to the cost of this project.

That is where Microsoft Azure comes in. I created my Azure account nearly three years ago, and have been using a student license ever since, and while my free cloud computing credits ran out a long time ago, I still have access to many of the azure features, including all their SaaS[33] programs, learning tools, app-services tools, and most importantly for this: their software.

Microsoft Azure — with the student license — gives you access to many of Microsoft's softwares and operating systems without their hefty price tag. Microsoft lists Windows Server 2019 Datacenter Edition at $6,155 (even though it usually retails at significantly less depending on core count[34]) Windows Server Datacenter Edition is included with the Azure student License, along with Microsoft Machine Learning Server, Microsoft R Server, Microsoft Hyper-V Server 2019, and many others.

Having access to all of these has been useful for many things other than this project, however I had never used any of the server licenses before — all my server experiences were

with linux servers — and so I had the license key and download link for my operating system of choice: Windows Server 2019 Standard Edition.

I don't work with Windows that much, my main computer is a Mac and the only other computer that I use with any frequency is currently running Arch linux. The tool that I usually use to make installer (or live) USBs is Balena Etcher, and I honestly forgot that you cannot use that tool to make anything with windows (live boot, recovery, install, etc.). So I downloaded Windows Server to my Mac and tried to flash it to a USB.

It didn't take me that long to realize my mistake, and I also realized that I didn't have any computer that is bootable to windows. For this reason, I had to borrow my sister's computer, download a Windows 10 ISO and the program Rufus (a tool for flashing Windows USBs), and create a Windows installer drive.

Unfortunately still, the only computer that I could put Windows on, my Lenovo Thinkpad x250, is somewhat broken. I purchased it a year and a half ago on eBay for a relatively low price and it occasionally crashes and applying pressure on its left hand side often causes the computer to have what I can best describe as a seizure[35]. For this reason I can only expect that there is something wrong with the motherboard, or the connection to the CPU (explained in the footnote above), however I chose to ignore this fact and continue on with the project.

Now that I had a computer running Windows 10, I copied my Windows Server ISO over from my Mac to the x250, and installed Rufus to the x250. From here it was a pretty straight forward process to format an unused thumb drive and flash Windows Server to it. It should be noted that every time I mention: downloading Windows Server, the Windows Server ISO, and flashing Windows Server to a thumb drive — that I am in fact referring to the Windows Server Installer.

From here, I plugged the thumb drive into the rear of the server. After going though the installation wizard (its a pretty straight forward process and I will not going through selecting the location of install, etc.) I created the admin account and password. For the admin user password, I used a randomly generated set of 20 numbers, letters (including capitals), and symbols. While the other passwords are passphrases based off of the leading letter for historical documents that I have memorized (with numbers substituting words and other adjustments), I know that I would not have any trouble memorizing a random set of numbers, letters and symbols.

I chose a standard password length for all passwords in this project of 20 characters, I chose this length because I knew that I would have no difficulty memorizing a couple passwords of this length, even given that two of them would be completely random. The only other completely random password was that of the Microsoft account that was used to register the Hyper-V VMs.

Once I had created the admin account and the system finished installing and initializing, I logged in and was greeted by Server Manager. From here I had to make the decision of how to manage the server, either with a keyboard and monitor or in a headless (remote GUI through management tools) configuration.

Ultimately, I ended up not going with a headless operation of the server, even though it was my original plan, due to the issues with the x250. My original plan was to keep the server in the basement, this had three main benefits: it is colder (which reduces the cost of running the server), it is out of the way (reducing footprint in both a physical and an auditory form), and it has a direct ethernet line from the router.

Due to the x250s issues, I turned to using my Mac to manage the server remotely. Windows Server can be managed remotely through four different approaches: Remote Desktop Protocol (RDP), Windows Admin Center (WAC), Remote Server Administration Tools (RSAT), and Secure Shell (SSH).

RDP is a way to remotely log into a windows machine and control the machine similarly to how you would control a virtual machine: using your own mouse and keyboard. However, RDP is a legacy program and is plagued by many security vulnerabilities. For this reason, I disabled RDP on the server.

WAC is a browser-based management program for managing both Windows Servers and Windows PCs. WAC, until recently[36], lacked the ability to manage Hyper-V and IIS; while it is now officially supported, it lacks many of the configuration settings and is sluggish and more difficult to do. It has the benefit of displaying performance metrics in an easy to use dashboard and WAC offers the ability to manage several roles as well as server functions from a single interface. Microsoft makes it clear that WAC should not be considered a replacement as a management service but that, rather, it should be used to complement RSAT.

RSAT is a collection of GUI and PowerShell tools used to manage all the roles and features of Windows Server as well as management of the server itself and even entire server clusters. RSAT has many of the capabilities that WAC lacks, however this will not continue and is already starting to shift. With the last two releases of Windows Server 2019, there are several roles and features that can only be managed with a GUI though WAC (they can still be managed through PowerShell in RSAT, however) and in the future Microsoft says that there will be GUI only roles and features that can only be managed through WAC.

SSH is the most universal remote management tool, and one that I have used to manage Raspberry Pis, home NAS servers, and my own Mac. There is something called “PowerShell Remoting” which is another way to manage Windows Server, however I did not include it as its own item because it uses SSH to establish a remote PowerShell session, and so I am including it in SSH. When you SSH into Windows Server, the main tool that you can use is Command Shell

(which is why PowerShell Remoting was developed), however there are ways to use other shells[37] to control the server.

From these options, the most preferred management tools would be RSAT with WAC, however these are not available to be used from a Mac, and therefore I was left with two options: RDP or SSH. Because I had disabled RDP for its many security vulnerabilities, I had to use SSH or not manage the server remotely. Faced with this choice, I chose to use an old desk as a temporary server rack next to my desk and use a monitor and keyboard.

Once I had decided to use Server Manager, I connected the server to ethernet (three gigabit connections to the router), and found an old VGA-compatible monitor and USB keyboard and mouse. Like most servers, the Dell PowerEdge R710 lacks both Bluetooth and Wifi natively, which eliminated the ability to use most of my peripherals, lacking HDMI or DisplayPort[38], I also had to put an old monitor on the side of my desk as well.

This pretty much concludes the section regarding installing the operating system, I would like to make one final note that I will not go over the reinstallation of the operating system between each attempt at setting up the server, and the only other time I will talk about installing the operating system is when I am covering the step-by-step process I took when creating the penultimate server.

# Setting up Windows Server with Udemy

My first approach was, naturally, to just start trying things having no idea what anything was or how it worked. I believed that my experience using other programs along with my intuition would be enough for me to get along. This proved to be far from reality.

This successive approximation style of approach came to an ignominious end when I realized I didn't even know that there were separate managers for the roles, that the roles wouldn't run unless you installed the necessary features, and that once you install features, you can't change things such as host name and server domain without a significant effort going through settings-trees on all the roles.

Faced with this new conclusion that I would need to make a serious research effort, I reinitialized the RAID array and reinstalled Windows Server[39]. However, I also missed one more aspect of this: you have to de-link the license key from the copy of Windows Server before you wipe it. By not de-linking the key, I removed my ability to activate Windows Server 2019 Standard Edition. [40] I still had the installer drive, meaning I was able to install Windows Server, select "I don't have a product key" and use it inactivated. This removed some features, and can only run for 90 days, however neither of these restrictions were particularly significant to the project.

Following these two failures, I took another approach. While you learn a lot from your failures, and I certainly did, I decided I needed to learn a lot through thorough research first… and then learn more through my inevitable failures to come. My natural trial-and-error approach to most problems is seen all throughout this project, and most other projects that I do.

For the research I started going through YouTube videos, and after probably seven-ish[41] hours worth of YouTube videos of the initial setup of Windows Server, I started going to Reddit, Stack Overflow[42], and other online forums searching for articles and posts regarding the same thing. By my estimates I spent over a week researching just the first step, this was partly due to the very different results and most YouTube tutorials were incomplete and many of the community posts had people saying that the presented steps were flawed. Ultimately I realized I would continue going around in circles if I continued down this road.

One of the YouTube tutorials that I particularly liked was the beginning of a Udemy course that was released for free so that people, just like myself, would watch it and come to the conclusion that getting the course would be the best approach. So I went along and got the Udemy course and started following it. There were several problems however, with three being the most severe.

First, the instructor did not explain why a certain set of actions should be taken, rather he would say something along the lines of ‘and then you must do the following’ and so I did the following. This led me to setting up a DHCP server role, an AD DS role, a DNS server role, and

more. None of these are actually useful for the project, and I didn't understand why the instructor said they were so necessary.

Second, just like nearly all the YouTube tutorials, the instructor had Windows Server installed in a VM purely as an educational and testing environment, meaning there was no work with network settings, routing, or accessing the server from any other device. Furthermore, he neglected to cover the details of an actual physical server, such as backups, which would be very problematic later.

Finally, while I did learn to use the interfaces and managers and where everything was, what he was teaching had little overlap with the project. The Udemy course was mainly focused on teaching someone their way around Windows Server in order to manage a network of devices for a small business, however because it was a virtual machine test environment , all the network management material was just mentioned theoretically and was ultimately not particularly informative.

The course started with setting up VirtualBox and setting up the sandbox testing environment. I skipped past all of these at first — I went back and watched them later to see if there was anything about networking there… but there wasn't — and began with the next section on navigating Windows Server.

After going though this, I did what the instructor recommended and followed along with my own install of Windows Server, following his same steps and only changing the values. The first step was changing the name of the server and workgroup to be something more practical — I used WORKGROUP and GarrettGRServer1 — and correcting the timezone.

The next step was to disable automatic updates, however he did not check and install the current security updates and while having manual control of updates can be an improvement to security (for a diligent and proactive security team), installing the most recent security patches is just as important if not more-so.

From here I, following along with the instructor, installed the Active Directory Domain Services (AD DS) role, and configured it. While I learned after this that AD DS has nothing to do with IIS or anything else related to this project, at this point I spent about 22 hours following along with the Udemy course pretty much exactly.

Next I set up DHCP, and then DNS, and then skipped Print Services. Even though I was following along with this course, I knew that I would not need to configure Print Services — while we do have several network enabled printers we utilize direct-IP printing[43] — and ignored this part of the course.

Next he covered Windows Server Update Services (WSUS) and an introduction to PowerShell. Both of these were quite useful to the project, however this was the exception. After

that he covered File Server Resource Manager (FSRM) and NTFS permissions. He then covered Organizational Unit Control (OUS).

