Fall 2018

Custom Windows Patching Methodology - Comparative Analysis

Brent Michael Henderson

CUSTOM WINDOWS PATCHING METHODOLOGY – COMPARITIVE ANALYSIS

by

BRENT HENDERSON

(Under the Direction of CHRIS KADLEC)

ABSTRACT

Windows Server Update Services has been a common mainstay among organizations with a heavy footprint of Windows operating systems since it was originally released as Software Update Services in 2002. While the product has grown in scope, the primary allure remains the same: WSUS offers organizations greater control over the patches that are released to their environment and saves bandwidth by allowing a centralized device to download and offer patches to internal clients rather than having each of those clients download the content they require from the Internet. Unfortunately, the product has a structural limitation in that it lacks the capacity to provide high-availability to the metadata synchronization process that must occur in order to deliver the most up-to-date patches to endpoints. WSUS metadata contains details about the individual updates, EULAs, and supersedence relationships.

Due to design limitations and the growing concern of outages, a solution was developed to supplement and perhaps replace WSUS in certain scenarios. This solution, dubbed the Custom Patching Manager (CPM), is an extension of a concept originally started by Alejandro Gómez Galindo and finds middle-ground between Windows Server Update Services and Windows Update using freely available software. The solution assesses the vulnerabilities of a system or systems, determines whether or not the patches are part of an approved list, determines whether or not the content for missing updates is available locally, acquires that content depending on the previous step, and applies the patches to the endpoint. This proof-of-concept proved functional and reliable but would benefit from some optimizations that have been recommended as future works.

INDEX WORDS: Windows, Patch, Patching, System updates, Cybersecurity, Security, Windows Server Update Services, WSUS, PowerShell, Microsoft Baseline Security Analyzer

CUSTOM WINDOWS PATCHING METHODOLOGY - COMPARITIVE ANALYSIS

by

BRENT HENDERSON

B.B.A., Georgia Southern University, 2010

A Thesis Submitted to the Graduate Faculty of Georgia Southern University

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

CUSTOM WINDOWS PATCHING METHODOLOGY - COMPARITIVE ANALYSIS

by

BRENT HENDERSON

Major Professor:	Chris Kadlec
Committee:	Cheryl Aasheim,
	Adrian Gardner

Electronic Version Approved:
December 2018

DEDICATION

To Sarah, Cora, Mom, Dad, and P.C.

The trials and tribulations that were endured at basically every step of the creation of this document are known by and were overcome in large part due to a very short list of people. Your encouragement bolstered and reinvigorated my resolve in times of wavering confidence, overbearing exhaustion, and despair at the thought of potential outcomes that fortunately did not come to pass. It is my hope that this document may serve as a minor monument to our having collectively overcome those obstacles together.

I want to give thanks to my Mom and Dad, Brenda and Mike Henderson, for being my fiercest advocates. Your innumerable sacrifices and steadfast support led to me being the person I am today.

To Sarah, my best friend and my wife, I thank you as well. Your love and encouragement made it possible for us to navigate the incredibly difficult circumstances we have faced throughout the past few years.

I also owe thanks to my Grandpa, James Monroe Henderson. Memories of your frequent encouragement to strive for scholastic excellence are second in influence only to your examples of what a man can be to family, friends, acquaintances, and strangers. Your legacy will always endure in the hearts and minds of those of us who had the privilege of being a part of your life.

And to my daughter, Cora Elizabeth Henderson: your star burns brightest of all. You will always be a light for us in dark places where all other lights go out.

ACKNOWLEDGMENTS

I would be remiss if I did not begin by thanking my friend and former colleague, Brandon Kimmons, for going above and beyond in ensuring that I had an easily accessible lab environment during those late-night hours between midnight and 3AM that I had to myself while I was on one of my many Cora shifts. The overwhelming majority of the work happened while everyone else slept and the research that was conducted simply wouldn't have been possible without your efforts.

I would also like to thank Dr. Christopher Kadlec for putting up with the fact that this thesis has undergone several false-starts and a couple of topic changes. Your guidance and friendship are appreciated.

Finally, I would like to thank Dr. Cheryl Aasheim and Dr. Adrian Gardiner for agreeing to assist this endeavor on relatively short notice. I recognize that there are far more interesting things to read in one's leisure time so your assistance and commitment are truly appreciated.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