By this point I had realized that the course was not going to be particularly helpful, the last two things he covered were Name Resolution Services (NRS) and IIS. He did not have any code for a website nor did he ever create a proper hosted website, however he did a decent job covering how to navigate IIS and the only thing that I had to research about using IIS after this course was SSL certificates which was apparently beyond the scope of that course.

While I gained a significant amount of vitally important knowledge on how to navigate and interface with Windows Server, I also knew that I would have to go somewhere else to learn more about how to accomplish what I had to do. For this purpose I turned back to what got me through my early education in programming: YouTube tutorials and Google.

# More YouTube Tutorials

At this point I knew that I couldn't just follow along step for step with someone else to accomplish this project. In response to this realization, I spent a lot of time watching 76 different tutorials (yes I seriously looked up 'windows server' in my YouTube watch history and counted). Unfortunately, all but two of these were done in VM testing environments.

There were four channels: "NLB Solutions", "Instructor Alton", "Microsoft Azure", and "IT/Ops Talk" that I ultimately found to be useful. NLB Solutions had the best explanation for the initial configuration of Windows Server, and Microsoft Azure had the best tutorial for best practices to implement after the initial configuration. From there, another channel "H-EDUCATE" and Instructor Alton were the most useful for working with IIS, and IT/Ops Talk had a very useful video covering installing and configuring OpenSSH on Windows Server.

The research that I completed during this section of the project was done with very little implementation of it on the server itself. For the most part, I was simply watching the tutorials and memorizing the different processes. Ultimately, I was taking the approach of becoming knowledgable in the process in its entirety, learning what each role did, how they interacted, and which roles and features I would actually have to configure (and how to configure them). Furthermore, I was going and researching everything that was mentioned, rather than accepting

what anyone said at face value. I would research it to see if it was something I actually needed, what it did, how it worked, and how to use it.

For this, the official Microsoft Documentation and the educational resources from Microsoft Azure were the most useful. While I occasionally ran across outdated information, most of the time it was very useful, if not overly detailed at times. For much of this project, I had hundreds of tabs open and was often opening more at a rate faster than I was closing them.

As for hardening the server, after my initial experiences with setting up the server, I had concluded that the best approach was to first get the server operating at full capacity and then harden the server. This did not mean that I would do no hardening of the server while configuring it, I simply did not make it my utmost priority while I was first setting up the server. Many of the later configuration steps that I took were for security, and after the server was configured I took many more steps in hardening the server, however that is later in the project.

This process of researching and amalgamating tutorials did not go as smoothly as I hoped either, as there was a lot of conflicting information. So I was having to combine parts of different YouTube tutorials, and they often had differences in their base configurations making them not completely compatible with one another without adjustments, which I had to figure out myself. Figuring out and fine-tuning these adjustments accounted for two more complete rebuilds of the server. The biggest difficulty was the networking. Due to all of the tutorials and videos having Windows Server installed on VMs, they lacked most of the actual network resolution details.

Some of them simulated this by using virtual networks with multiple VMs, however this can only go so far in being a facsimile for the real thing.

I was able to figure several of the initial settings without much difficulty. I was able to find my “default gateway” (my home router’s IP address) by going into the advanced network settings on my Mac. I also knew that my router had dns resolution so I used that for the dns on the server. Something that I didn’t know at first but soon figured out was that there had to be separate ethernet connections for the virtual switch and the web server.

The virtual switch requires a dedicated NIT, which means that NIT can’t be used for either connecting to the internet or hosting a web server. When I was still working in my room, I had run up a single ethernet cable through the floor (from the router up through the ceiling of the office with the router into my bedroom). In order to get the necessary connections for both the virtual switch and web server, I used an old ethernet switch (unfortunately this limited my speed from over a gigabit to less than 100mb/s).

At the time, I was unaware of port forwarding and my general knowledge on networking as a whole was lacking. It wasn’t until the penultimate configuration of the server that I reached out to someone much more knowledgeable than myself for assistance that I learned about what I needed to implement to resolve most of my networking issues.

# Creating a Website

After spending several weeks researching the project with YouTube, Microsoft documentation and just about every resource I could think of, I ended up taking a break to go back to my roots and sit down to do some coding. I have coded several websites in the past, and currently have three that are running at some capacity. I am working on starting a side photography business, so I decided to create a photography website.

The only drag-and-drop-esque editor I have used in the past is WebFlow and my first thought was to make a very quick website, or even use my current WebFlow website, and pay to export the code for the website and host that on the server. However, I decided that I wanted to work on my coding skills and also do something that I, quite frankly, enjoy more than IT system management and code the website.

My sister goes to the Fashion Institute of Technology for graphics design, and I have worked with her to design and develop websites in the past, so it is something that I am both familiar with and something that I already have the tools to do.

In order to get a base for the website, I started a new project in a program called Bootstrap Studio. Because I have the Github Student Developer pack, I have a license for the full version of Bootstrap Studio, and I have used it to both create and host websites in the past. When

I hosted a website on Bootstrap Studio Sites I used a custom domain that I got free from Github (with the student pack) from name.com and I unfortunately returned to them for this project, however why that's unfortunate will be explained in the "Intranet vs Internet" section.

One of the unique things about Bootstrap Studio — and the main reason why I use it as a starting point for most of my website projects — is that you can use their large libraries of element templates, page templates, animations, icons, and more (even for commercial use). Furthermore, you can use the editor or edit the HTML and other files directly, or do both at the same time. My standard approach is to make a general outline (where I want the navbar, how many pages I want there to be, etc.) and then export the code and open it in my preferred editor: CodeRunner 3, and open the index.html file in safari.

There are also third-party fonts and icons available, I personally prefer to use "boxicon" icons and google fonts rather than the fonts and icons that are included in Bootstrap Studio. When creating the code base in the editor, I created the main page to have six sections and created another page as a template for what the details page on one of the portfolio items would look like. The last thing I did in the editor before exporting the project and editing it purely in my IDE was to import animations for the different elements (mainly scrolling and data visualization animations).

Once I opened the file in my IDE, I replaced the filler images that were included when I imported the elements, and put in my name and information. While it would have been quicker

to put in the appropriate text, images, icons, links, etc. in the editor, I prefer to work directly with the HTML. After inputting the website's title, I had to create the header, which included links to my different social medias.

For this, I opened up the boxicons.css file and searched for GitHub, LinkedIn, Instagram, and the rest and pasted in the correct `href` (hypertext reference) for the icons, along with the correct links and I included code to ensure the links would open in a new tab and another snippet of code to the anchor element that I learned from listening in on one of my sisters classes: `rel="noopener noreferrer"`. This ensures that the the new page runs as a new process and prevents it from accessing the `window.opener` property as well as preventing the `Refer` header from being sent to the new page.

Next for the navbar, I made an anchor element element (with a scroll-to) to each section. While I initially created six sections, one section (the resume section) was commented out of my code because I did not ever end up creating an up-to-date resume to put in there. This left me with five sections and five elements in the navbar. In addition to the name of the section of the website that each element will scroll to, I also added a boxicon icon (there were the default icons from when I imported the navbar template but I replaced them with the icons I wished to use).

For the Hero section of the website (called "home" in the navbar), all I had to do was replace the filler text with my name and the data-typed filler text with what I wanted it to scroll through as well as the word "photography" at the end to create an interesting title for the landing

page of my website. Having a title with moving elements, especially something that carousels or scrolls though several elements, is very important for drawing people's attention and making the website more memorable (I've listened in on more than one of her classes… just not the art history ones).

For the About Me section of the website, I wrote two passages and filled in filler text, however the only adjustments I had to make to the code itself was putting in the new path to the photograph for it to display For the next section, the Facts section — in-between the About Me and Skills sections (with no link in the navbar) — I wrote two more sections, filled in filler text (didn't update the icons because I had spent too long on the website by now and needed to get back to the project), and adjusted the starting points for the data counters so that as they faded into view they would be around zero rather than having already counted up to their maximum value[44].

I made no changes to the Skills section (in-between the Facts and Portfolio Section also without a link on the navbar) other than replacing filler text and filler values. After this, between the Skills and Portfolio sections, is where the Resume section was originally going to be, however I commented out all the code for the Resume section.

For the Portfolio section, I wrote a small blurb at the top, created tags for all the images in my portfolio, filter buttons along the top to filter by the tags, entered the correct path for the

images being displayed, and wrote a “coming soon” disclaimer next to the link to take the view to the page for more details about the portfolio elements.

Similar to the Facts Section, I did not replace any of the icons in the Services section due to the fact that the website had added almost another three weeks to the project at this point. I did, however, replace the filler text. In-between the Services and Contact section was supposed to be another section, titled the Testimonial section. However, for the same reason as the resume section, I didn’t have time to ask any of my past photography clients for testimonials and chose to comment it out.

The final section of the website is the Contact section, where I added an embedded Google Maps frame that I found on Stack Overflow with the intention of having it point to the municipal building in my area or something similar to that, as photography can often require you to know the general location of the photographer in order to plan a photoshoot. I was unable to easily change the location the frame pointed to and did not wish to spend the time to trouble shoot the issue and simply commented it out. In Bootstrap Studio I also added a php email contact form, however I never set this up as it was beyond the scope of the project, I had only a little experience working with php, and I had to re-focus myself on the main purposes of the project.

I updated the navbars and heroes of the other pages (Portfolio-Details-Template and Inner-Page-Template pages), however I did not do any further work on the other pages, because

there are many fully functional single-page websites and if I choose to use this website as the website I continue using in the future I will simply make those edits then. After I ‘finished’ working on the website, I took all the html files, assets, and everything else, put them in one folder, and used a thumb drive to transfer it over to the server.[4]

# Setting up the Penultimate Windows Server

Now I had to actually set up the server. The ultimate goal was for this to be the very last time the server had to be configured, and because of that, I am going to go through nearly all of the steps taken in setting it up. While I did have a system failure and had to reinstall and reconfigure the server, I repeated nearly exactly the same steps, including using the same static IP addresses, VM names, and everything. I have mentioned some of the steps that I took previously, and so there are some parts of this section that are repetitious, however I included them here regardless for the sake of comprehensiveness.

There are three subsections that make up this section: installing and configuring Windows Server, configuring Hyper-V, and configuring IIS. However they are not wholly separate, as many of the steps taken in configuring Windows Server were prerequisite configuration steps in order to be able to configure both Hyper-V and IIS. In addition, some of the configuration goes into the hardening of the server, however the details for that will be in the “One Last Time” section.

For installing Windows Server, the first step is to create installation media. This is done by gaining access to a Windows Server ISO, putting in a thumb drive (or whatever you intend to use for the installation media), and using a program like Rufus (the program I use) to flash that

installation ISO to the aforementioned media. From there, insert the installation media into the server, and boot to the installation media.

Once the server has booted from the installation media, select “install” on the screen that comes up, hit “continue”, and then enter the product key. I got my product key from Microsoft Azure with the Student License as previously mentioned. Next, accept the license terms and on the next page select “advanced: custom install”. From here, all that must be done to complete the installation of Windows Server is to select the install location (the RAID array virtual disk), wait for Windows Server to install, and then create the administrator account.

I made the administrator account password a series of twenty randomly generated characters (letters, numbers, and symbols). The only other password with this higher security is the password for the Microsoft account used to set up the two Windows 10 VMs. The other passwords were also of the same twenty character length, however they were passphrases, with a much lower quantity of symbols and numbers.

The next step is to begin the configuration of Windows Server. The first configuration option you are presented with is if you wish the device to be visible to devices on the network, I set this to “false”. While on a business network there could be some advantages to this, this would really only be suited for de-centralized networks without a server, and creates several security vulnerabilities. From here, there is not a fixed order of the steps, however this is the order of the steps that I took.

First I set the server name (and description) and workgroup to “GarrettGRServer1” and “WORKGROUP”, respectively. I chose to do this first because once roles and features are downloaded, changing these, while possible, is very difficult and requires you to go into the configuration settings for every role and feature and adjusting each one independently. Next, I set the timezone to the correct timezone. While this may not seem important, it is significant in computer security.

While, admittedly, this is more important for servers of businesses, in the spirit of the project, this is something I made sure was corrected. This is also most significant for web servers, like the server I was configuring. If the time is off, even just by a few minutes, the log files will not be accurate, but there is much more to it than just log files. If its at a company, you can use the time accessed for a file and correlate it with security footage or another way to determine if the user (possible suspect of an inside-job) was at their computer. However, even if the attack was remote, the attacker may use a dynamic IP, and if the system time is off (even by just a few minutes), it is significantly more difficult (if possible at all) to determine who the attacker was or even where it came from.

Furthermore, the times won’t just be logged wrong on log files, but the “MAC times”[46] will also be incorrect (the footnotes regarding MAC times should be checked out, as they are very interesting). Having accurate MAC times is critical for incident response. Many forensic analysis tools read the MAC times and sort all the files into a timeline to figure out all the files

an attacker may have accessed or modified. Other than security, having the wrong system time can affect the server's logistics[47], whether that is unpredictable downtime, interruptions to clean temporary files, backups, or rebuilding system databases.

Going back to the steps in configuring Windows Server, the next step was to install all the available updates. After they had been installed and the server had been restarted, I changed the update mode from fully automatic to manual. You cannot set the update mode to manual using the GUI, however it is extremely simple with Command Prompt. Using the command "sconfig" and selecting windows update settings (by typing 5) and then selecting manual updates (by typing m) you can set the update mode to manual.

Having just used Command Prompt, I chose to configure the iDRAC next using PowerShell. Through PowerShell all the configuration I did was the IP Address, as I had already configured the rest of the settings in the Dell USC. Continuing the theme of configuring Network settings, I configured a static IP address and assigned another of the ethernet connections to be a virtual switch — I also left a third NIT open and using DHCP, without a specified role.

I continued working with networking, but turned my focus more towards configuring some of the basic security. The simplest configuration was to disable RDP (as mentioned previously), and making sure RAS (Remote Access Services) was enabled. Next I enabled automatic IE Enhanced Security, and disabled Azure Cloud Connect Services. The most important security configuration was to configure Windows Defender Firewall. When

configuring Windows Defender Firewall, the thing I had to look for initially was to find port 80, 8008, 8080, 443, and 8443. Later I also opened up port 24 when troubleshooting (the aforementioned alternate ports for HTTP an HTTPS were also opened for trouble shooting purposes).

These were the main points to the configuration of the operating system of the server, and I do not believe that I left anything out, there were some things I may have done differently on this build of the server, as I didn't make an effort to memorize the process until the very final build of the server. That is why there are some things, like the static IP, that I did not know to configure for this configuration, however it was one of the first thing that I configured in the final build.

Once the Server OS was configured, the next step was to configure roles to run on the server. Any GUI version of Windows Server (and some of the headless versions) comes with the Files and Storage Services role pre-installed and configured at a very basic level. The other roles, on the other hand, had yet to be installed. I installed the three following roles and the features to run them (also listed below):

**Roles:**

1. Hyper-V
2. IIS
3. Remote Access

**Features:**

1. .NET framework 3.5
2. BitLocker Drive Encryption
3. RAS connection Manager
    1. Hyper-V Management tools
    2. Windows Update Services Tools
    3. Remote Access GUI and Command-Line Tools
4. Windows Subsystem for Linux
5. Windows Server Backup
6. Web Server
    1. Default Document
    2. Directory Browsing
    3. .NET Extensibility 4.7
    4. ISAPI Extensions
    5. ASP.NET 4.6
    6. WebSocket Protocol
    7. HTTP Errors
    8. HTTP Redirect
    9. HTTP Logging
    10. Static Content Compression
    11. Request Filtering
    12. IIS Manegement Console

After the server restarts and everything is installed, all the roles still have to be configured. I chose to start with the configuration of Hyper-V first because I expected it to be easier for me because I had used VMware and VirtualBox rather extensively in the past as well as worked with Azure VMs which are running on a Hypervisor.

After opening the Hyper-V role and right clicking on the server, launch the Hyper-V Manager. In order to create a virtual machine, you first need an ISO, so I downloaded a Windows 10 Pro for Workstations ISO on my Mac, transferred it to a USB drive, and transferred it into the document folder on the server. Now that I had an ISO disk image as well as all the features needed to run Hyper-V, I returned to the Hyper-V manager and hit "New". After hitting new, I selected "Virtual Machine" and specified that it will be a Gen. 2 virtual machine[48]. Now I had to decide how much memory and storage to allocate to the VMs. Given that I had 64GB of memory and 1.8TB of useable storage, I felt comfortable allocating 16GB of dynamically allocated memory to each VM and 511GB of expandable storage to each.

Selecting "Gen. 2" and allocating computer recourses for the VMs, I input those values into the Hyper-V manager and selected to create the VM with the ISO image already mounted. I selected the Windows 10 Pro for Workstations ISO image from before and hit "Create". Now that the VM was created, I connected to the VM, hit "Start" and pressed the space bar to boot to the virtual DVD/CD ROM (the ISO image). From here, the process was the same as it is to install Windows 10 on any other device, however with some setting specifically chosen for security reasons.

After booting from the installation media on the VM, I selected "install", hit "continue" and and entered my product key. Then I selected the virtual disk I created as the installation location and waited for Windows 10 to install. Next I selected "United States" as my region and selected "us" as the layout for my keyboard and skipped adding an alternative keyboard layout. I

selected that I was setting up this device for personal use and signed into the Microsoft account I created for these VMs. Next, I created the pin for the computer that was a passphrase based on historical documents that I had previously memorized. This meant I was able to give them a length of 20 characters that included symbols (punctuation from the document), numbers (replacing words ie. for becomes 4), and letters (the first letter of each word or substitutes ie. see becomes c) without having to worry about forgetting it.

Next I configured all the trackers and identifiers (I disabled all of them) and skipped auto customization and tailoring. Next I also skipped connecting an Android phone, disabled OneDrive, skipped installing Microsoft Office, and disabled Cortana. Then once the VM restarted, I signed in, selected that it was not discoverable on the network, and installed all of the available Windows updates. From here, I repeated the steps to create a second virtual machine, named VM-0 and VM-1 respectively. Now that the Hyper-V role had been configured, I began my first attempt at configuring IIS Web Server. I would, however, not successfully configure this until I did more research and gained a better understanding of how the system works, as I describe in the next section.

This initial attempt at configuring IIS began with opening IIS Web Server Manager, and navigating to the “sites” panel. From here I right-clicked on “sites” and selected “add website”. In the website configurator window, I entered the name of the site, entered the path to the folder with the index.html file (and all the assets I imported from my Mac after creating the website) and entered my domain into the hostname field. I also went into the “root document” panel under

“my site” and pointed it to look for the index.html file specifically[49]. This will only sort of work, if you hit the “browse:*80” button on the inspector panel in the IIS manager, the server will be able to show your site locally, but it is both insecure and undiscoverable to any other external machine (a machine on the Wide Area Network (WAN) rather than the Local Area Network (LAN) ).

# Learning More About Internet Protocols

While working on setting up IIS Web Server I realized that I lacked a fundamental understanding on networking as a whole and that I needed to supplement my knowledge in this before I was going to be able to create a solution. Up until this point, in this aspect, I was trying to find someone else's solution to my own problem that I didn't understand, and I realized that there would be no way to get this to work under those circumstances.

So I reached out to my uncle, Scott Stoller, who has a PhD in computer science security and teaches undergraduate and graduate classes at Suny Stonybrook (which just so happens to be one of the schools I'm applying to[50]). While he does not work with networking anymore, he was able to tell me to look into how routers work, how internal vs. external IP addressing works, and about how nameservers and domains work as a whole. While it may have been nice if he was able to give me a complete solution on the spot, he was able to point me in the right direction and guide my subsequent research.

**A thought experiment:**

Let's consider a situation where two devices are sending data across the LAN, such as when a PC is sending data to a printer using direct-IP printing (how printing is handled in my own LAN). The PC has an IP assigned through DHCP by the router (lets say 192.168.1.101) and the printer has a static IP (lets say 192.168.1.102). The PC will select

a port for its TCP connection with the printer, which is listening on another port; the router will have stored what port the printer is listening on (1025) and the PC will choose a port (1024) and the router, if the port is available, will then route the connection from the PC to the printer. The print information will be sent from the PC to the printer through a series of TCP packets.

This is a slight over simplification, as subnet mask resolution and a Address Resolution Protocol (ARP) request to determine the MAC address of the target is used in order to complete the communication, however that is essentially how such communication is conducted. This is similar to how devices communicate with the WAN, however it gets slightly more complicated.

**A (slightly more involved) thought experiment:**

Let's consider a situation where a device on a LAN is sending data to the WAN, such as when a PC is accessing a particular web server to send data. Lets take the same PC as last time (with the IP address 192.168.1.101) is trying to access a server with the IP address 211.100.100.1. The PC checks to see if the destination is on its same network, and since it is not, it send the request to its default gateway (the router). The router it is configured to use has the internal IP address 192.168.1.1 and has an external IP address 87.100.100.10. The PC knows the server is listening for an HTTP request on port 80, and so in its request to the default gateway it specifies in a TCP segment (inside the TCP Packet) that it has a source port 1024 and a destination port 80, it will (in the same TCP

Packet) specify the source IP address (the PC's IP address) and its destination IP address (the web server's IP address). The PC then encapsulates this (with the source and destination IP address[51]) and sends it to the default gateway's MAC address.

The router has a routing table, which it this case would show two entries, one for each gigabit interface (G0/0 and G0/1). Devices on the LAN are connected to the G0/0 interface, so the entry is " c 192.168.0.0/24 connected, G0/0 ". The second interface, however, has an entry " c 0.0.0.0/0 connected, G0/1 ". This is unusual because it has an all zero IP address and subnet mask, however because the G0/1 is the WAN gigabit interface, it has a special trick. The all zero IP address and subnet mask will match all possible destination IP addresses and forward them from the WAN gigabit interface to the ISP.

This is called a "default route" and it is used because a router will only forward a packet if it has a match for the destination IP address in its routing table. If it did not use the default route, it would instead need to have an entry in its routing table for every possible destination on the internet which would require a tremendous amount of energy and computing recourses to look through that table, update it, and more. By forwarding everything to the ISP, the router is using the ISP's routers assuming they have the resources and routing information to deliver the packets.

In order for the router to forward the packet from the G0/1 interface to the ISP, it has to update the TCP packet. The initial TCP segment which includes the port the PC is listening on and the port it is trying to connect to the destination on is left unaltered, however the rest of the packet and the encapsulating frame is changed. The router preforms something called a Network Address Translation (NAT) which replaces the source IP address on the TCP packet with the router's external IP address, and then encapsulates this updated TCP packet in a frame with the source MAC address of the router, and the destination MAC address of a router within the ISP. Subsequent routers used to forward the TCP packet towards the destination web server will not preform a NAT, however they will re-encapsulate the TCP packet with their own MAC address and the MAC address of the TCP packet's new destination.

So how is this exactly related to my project and the IIS Web Server that I am hosting on the server? Well, for any router on the WAN, the IP address of my server is 73.195.22.168 (the public external IP address of my router that is uniquely identifies it to the WAN), however the server has its IIS site bound to the IP address 192.168.1.120, so there has to be an intermediate step preformed to get the TCP packet to the server. This step is called port forwarding.

If the router had not had a computer send it a packet with a segment with a source of port 80 (or 443 or whatever other HTTP/HTTPS port), if the router receives a packet with that destination port, it will simply drop the packet. However, if you log into a router's settings and access the port forwarding table, you can create a fixed port forwarding rule so that whenever a

packet comes with a segment with a destination port of 80, it will always get forwarded to an IP address.

So by setting up a static IP address on the server, and binding it to the web server (add multiple bindings with all acceptable ports if more than one), you can create an entry in the port forwarding table ( in :80 forward 192.168.1.120:80 ). Once you know that a certain port of the server is bound to the web server, that port also has to be opened on the firewall to ensure that it's allowed through (on both the router and the server).

When creating DNS records for the domain, you need to specify the IP address the domain answers to, however, as it was mentioned previously, the router's IP address is assigned dynamically using DHCP from your ISP. This is where another form of DNS comes in: DDNS or Dynamic Domain Name System. A DDNS client is a way to get around the need for a static public IP (which can be expensive) by automatically checking and updating both A and AAAA DNS records.

The source IP address can be assigned dynamically (as it was in the thought experiments above) and it can send data to external clients without any difficulty, so a DDNS client is used to send a TCP packet holding the information about the current external IP of the router to the DDNS provider. There are two ways to implement this, in the router, or on the machine that is being accessed through the router (the web server). Most home routers have DDNS clients pre-

installed (most commonly no-ip) however all the DDNS providers I have seen also have a client that can be installed on the host machine.

If I created a DNS record without using DDNS, it would work for a period of time until my ISP changed the external IP address of my router. In order to have this continue working, DDNS is used to keep the answer value up to date. The first DDNS client that I used was no-ip, which is what is built into my router. I signed into the no-ip client on my router, specified my hostname, and set it up to auto-update the external IP address. Unfortunately, however, my router would not connect to the no-ip servers, and after trouble shooting, talking to no-ip, going though the NetGear community forums (you have to pay for support unless you made the purchase within 6 months of calling), and searching online, I gave up using the router client and chose to use the server client instead.

To do this, I downloaded the .exe executable on my Mac, transferred it to the server via thumb drive, and launched the client. This client was able to connect to the no-ip servers and update the DNS records. This gave me the hostname “garrettgr.ddns.net” however I used a domain I had gotten for free through Name.com (garrettgonzalezrivas.social) and used a CNAME record (DNS alias) to point it to my no-ip hostname using ‘Masking” to hide the URL that it directs to and provide the page title “Garrett G-R”.

When using these URLs from computers on my local network, I was able to connect to the web server and interact with the website through HTTP without difficulty, however now that

I had accomplished this, I turned my goals towards security. First I realized that I would need to switch from using HTTP to using HTTPS, and I knew that the difference was the encryption process used with HTTPS however I did know how this worked or how to get a site/server/ domain to use the HTTPS protocol. In order to be able to harden the server, I was going to have to do even more research.

The first thing I discovered was that HTTPS is secured through a system called SSL[52], or the Secure Socket Layer, however I soon found this to be misleading as this was replaced by the TLS, or Transport Layer Security, protocol. Despite this, the term SSL is still widely used, and on both IIS and SSL providers (and domain/DNS providers) they almost exclusively use the term SSL and even the name of the certificate used for the TLS protocol is an “SSL Certificate”.

The TLS protocol has two different sets of keys, the first is used for the TLS handshake, and the other is used for the session itself. The TLS handshake uses asymmetric encryption (also called public key encryption), meaning that it has a public key that is available publicly and a private key which is used server-side and kept private. While this may be a circular definition, the names are pretty self explanatory for how the keys are used, the important thing is that the data encrypted using the public key can be decrypted using only the private key, and vice versa.

During the TLS handshake, the client and server communicate using the public and private keys to exchange randomly generated data, which is used to generate the second set of keys used by the TLS protocol: the session keys. Unlike the asymmetric encryption used in the

TLS handshake, the session keys use symmetric encryption, meaning that both the client and server utilize the same key. Once the session keys are used to establish a connection, the public and private keys are not used again, and once the session ends, the the session keys are not used again, and a new set will be generated when a new session is started.

There is, however, even more to the TLS protocol than this, it also includes a Message Authentication Code, or MAC (it should be noted that this is different than a MAC address which is used within a frame as a unique identifier of a network interface and stands for Media Access Control address), which is a digital signature originating from the server used to confirm that the communication is coming from the actual web server. This is to prevent something called an on-path attack and domain spoofing. The MAC signature assures both that the data originated from the web server and that it was not altered en route to the client.

In order to use the TLS protocol you need an, admittedly misleadingly named, SSL certificate. An SSL certificate contains data about the identity of the server's owner, the public key for the TLS handshake, and some other data. While, technically, any server owner can create what is called a self-signed SSL certificate, browsers do not 'trust" SSL certificates that were not issued by a Certificate Authority (CA). In order to obtain an SSL certificate, you purchase one from a CA and install it on the server. The CA will confirm that you are who you say you are and keep a copy of your SSL certificate.

Now that I had learned about SSL and HTTPS, I knew I would have to make some adjustments to how I was managing my domain in order to get an SSL certificate. My first approach was to go to name.com where I had my terminal domain, which I knew was a CA, and purchase a single-domain SSL certificate through them (I will go though this process in the next section), however this proved unsuccessful and I had to go searching for another CA and domain provider for my web sever.

# One Last Time

After the corruption of the previous — the penultimate — build of the server, I repeated essentially the same steps in setting up and configuring the server for this build. Only rather than spending over a week configuring it, I did it from about 10pm to 2pm. I do not know exactly what happened to the server that caused the failure, however I have a suspicion.

When I was working on the server at the time, I was looking through the IIS SSL settings as I was researching hardening the website and web server as a whole. The server came up with a notification that it had to run updates. Since I was not actually doing anything, I told the server to install updates and restart. Unfortunately, this is where something went seriously wrong. Upon booting back up, I was presented with a message saying my Windows was damaged and could not be repaired and to install media including a snapshot of the server or a backup of the server.

Unfortunately, the tutorials that I watched had their servers running on VMs and likewise never bothered to configure WSB and I had not thought to do it. The only backup service I had used in the past was Time Machine on my Mac to backup to iCloud; however I have used the repair feature on Windows 10 installation drives to repair PCs, so I inserted my Windows Server 2019 Standard Edition installation media. After booting from it I selected the option “Repair you computer” rather than “install”, after selecting this, it searched for local backups or snapshots

and returned saying that it was unable to find anything. Next I attempted the same thing with a Windows 10 installation media, with the same result.

My next course of action was to check out the raid array to see if something was wrong and if it was if I could rebuild the array and recover the server. Booting into the Raid Controller Configuration menu, I was presented with a failure on VD0 saying that PD1 was missing and that there was a “foreign” physical drive installed in drive bay 1[53].

First, I tried to repair the RAID array, this however failed, and so I decided to rebuild the RAID array. Rebuilding a RAID array can result in the loss of data even in the best circumstances. With SMR drives, there is a substantially higher chance that the rebuild would fail and result in total data loss. And that is exactly what happened when I put in a new drive to replace the drive that I thought had failed with a fresh drive and told the RAID controller to rebuild the array.

It was now towards the end of July, and I was completely without a running server. I knew that I would have to work quickly, efficiently, and error free. Realizing this, I did what I do whenever I need to get a large amount of work done in a short period of time: I drank too much coffee and stayed up all night. In about fourteen hours, I not only got this new build of the server up to the point it was at before, but I took several new steps in its configuration to incorporate what I now knew.

In order to get a new RAID array configured, I replaced PD0 in drive bay 0 as I knew it to be fully functional, and for drive bay 1, I use a different 1TB HDD, which I ran hardware diagnostics on to ensure that it too was fully functional. I then cleared all the existing configurations (VD0) and formatted PD0. From there I created a new Virtual Drive — also named VD0 — added both PD0 and PD1 and initialized it. This process was significantly faster than the first time I did it, and I believe that this may be in part due to the previous PD1 drive having a fault that required the initialization write to have to loop through a lane (or likely many lanes) several times (a common fault with failing SMR drives).

From here I followed the same steps in installing windows server, and even the initial configuration was identical in the roles and features installed (there is a chance that some of the installed features were added in this new configuration and not in the penultimate or other earlier builds however those are the features seen on the final server). The first major deviation was in setting up IIS Web Server. After selecting “add website” in the website configurator window I entered the website name, the physical path, and the hostname for the website. However, I also changed the protocol to HTTPS, and the binding with the static IP to port 443. From here, I did the same adjustments for the root file but then I went to the advanced settings menu. I first enabled HSTS[54] (HTTP Strict Transport Security) and opened the HSTS settings. Here I set “includesubdomains” to true, set max age to “31536000” (one year — considered a best practice), enabled pre-load, and most importantly enabled redirection from HTTP to HTTPS. In the HSTS menu, I also had to specify the connection type to be HTTPS rather than HTTP.

Now that I had all this configured, I had to navigate to Windows Defender Firewall and make sure that both ports 443 and 80 were open for incoming connections. Next I went to configure an SSL certificate for the server. My first approach was to actually ignore the DDNS system (no-ip) and use name.com to issue me an SSL certificate and continue using the domain I already had, and using a program (I used honeybadger because I got it free through GitHub Student Developer pack) to check the uptime of my website, and if it wasn't accessible, I would update the IP address manually in the DNS records. I would have to do this because name.com does not support external DDNS clients and does not offer a native client.

In order to get an SSL certificate, the first step is to pick one and buy it — then you need to actually get it issued to you. The first step in getting it issued is generating a CSR, or the Certificate Signing Request, which includes information — your organization, common name, and location — that the CA needs in order to create your certificate. To do this, I navigated to the main page of the IIS manager and selected "Create Certificate Request…", entered all my information, and exported it to a txt file. From here I put this txt file on a thumb drive, transferred it to my Mac, and pasted the CSR into name.com to create my certificate. Next you have to validate your certificate, this is to prove that you own the domain, and is called Domain Control Validation (DCV). To do this, you can upload a validation file, add a CNAME record, or receive an email; I chose to validate through receiving an email.

From here you go download the SSL certificate files and go to the "Complete Certificate Request…" panel in IIS to import your certificate — or that's what you would do next.

Unfortunately, name.com does not allow you to download their SSL certificates, and instead will only show you the key as plain text in a PEM format that you can copy and paste into something else. Now this should be able to be remedied, you can copy and paste this into a txt file, convert it to a .PEM file and then use OpenSSL to convert it into a .pfx that IIS can use (IIS uses proprietary formats). Unfortunately, however, after two days of work, many attempts with OpenSSL and other tools, I was unable to create anything that would be accepted as a valid certificate. I concluded that it would be better to find a different CA that would be able to issue a certificate in a more standard way that would allow me to download the certificate in every format and then use only the ones that my web server needed.

This turned me towards the only other domain provider I was familiar with namecheap. I had used namecheap once before when I was hosting a website on GitHub Pages, so I researched if I could use their SSL certificates on IIS10, and I found a large amount of very helpful and very useful material and documentation on precisely that topic (from both namecheap and other sources). From here I purchased a new domain and a new SSL Certificate. Going through the same process I generated a new CSR and validated my new domain with the providing CA and was issued a valid SSL (DV) certificate from Sectigo RSA Domain Validation Secure Server CA (the providing Certificate Authority). I was able to download a .zip file to my Mac that contained the certificate in ten different formats (six ASCII and four binary). I copied the .pfx certificate over to a thumb drive and copied it to the server. I then opened IIS manager, selected “Complete Certificate Request…”, put in the path corresponding to the location of the .pfx file, and the friendly name of the website (the domain). Once I completed all these steps I was presented with

a message saying the certificate was active and that the corresponding private key was installed on the server.

The SSL certificate secures both www.garrettgr.com and garrettgr.com, and the next thing was to set up DNS. One of the reasons I chose to go with namecheap in my research was because they had their own DDNS client that I could install on the server. To set up DDNS I first created what they call an “A + Dynamic DNS Record”. After selecting that DNS record type I had to specify the host, the value, and the TTL (Time To Live). Because I had two subdomains in my SSL certificate (a www. subdomain and a wildcard @. subdomain) I created two records, and filled out each host accordingly (one with www and one with @).

I then set TTL to automatic because I didn’t really understand how it works and while some people said that 300 was best practices, and when I read through forums I found that most people didn’t understand it and either went with the default of the DNS provider, used 300, or used automatic if the DNS provider had that option.

Once the DNS records were set up, I had to link them to the DDNS client that I installed on the server. To do this, I generated a unique code for each record and created a “profile” for each of the records and used that code to bind that profile to its corresponding record. I also had to specify the host, domain, and the update period. Finally, I checked "automatically detect my public IP” and, after refreshing the client, it was able to connect and update the DNS records on namecheap.

The final step in configuring DNS was DNSSEC, or Domain Name System Security Extensions, which is a set of protocols that add cryptographic authentication for responses from DNS servers. This mainly protects against DNS spoofing and hijacking. Unfortunately, router pharming and compromised DNS resolution at the ISP level cannot be protected against. DNSSEC works by using public key encryption (the same kind of asymmetric encryption used in the TLS handshake), and a strict hierarchy structure. On its most basic level DNSSEC works by whenever a DNS request is made (requests IP address for domain) by a client, the request is sent to a recursive name server. When the recursive name server then requests the IP address it also requests the DNSSEC key associated with that zone; this key ensures to the recursive name server that the IP address it is receiving is identical to the record on the "Authoritative Name Server" and was not modified in transit.

There are two types of zones: parent zones and child zones, where the parent zones (the higher level zones) sign the public keys of the child zones (lower level zones). The public key is published with DNS while the private key, like with the TLS handshake, is kept private. The Authoritative Name Servers are what actually answer DNS queries, however other DNS servers are allowed to cache data for a period of time to improve response times. After the recursive name server proves that the IP address it received is both valid and not tampered with (by using the DNSSEC key) it resolves the domain[55].

This provides security in two main ways: that the IP address is the correct IP address for that domain, and that the domain name exists. Together these protect against man-in-the-middle attacks, redirection attacks (the domain is redirected to a fraudulent IP that takes the client to a pharming or phishing site), DNS spoofing, and DNS cache poisoning. DNS spoofing and DNS cache poisoning are essentially the same thing, they both attack the DNS protocol at the resolution process such that when queried, the DNS server returns the incorrect IP address.

Now that I had DNS set up, and everything with my domain and SSL providers configured, I restarted IIS and went to my Mac to confirm everything was working. I typed in "www.garrettgr.com" and it took me to my website without any difficulty whatsoever, and furthermore it connected through HTTPS and I was able to view the SSL certificate details in safari. Next I typed"http://www.garrettgr.com" in order to hit port 80 on the server, however HSTS acted as intended and redirected my browser to use the HTTPS protocol and it connected through the secure connection. Next I used my semi-functional PC, and both my mothers and sister's PCs to confirm that it was working on multiple devices and it had no problems.

# Intranet vs Internet

Next I tried to access my website from the WAN, or the internet, rather than the LAN, or the intranet, as I had been doing before. This I knew would be the final test of whether or not the web server was actually functioning as intended or not. To do this, I turned off WiFi on my Mac, connected it though my phone using bluetooth tethering and used cellular data to ensure my connection was coming from outside my network.

I first tried simply to navigate to www.garrettgr.com on safari and got the following message:

```
Failed to open page:
Safari Can’t Open the Page

Safari can’t open the page “https://www.garrettgr.com”
because the server where this page is located isn’t
responding.
```

Seeing this, I immediately, knew that something wasn’t working right, so I started to go thorough a debugging process. My goal was to eliminate everything that the problem couldn’t be, and whatever was left must be causing the problem — Sherlock’s Dictum[56]. The first thing that I tried was to go and open other websites on my Mac still using the bluetooth tether to

ensure that I was able to access the web and that the connection worked. Because I encountered no difficulties, I tried my next thing: a different browser.

The only browsers that I have on my Mac are Safari, Firefox, and Tor. I know that Tor can have some difficulties with connecting some websites, so my other readily available choice was Firefox. I was also aware, however, that Firefox has had many issues with SSL and establishing a secure connection. I decided that the message I received would likely indicate whether it could not connect to the server or if it had a problem with SSL. Launching Firefox, I repeated the exact same steps, and was faced with the following message:

```
Unable to connect.
```

Admittedly this looks like it provided no information at all but, in one significant way it did. When Firefox encounters its notorious SSL error it presents the message:

```
Secure Connection Failed
```

With this result, I had decided that it must be a connection failure to the server. So I then tried to connect to the server again though the LAN by reconnecting my Mac to Wifi and the website opened without any difficulty. This is where I forgot something that lead to some troubles in the next step — however I will explain that in just a moment. From this I determined that the issue could not be the firewall on the server because it allowed connection to port 443

(and 80) and therefore should be reachable (and was reachable on the LAN), however there might be a firewall on the router blocking external connections.

My router, unfortunately, does not have configureable firewall settings, and logging into it (even in the advanced dashboard) the only security options it shows are: access control, vpn client, block sites, and block services. Access control is a panel that allows you to block certain devices from accessing the network and the service was (and is) not enabled. The vpn client is a panel where you can purchase use of NetGear's VPN client, this was (and is) not purchased nor in use. Block sites is a panel for managing content filtering either through keywords or domains — this was never configured. Block services is a panel where the network administrator can block certain clients (from their MAC address) from accessing certain services (such as http), this was and never has been enabled.

From here I concluded that there are likely three things that could be going wrong: the ISP or router had built-in filtering for incoming connections, the DNS/DDNS client was not working, or the server was not configured correctly. This is where my mistake from earlier became an issue: reconnecting to my phone to establish a connection through the WAN, I tried to navigate to www.garrettgr.com and to my delight it connected immediately, and I was able to scroll through the website, enlarge the images, and I decided that maybe the domain or DNS provider simply had to update. I declared the project completed, and didn't look at it for nearly a day until my friend asked if she could see my website and I texted her the URL.

She was, as you may have expected, unable to connect. I had forgotten that most modern browsers will cache even the html and assets of a website rather than just their IP address. Now I realized that I still had more work to do on this project before it would be complete.

I figured the two next things to try would be to first: try to access the web server by typing in the external IP address of my home router and even specifying port 443 rather than using a domain (to eliminate the possibility that the domain or DNS was not configured properly), and to second: contact my ISP to see if they had any filtering on their home consumer connection.

I received the exact same error message when I tried to access the server through my router's IP address, eliminating (or at least partially eliminating) that the failure was in the configuration of DNS, DDNS, or the domain. However I had the idea that possibly the failure was in the SSL certificate (I had, after all, had significant difficulty with it). To eliminate this possibility, I went into the settings on IIS and disabled HSTS (so that I could connect through HTTP without it being redirected to HTTPS), made sure that I had a binding for port 80, and that port 80 was open on Windows Defender Firewall.

After doing this I first tried to access my website on the LAN with http://www.garrettgr.com and was able to establish a connection using the HTTP protocol. After this, I reconnected to my phone to establish a connection from the WAN and tried the same thing. Unfortunately, I was greeted with the exact same message as before, that safari was unable to

establish a connection. My approach at this point was to — again — systematically eliminate everything that the problem wasn't. To do this, I called Comcast (my ISP) and asked them about any potential connection filtering.

Because we have our own 3rd party router, I found it doubtful that they would have any filtering (although I believed it might be a level of connection kind of thing to make people pay to host anything or something like that), and they didn't. This left either something on my router or something with the server. So the next step was to look into the router. I knew that it was common for routers to block port 80, so I reconfigured the bindings on my website to be on the port 8080 (one of the alternate ports of HTTP with port 8008) and tried to access the server from the WAN using its IP address and specifying port 8080.

This was one of the solutions that I had the highest hopes for doing, and I in fact expected that port 80 would be blocked by default — however I had not expected that port 443 to be blocked (and I am still not sure it is) — however this again was met with failure. So I tried port 8008 and still found that it was unable to connect. Deciding to return to HTTPS, I added a binding for port 8443.

Port 8443 is technically not an officially recognized alternate port for HTTPS, but is so widely used and accepted as an alternate HTTPS port that many systems are created to recognize 8443 as an alternate for HTTPS. After creating that binding and again trying to connect to my website, I was again met with failure.

Up until this point all the ports that I had been trying to use were HTTP or HTTPS ports, and I had the idea that NetGear (the company that makes my router) may deliberately filter all the common web hosting ports on their consumer grade routers in order to make customers pay for them to be unlocked or buy an enterprise-grade router.

For this reason I tried another binding: HTTP on port 24. I didn't have particularly high hopes for this, however, and my suspicions were proven right when it still would not connect. At this point I did not know what else to try, but I turned to NetGear and their community forums to see if I could find some answers. I unfortunately was unable to use NetGear support a they only offer free support for six months after your purchase, and this router was purchased over two years ago.

Going through NetGear's community forums, I found something particularly interesting, on the posts regarding similar issues, about a third of the users said to go to the firewall settings (and even included screen shots) and to configure it in just such a way, while another third exclaimed that they did not have any firewall settings (and included screen shots of the lack thereof), and the final third seemed to be generally confused and not saying anything of relevance.

I unfortunately found myself in the second category, and scoured through the control panel on Safari, Firefox, Edge (on my x250), and even used their beta downloadable client (also

on the x250) and was still unable to find any way I could configure the firewall. I looked though both the “basic” and “advanced” sections ad nauseam and spent several days on the simple task of trying to figure out how to configure the router’s firewall before I decided I needed to move on to another approach.

I figured there were two ways to get past this issue with the router’s firewall: use a different router, or change the software on the current router. My first choice, naturally, would be the second option, as I had actually spent some time looking at OpenWRT in the past. Unfortunately my mother was not comfortable with me installing a build of OpenWRT to any other open-source or non-native software on the router.

I am unsure if this would have solved the problem, it would have certainly let me configure all the firewall settings, and I even identified a rather highly regarded build of OpenWRT that seemed rather ideal for this scenario, however I knew that there was no way I was going to be allowed to do it. So I had one more option available: use a different router. I had two ideas of how to do this without having to pay the rather high price of a router. The first was to use Comcast’s rented router for one month and then switch back to our own router, however my mother was concerned that Comcast would use that as some way to increase her bill, and the second was to get a router from Walmart or BestBuy and return it.

While this second option would only cost a relatively cheap restocking fee, ultimately it was also ruled out as a possibility (also because my mother did not feel comfortable with it). The

next thing I looked for was an old or used router on Facebook marketplace or eBay that would be cheap enough but also be able to function for this project (with the intention of running it as-is if it had a configureable firewall or installing OpenWRT on it). Unfortunately, nothing came up in my searches.

This left me without many options, in fact for a while I wasn't able to find anything else to try. When I did think of something to try, it was in fact, an obvious thing to try: using BlackArch and Kali linux scanning tools for enumeration. Unfortunately I did not save the results of my nmap scans, however on my LAN I was able to scan the web server with its local address, the external IP address of the router, and my domain name. I was also able to specify for it to scan the SSL certificate, supported TLS algorithm, and vulnerability to man-in-the-middle attacks (not useful for trouble shooting the issue but good for testing the security which was the goal of the project) using:

```
nmap --script ssl-cert, ssl-ccs-injection,
ssl-enum-ciphers -p 443 www.garrettgr.com
```

There are actually three different scripts being run, however I wrote all three of them into a single line. There are other scripts that can be used for searching for known vulnerabilities but the only one that I ran was checking for man-in-the-middle attacks because it was one that I had seen in TestOut LabSim and felt like it would be incomplete if I didn't utilize it in the project.

When running this on BlackArch linux (my preferred penetration testing distro although I did use kali linux for some parts as that was also part of the project), connected to the LAN, I was able to view all the details and had no issues. Furthermore, I was able to confirm that ports[57] 22, 80, and 443 (for ssh, http, and https respectively) were all open.

The results from this scan and everything that I outlined above led me to the conclusion that the issue was not with how the server was configured but something out of my control with the router. The only thing that put that into question for me was that I was still unable to connect to the server through port 24, as it would be unlikely that a firewall would have a rule specifically blocking an uncommon port, however I had nothing else that I could think of to try.

And that leaves me at this present day, where I have just about finished this report, even though what I have to report is, at least in some ways, failure. At the same time, however, the project succeeded in another of its goals: to teach. This project has been a remarkable learning experience, I have learned not only more about the internet, security, and system management / IT — but also more about myself. I have included much of what I learned (that was relevant to the project) from my research, however I haven't really talked about what I learned about myself from this project.

I have worked on some independent projects in the past, mostly small coding projects that I found or thought of that I thought would be fun, but I had never worked on a project of this magnitude that I had to manage completely on my own. One of the most challenging parts of this

project wasn't the project at all, but continuing to work and make progress on it. In the past, I have often procrastinated until the deadline and then used that pressure to motivate me to be able to complete it. With this project, I wasn't able to do that. If I had waited until the deadline there would have been enough time to be able to complete it, especially with all the complications and difficulties that I encountered.

The first thing I tried in order to keep myself working efficiently was to artificially create deadlines for myself. Saying things like "I have to get to this point by that day" or "I have to have that set up before this time". Unfortunately, I didn't find these to be particularly helpful, so I tried using a schedule that I would work for a certain amount of time each day and would complete the project by a certain date. This had two main problems: just like the previous attempt it wasn't particularly successful, and it isn't really possible to create a schedule for a project that you don't know the length of.

The final approach that I ended using was to just stop over thinking everything. I had found that I was spending more of my effort worrying about working on the project than I actually was spending on the project itself. I know I have done this before, where I would get consumed with the idea of starting some project and working on it that I get stuck in my own head and unable to actually do anything. The way I was able to break out of this cycle was to write just a very basic checklist in my notebook (to keep track of the project in general) and just start working.

While this was certainly better than not working at all, there were certainly problems with this method. I found that without having a direction, I would spiral into trying to make a particular aspect of the project perfect and lose track of the actual goal of the project. I can't honestly say that I discovered some great thing that broke me of this, however I did become aware of it, and once I noticed it, I was able to get myself back on track with the project. To reduce this in the future, I started journaling. I would write down my goals for the day and take notes as I either accomplished them or had to change them (and if so, what the reason that made me change it was).

With all these, I became much better at self regulation, I also found that by working with other people near by I was able to be much more effective. At some points when I was writing this report, I worked it in a Panera because I was able to stay much more focussed, I don't know why but I know that it works. I also moved my desk out of my bedroom and into the office that my sister and mother share, and just being in that working environment I became significantly more productive.

Once my health was better, I starting going to my fencing club, and I also found that adding that physical exercise to my routine was a good way to self medicate for my ADHD since I currently can't take any medication for it. Overall, this project has helped me significantly with self discovery. I also learned to research much more efficiently.

When I started out, and even pretty far into the project, I was researching by watching dozens of YouTube videos, reading reddit threads, community forums, and documentation. While this isn't a particularly bad way to research, it is easy (at least for me) to get lost in the research and lose sight of what you needed to figure so that you can continue working. What I found was that by creating a spreadsheet of things that I needed to understand (and adding to it as I discovered questions I didn't even know I needed to ask), I was able to direct my research and make it much more efficient.

The overall theme, I suppose, is that I learned how to manage my time, my motivation, and my focus. By the end of the project I was able to catch myself whenever I started to drift off from what I needed to be doing, I was able to keep myself from getting distracted by my dog or my phone or anything else, and I was able to actually get up and start working on the project, even though there wasn't that feeling of a deadline crushing down upon me. It is a little ironic that I am talking about an approaching deadline, as I am writing about this in the middle of August, however I have been working this new way for over two months. I have also been using it for nearly the whole month of August so far in order to get this report done. Coming into my senior year, I know that I will be able to put these skills that I have learned from this project to manage my workload, despite my busy schedule, to get everything done in a timely manner.

# Endnotes

**1.** Actually twelve all told, including both the overview section and this endnotes section, however there are ten sections containing the process of the project.

**2.** And many 3rd party sellers also offer Windows Server 2012 R2.

**3.** This is less of an issue with both Windows Server 2016 and Windows Server 2019 as they have what they call “functional levels” which allow them to run software and interact with servers as though they are older versions of Windows Server while still having the added security and efficiency of the newer versions.

**4.** Mostly just print outs of system information, network status, and other basic commands to make sure that the connection was successful.

**5.** I have started using VS code more recently, and have also used Xcode to a certain degree (and Processing 3), however there are only two IDEs that I have actually used to any significant amount: CodeRunner 3 and the JetBrains suite (which I also got through the GitHub student pack)

**6.** Attempting Windows repair with installation media for both Windows Server and Windows 10, attempting to use bare metal screen shots of other servers (with no roles but with the Windows Server operating system installed), and terminating in the attempted rebuild of the RAID array.

**7.** This is hyperbole, I in fact have no idea what the lethal dose of caffeine is for a horse nor how much caffeine I consumed in this time period, however it was a considerable amount.

**8.** The only section of the project that reviled the amount of detail and research I put into the beginning of the project was the eighth section when I researched internet protocols. And while I did a similar amount of research, I did it in only a short number of weeks rather than over two months.

**9.** This analysis will ignore licensing costs, renewal costs, updating costs, and other similar issues that do not need to be dealt with when using a student license.

**10.** The original end-of-life was slated to be January 14, 2020, however this date was extended to be October 10, 2023.

**11.** Every bare Windows Server install (post 2004) has included File and Storage Services by default, and the roles that I added for this project included features for Internet Small Computer System Interface (iSCSI) Virtual Disks, File Share, and Work Folders, however none of these have been initialized for security purposes (more information in the section about hardening the server). Furthermore, at one point in this project (on two installs of Windows Server 2019 Standard Edition that were later destroyed) I did initialize, configure, and try to use Active Directory, however this approach was changed for the final several iterations of the server in the project.

**12.** Hyper-V in both Windows Server 2019 (all editions) and Microsoft Hyper-V Server now natively supports hosting Linux Virtual machines.

**13.** Windows Server 2019 takes on a “expect breach” model to security. While the standard model for security that was to assume that firewalls along the system’s perimeter would stop any and all security compromises, the new model is to assume that “all servers and applications within the core of a datacenter or server cluster have already been compromised.”

This model applies to single-server, or “isolated”, server configurations as well. While this new model assumes that the system is already compromised rather than wholly depending on stopping any attack dead at the firewall, it still relies on having a strong firewall to make the work that is put into detection, mitigation, and expulsion of security compromises ultimately redundant and unused.

**14.** As well as natively run Linux VMs.

**15.** The other three editions (Windows Server Essentials, Standard, and Datacenter) can all be installed and run headless with nearly full functionality, however Windows Server Essentials especially was designed to run with a Graphical User Interface (GUI).

**16.** Windows Server Core is not its own unique operating system, it is however an architecture that can run both Windows Server Standard Edition and Windows Server Datacenter Edition. It is a smaller, more efficient, and headless build of the common Windows Server architecture.

**17.** The main benefit of the smaller patches is that it takes less time for the server to reboot after the patches are installed, further reducing server downtime.

**18.** That is, the most similar since the overhaul of Windows Nano Server in late September of 2020.

**19.** It was also preferable if the server was already several years old so the price would not depreciate significantly over the course of the project and much of the cost could be recuperated by its sale. In fact, I am actually planning on selling each part of the server as well as the gutted server itself to make a profit, the RAM and CPUs alone can sell for nearly double

what I paid for the server. Accounting for the other costs of the project, I expect to make at least some profit off the sale of the server.

**20.** Dell PowerEdge RAID Controller series-6 / integrated.

**21.** Using Multi-Tier Caching Technology (MTC) the cache consists of SRAM, DRAM, and NAND flash caching, as well as a Media Cache all on top of the standard permanent storage. This Media Cache however is a dynamic allocation of the permanent storage itself and is not part of 128MB of cache, and the NAND flash is also not considered part of the cache. I was unable to get any accurate numbers on the split between the SRAM and DRAM capacities other than that there is more DRAM and combined they are 128MB. Furthermore, reading through manuals, data sheets, and other documents I was unable to determine even an accurate estimate on the size of the NAND flash memory other than a vague "10s of gigabytes" in a document about how their MTC system works.

**22.** The controllers use a method to keep the transfer constant — even if it means that the transfer rate will drop from (according to my own experiences) ~80-140MB/s to about ~5-20MB/s — this is achieved by the controller clearing up a lane of data from the cache by writing it to the SMR tracks and then filling up the cache with new data again.

**23.** Which I ignored, and at one point required me to start over, however that is much later in the project.

**24.** Error Correcting Code (sometimes called Error Checking and Correction)

**25.** Double Data Rate Three

**26.** Synchronous Dynamic Random Access Memory

**27.** Registered Dual In-Line Memory Module

**28.** Current Intel and AMD server CPUs are both made using a 45nm architecture. Other CPUs use even more advanced architectures, like the Apple Silicon M1 chip in the laptop I am writing this on which uses a 5nm architecture. If this is considered an unreasonable comparison as the Apple Silicon M1 is technically a SoC rather than a CPU and has the best transistor density of any production chip, IBM has been able to make a processor with a 2nm width multi-gate MOSFET. If this is also disregarded because it is a prototype chip, AMD has several production CPUs with a 7nm process.

**29.** ARCTIC MX-4 (ACTCP00002B) carbon-based thermal compound was used the Thermal Interface Material (TIM) as both a filler and thermal conductor between the Integrated Heat Spreader (IHS) — the die of the CPU — and the aluminum heat base of the passive air cooler that was used. Carbon based thermal compound was chosen because it has high heat transference but is a very poor conductor of electricity.

**30.** IDRAC settings, OS Deployment settings, Encryption configuration, HII (Human Interface Infrastructure) configuration, vFlash media configuration, RAID security key configuration, physical security configuration, and part replacement configuration. Of all these features, I only had to work with the IDRAC settings and part replacement configuration (when I replaced the IDRAC). While I likely should have configured the hardware encryption and the other physical security options, I did not because of how many times I had to reset the server, re-initialize the physical drives and RAID array, and use thumb drives to transfer files between my Mac and the server. Ultimately, in that respect, I prioritized ease of use over maximum security with the intent to configure these once the server was running fully. However since I am still not at that point, I have yet to configure these.

**31.** Important note: this is purely in the spirit of levity and I use documentation, stack overflow, and GitHub to find the correct way to solve problems in my programming… however, in this case I did in fact spend about three hours figuring everything out myself — including things like the levels of RAID arrays, rebuilding RAID arrays, and replacement drives that had no practical application to this project.

**32.** Directly from Microsoft it is $972 however I have seen it for as little as $411.99, and 443.99-489.99 dollars seems to be the most common cost of a license.

**33.** Software as a Service

**34.** Windows Server 2019 Datacenter - 2 Core retails for little over 400 dollars while Windows Server 2019 Datacenter - 16 Core retails for $3,299.99 (from Microsoft).

**35.** I have removed the cpu of a running computer (desktop) in the past, and it just continued displaying the last frame in the buffer, however it would flash bands of color across the screen right as the cpu disconnected. Now I have also had those same kind of image artifacts on a computer with failing memory, however I doubt the memory to be the culprit. I mostly ruled this out because I had put in its SODIMM from an HP EliteBook that I used before it and it never had any issues. The Lenovo x250 has a decent amount of “deck flex” (the amount of deflection in the base of a laptop) and I believe that this puts a strain on the motherboard PCB which in turn, due to a delimitation of the solder joint between the due, introduces a partial loss of connection with the CPU.

**36.** The ability to manage IIS  was lacking until June 27th 2020, and Hyper-V management was supported about a year and a half before that on January 15th 2019.

**37.** Even bash can be used if Windows Subsystem for Linux (WSL) is installed on the server.

**38.** Something that to me was remarkable, was the graphics on the server, there was a significant amount of tearing, artifacts, and “jello-ing” where you could watch the screen refresh top to bottom. I don’t know if this is from using VGA for just the on-board graphics of the server. I don’t have much experience because even most of my laptops have dedicated graphics, and my only other display is always run through DisplayPort.

**39.** The most effective way I have found to create a fresh installation of Windows Server, reinitializing the RAID array wiped all the data by writing 0 over every bit and then reinstalling Windows Server, well, gave me an installation of Windows Server.

**40.** However, I still had license keys for all the other versions of Windows Server from Azure, and if necessary I could create a Windows Server 2019 Datacenter Edition installation media and activate that, however I concluded that it did not make a significant impact on the project.

**41.** This is based off an estimate of about 20 videos with an average length of about 20 minutes. It is likely more than this but I decided this was likely a close enough figure.

**42.** For the first time on my life, while writing this report, I opened Stack Overflow’s home page. For me the closest thing Stack Overflow had to a home page was the google search results, but when looking to see if Stack Overflow was one or two words in this report I just typed in “stackoverflow” into google and opened their homepage. While this alone is an amusing anecdote, I wanted to point out a particular line they had one their website:

"Every {developer, data scientist, system admin, mobile developer, game developer} has a tab open to Stack Overflow"

With the text in the braces scrolling through on a loop. This was a combination of excellent marketing and graphic design that delivered a very powerful message. While this may seem irrelevant to this project — and in a way it is — when I just saw that it reminded me of how many elements I borrowed from other websites that I saw when I found something I particularly liked and recreated in Bootstrap Studio.

**43.** My sister runs a business (a surprisingly successful business that has paid for a surprising amount of her tuition) that involves printing with several different mediums, including printing directly to vinyl, a high quality photo centric printer, and several others. This results in seven printers that are all accessed through the network and all use direct-IP printing through Wifi, and while it wasn't directly related to the project learning about how direct-IP printing works and more about different types of LAN connections was very interesting and I probably lost two or three days going down that rabbit hole.

**44.** By starting the counters at a negative number, by the time each of the counter elements faded in, continuously counting, they would be at around zero, this also meant that I could offset the fading in of each elements to cascade and account for the change in timing by adjusting the value of that starting number.

**45.** After I did most of the work on the website — about two weeks in — I did some of the work I describe in the next section and by the time I exported the 'final' version of the website I already had IIS running with the an older, base configuration of the website.

**46.** Modern computer systems log the last time a file was accessed, modified, or even just had its meta-data changed in the “MAC time” files.

**47.** An interesting story I read about while researching web server security was found on:

http://old.dfrws.org/2007/proceedings/p31-buchholz.pdf

Where the author describes how some observed servers seemed to have multiple wrong times. Essentially, one query to the server would return one time offset, but other queries (to the same server) would return a different time offset, and the server would always return one of the two time offsets on subsequent queries. The author’s conclusion, which I agree with, is that it must have been two or more physical systems (with different time offsets) using a shared IP address for load-balancing purposes.

**48.** UEFI-based firmware, 64-bit operating system, and provides support for some of the newer virtualization features.

**49.** By default it looks for 15 different common file names for the root html file, however, it is considered a best practice to put a specific link to the root file rather than have IIS pick a file on its own (faster, more secure, and lower chance of something going wrong).

**50.** At this school my aforementioned uncle is a professor, my aunt (Uncle Scott’s wife) who also has a PhD in computer science is also a professor but teaches graduate students and does research mostly, and my cousin just got her masters in physics.

The ironic part of it is that it is one of the few schools I am applying to that does not ask if you have any family who attends or works at the university.

**51.** Again this is slightly more complicated using ARP to get the routers MAC address and the such like however, that is not important for this.

**52.** SSL has not been updated in nearly 23 years (SSL 3.0 being replaced by TLS in 1999) and has several known vulnerabilities. Furthermore, most modern web browsers don't even support the out dated SSL protocol at all.

**53.** I use zero indexing for pretty much everything, and in this instance the server's first drive bay is labeled and recognized as "0". PD1 (or physical drive 1) is the HDD in the second slot on the server.

**54.** Used to protect against man-in-the-middle attacks, protocol downgrade attacks, cookie hijacking, and more.

**55.** Domain resolution means that the DNS server provides the requested IP address.

**56.** Eliminate the impossible, and whatever is left, no matter how improbable, must be the truth.

**57.** It also identified other open ports such as 5001 — an alternate port for SSH — however these ports being open was not as important for trouble shooting.

# SystemInfoPowerShell_txt

```
WindowsBuildLabEx                          :
                                 17763.1.amd64fre.rs5_release.180914-1434
WindowsCurrentVersion                      :   6.3
WindowsEditionId                           :   ServerStandard
WindowsInstallationType                    :   Server
WindowsInstallDateFromRegistry             :   7/26/2021 8:54:50 PM
WindowsProductId                           :   00429-00105-30759-AA595
WindowsProductName                         :   Windows Server 2019
                                               Standard
WindowsRegisteredOrganization              :
WindowsRegisteredOwner                     :   Windows User
WindowsSystemRoot                          :   C:\Windows
WindowsVersion                             :   1809
BiosCharacteristics                        :   {4, 7, 9, 11...}
BiosBIOSVersion                            :   {DELL   - 1, 6.6.0,
                                               Dell Inc. - 60601}
BiosBuildNumber                            :
BiosCaption                                :   6.6.0
BiosCodeSet                                :
BiosCurrentLanguage                        :   en|US|iso8859-1
BiosDescription                            :   6.6.0
BiosEmbeddedControllerMajorVersion         :   255
BiosEmbeddedControllerMinorVersion         :   255
BiosFirmwareType                           :   Uefi
BiosIdentificationCode                     :
BiosInstallableLanguages                   :   1
BiosInstallDate                            :
BiosLanguageEdition                        :
BiosListOfLanguages                        :   {en|US|iso8859-1}
BiosManufacturer                           :   Dell Inc.
BiosName                                   :   6.6.0
BiosOtherTargetOS                          :
BiosPrimaryBIOS                            :   True
BiosReleaseDate                            :   5/21/2018 8:00:00 PM
BiosSeralNumber                            :   1292NM1
BiosSMBIOSBIOSVersion                      :   6.6.0
BiosSMBIOSMajorVersion                     :   2
BiosSMBIOSMinorVersion                     :   6
BiosSMBIOSPresent                          :   True
BiosSoftwareElementState                   :   Running
BiosStatus                                 :   OK
BiosSystemBiosMajorVersion                 :   6
BiosSystemBiosMinorVersion                 :   6
BiosTargetOperatingSystem                  :   0
BiosVersion                                :   DELL   - 1
```

```
CsAdminPasswordStatus                    :   Unknown
CsAutomaticManagedPagefile               :   True
CsAutomaticResetBootOption               :   True
CsAutomaticResetCapability               :   True
CsBootOptionOnLimit                      :
CsBootOptionOnWatchDog                   :
CsBootROMSupported                       :   True
CsBootStatus                             :   {0, 0, 0, 38...}
CsBootupState                            :   Normal boot
CsCaption                                :   GARRETTGRSEVER1
CsChassisBootupState                     :   Safe
CsChassisSKUNumber                       :
CsCurrentTimeZone                        :   -240
CsDaylightInEffect                       :   True
CsDescription                            :   AT/AT COMPATIBLE
CsDNSHostName                            :   GarrettGRSever1
CsDomain                                 :   WORKGROUP
CsDomainRole                             :   StandaloneServer
CsEnableDaylightSavingsTime              :   True
CsFrontPanelResetStatus                  :   Unknown
CsHypervisorPresent                      :   True
CsInfraredSupported                      :   False
CsInitialLoadInfo                        :
CsInstallDate                            :
CsKeyboardPasswordStatus                 :   Unknown
CsLastLoadInfo                           :
CsManufacturer                           :   Dell Inc.
CsModel                                  :   PowerEdge R710
CsName                                   :   GARRETTGRSEVER1
CsNetworkAdapters                        :   {Ethernet, Ethernet 2,
                                             Ethernet 3, Ethernet
                                             4...}
CsNetworkServerModeEnabled               :   True
CsNumberOfLogicalProcessors              :   16
CsNumberOfProcessors                     :   2
CsProcessors                             :   {Intel(R) Xeon(R) CPU
                                              X5550 @ 2.67GHz, Intel
                                              Xeon (R) CPU X5550  @
                                              2.67GHz}
CsOEMStringArray                         :   {Dell System, 5[0000]}
CsPartOfDomain                           :   False
CsPauseAfterReset                        :   -1
CsPCSystemType                           :   EnterpriseServer
CsPCSystemTypeEx                         :   EnterpriseServer
CsPowerManagementCapabilities            :
CsPowerManagementSupported               :
CsPowerOnPasswordStatus                  :   Unknown
CsPowerState                             :   Unknown
CsPowerSupplyState                       :   Safe
```

```
CsPrimaryOwnerContact                       :
CsPrimaryOwnerName                          :   Windows User
CsResetCapability                           :   Other
CsResetCount                                :   -1
CsResetLimit                                :   -1
CsRoles                                     :   {LM_Workstation,
                                                LM_Server, NT,
                                                Server_NT}
CsStatus                                    :   OK
CsSupportContactDescription                 :
CsSystemFamily                              :
CsSystemSKUNumber                           :
CsSystemType                                :   x64-based PC
CsThermalState                              :   Safe
CsTotalPhysicalMemory                       :   68698849280
CsPhyicallyInstalledMemory                  :   67108864
CsUserName                                  :
                                          GARRETTGRSEVER1\Administrator
CsWakeUpType                                :   PowerSwitch
CsWorkgroup                                 :   WORKGROUP
OsName                                      :   Microsoft Windows
                                                Server 2019 Standard
OsType                                      :   WINNT
OsOperatingSystemSKU                        :   StandardServerEdition
OsVersion                                   :   10.0.17763
OsCSDVersion                                :
OsBuildNumber                               :   17763
OsHotFixes                                  :   {KB5003541,KB4512577,
                                                KB4535680,KB5003711...}
OsBootDevice                                :   \Device\HarddiskVolume2
OsSystemDevice                              :   \Device\HarddiskVolume4
OsSystemDirectory                           :   C:\Windows\system32
OsSystemDrive                               :   C:
OsWindowsDirectory                          :   C:\Windows
OsCountryCode                               :   1
OsCurrentTimeZone                           :   -240
OsLocaleID                                  :   0409
OsLocale                                    :   en-US
OsLocalDateTime                             :   8/1/2021 9:45:33 PM
OsLastBootUpTime                            :   7/29/2021 9:52:37 PM
OsUptime                                    :   2.23:52:56.1375857
OsBuildType                                 :   Multiprocessor Free
OsCodeSet                                   :   1252
OsDataExecutionPreventionAvailable          :   True
OsDataExecutionPrevention32BitApplications  :   True
OsDataExecutionPreventionDrivers            :   True
OsDataExecutionPreventionSupportPolicy      :   OptOut
OsDebug                                     :   False
OsDistributed                               :   False
```

```
OsEncryptionLevel                          :   256
OsForegroundApplicationBoost               :   Maximum
OsTotalVisibleMemorySize                   :   67088720
OsFreePhysicalMemory                       :   60381844
OsTotalVirtualMemorySize                   :   77050192
OsFreeVirtualMemory                        :   70605984
OsInUseVirtualMemory                       :   6444208
OsTotalSwapSpaceSize                       :
OsSizeStoredInPagingFiles                  :   9961472
OsFreeSpaceInPagingFiles                   :   9961472
OsPagingFiles                              :   {C:\pagefile.sys}
OsHardwareAbstractionLayer                 :   10.0.17763.2061
OsInstallDate                              :   7/26/2021 4:54:50 PM
OsManufacturer                             :   Microsoft Corporation
OsMaxNumberOfProcesses                     :   4294967295
OsMaxProcessMemorySize                     :   137438953344
OsMuiLanguages                             :   {en-US}
OsNumberOfLicensedUsers                    :   0
OsNumberOfProcesses                        :   116
OsNumberOfUsers                            :   1
OsOrganization                             :
OsArchitecture                             :   64-bit
OsLanguage                                 :   en-US
OsProductSuites                            :   {TerminalServices,
                                           TerminalServicesSingleSession}
OsOtherTypeDescription                     :
OsPAEEnabled                               :
OsPortableOperatingSystem                  :   False
OsPrimary                                  :   True
OsProductType                              :   Server
OsRegisteredUser                           :   Windows User
OsSerialNumber                             :   00429-00105-30759-AA595
OsServicePackMajorVersion                  :   0
OsServicePackMinorVersion                  :   0
OsStatus                                   :   OK
OsSuites                                   :   {TerminalServices,
                                           TrminalServicesSingleSession}
OsServerLevel                              :   FullServer
KeyboardLayout                             :   en-US
TimeZone                                   :   (UTC-05:00) Eastern
                                                Time (US & Canada
LogonServer                                :   \\GARRETTGRSEVER1
PowerPlatformRole                          :   EnterpriseServer
HyperVisorPresent                          :   True
HyperVRequirementDataExecutionPreventionAvailable
                                           :
HyperVRequirementSecondLevelAddressTranslation
                                           :
HyperVRequirementVirtualizationFirmwareEnabled
```

```
                                          :
HyperVRequirementVMMonitorModeExtensions  :
DeviceGuardSmartStatus                    :   Off
DeviceGuardRequiredSecurityProperties     :   {0}
DeviceGuardAvailableSecurityProperties    :
   {BaseVirtualizationSupport}
DeviceGuardSecurityServicesConfigured     :   {0}
DeviceGuardSecurityServicesRunning        :   {0}
DeviceGuardCodeIntegrityPolicyEnforcementStatus
                                          :
DeviceGuardUserModeCodeIntegrityPolicyEnforcementStatus
                                          :
```