NOMENCLATURE AND REFERENCED TECHNOLOGIES

- **Active Directory Domain Services (Active Directory, AD)** – Active Directory Domain Services is a feature of the server variants of Windows operating systems that provides directory services to networked resources. As a directory service, it stores and controls access to a hierarchical collection of network objects and the metadata associated with those objects. One example of such a relationship is a user object and its underlying metadata in the form of the legal name, phone number, email, and password of that user.
- **Cabinet Files (CAB Files)** – CAB files are compressed archives of one or more underlying files that can assist in the expedient delivery of content between networked resources.
- **Endpoint** - Any server or workstation client in an IT infrastructure.
- **Hypervisor** - Hypervisor technology is what makes server virtualization possible, providing an abstraction layer between "guest" operating systems. This abstraction creates the potential for a one-to-many relationship in that one set of hardware can be simultaneously utilized by multiple operating systems. There are two types of hypervisor - Type 1 and Type 2. A Type 1 hypervisor is integrated at the "bare-metal" layer, meaning it sits directly atop the hardware layer with nothing between the two. In contrast, a Type 2 hypervisor sits atop a host operating system such as Windows, Linux, or MacOS.
- **Microsoft Baseline Security Analyzer (MBSA)** – The Microsoft Baseline Security Analyzer is a proprietary tool provided by the Microsoft Corporation to assist administrators with identifying, among other things, missing security updates on the endpoints upon which it is invoked or targeted toward.
- **Patching** - System patching is the process whereby an administrative or governing body applies amendments supplied from the vendor to software in order to alter functionality of the original source code. Within the context of this text, the term is referencing security-related patches.

- **PowerShell** - PowerShell is a command-line shell and scripting language built on top of the Microsoft .NET framework that is native to the Windows operating system.
- **Scripting** - A scripting language is typically a high-level programming language that is capable of being interpreted by a specific runtime environment without being compiled. There are two primary ways that scripting languages are used - "one-off" solutions to satisfy an immediate need or, for more complicated problems, "scripts" which encapsulate solutions to more demanding tasks. Depending on the size and sophistication of the task, scripts may contain many thousands of lines of code.
- **System Center Configuration Manager (SCCM)** - SCCM is a systems management suite that allows for the deployment of software, patches, operating systems, and enforced configurations to vast numbers of endpoints as required.
- **Virtualization** - Virtualization is the process whereby multiple instances of operating systems can be provided with access to a single underlying set of hardware resources.
- **Windows Server Update Services (WSUS)** - Windows Server Update Services is an administrative tool that allows for the synchronization, curation, and deployment of first-party updates from Microsoft to Windows endpoints.

CHAPTER 1 - INTRODUCTION

1.1 Problem Statement

Regardless of whether an organization is a small and midsized business or a global enterprise, the need to protect one's intellectual property and other sensitive data is tantamount as the burgeoning concerns of attack frequency, sophistication, and state sponsorship of intrusion methodologies worsen. Among the most readily addressable cybersecurity concerns is the presence and formalization of a routine patching strategy. As of 2018, the most commonly exploited vulnerabilities in the world that resulted in data breaches were enabled via vectors that the vendor had already released patches for (Higgins, 2018).

Organizations that leverage the modern Windows operating system have historically leveraged either Windows Server Update Services or Microsoft Update to deliver patches depending on the size, staffing, and overall capabilities of the organization's Information Technology team. More advanced organizations may leverage a configuration management framework such as System Center Configuration Manager to perform patching but are often still dependent upon WSUS either directly or indirectly without realizing it. Unfortunately, the reliability of WSUS has become a growing concern as outages have become more frequent. The synchronization mechanism is the greatest cause for concern as even "highly available" WSUS infrastructures ultimately depend on a single upstream server to synchronize metadata with Microsoft's update catalog. An organization may find itself without the ability to provide current updates if the synchronization process fails (Docs.microsoft.com, 2017). Given these challenges, one can reasonably posit that alternative methodologies are worth exploring and that a data-driven analysis should be conducted to ascertain the viability of such an effort relative to standard enterprise practices.

1.2 Background

The Windows operating system is no stranger to security vulnerabilities that must be regularly patched and Windows 10, the latest entry in that family of operating systems, is no exception (HelpNetSecurity, 2018). The underlying NT architecture, especially in the early 2000 consumer-level release such as Windows XP, created popular targets for malware due to a litany of questionable security decisions such as Administrative access on the default user, lack of guidance on password complexity guidance during the creation of the default Administrative account, lack of guidance to create a non-Administrative account for day-to-day usage, lack of onboard antivirus scanning capabilities, and firewall functionality being disabled by default. The SANS Institute, a prolific provider of cyber security research and certifications, released a document entitled "Windows XP: Surviving the First Day" to address these concerns due to proliferation and exploitation of these vulnerabilities that could affect an Internet-connected Windows XP device in less than half an hour (SANS Institute Internet Storm Center, 2003). Organizations needed greater control of the Windows patching process and Microsoft delivered that control in the form of Windows Server Update Services. Public disclosures of vulnerabilities are trending upward industry-wide and the Windows operating system holds a significant portion of market share in the consumer operating system space (Rains, 2018).

Organizations that have a heavy reliance on Windows generally have an equally heavy reliance upon Windows Server Update Services (WSUS) or Microsoft Update (Shields, 2016). With WSUS, this reliance can generally come as either a standalone deployment of WSUS or a hybrid deployment of WSUS.

In a standalone deployment, the endpoint patching of an organization is handled entirely from the WSUS instance. Systems Administrators utilize WSUS directly to perform tasks such as configuring proxy settings, selecting applicable Products and Classifications, synchronizing update metadata, downloading content, and configuring WSUS maintenance. In this configuration, WSUS

can be leveraged exclusively to manage the Microsoft first-party patching requirements of the organization provided that those products are currently supported (Appendix L).

A hybrid deployment is an indirect utilization of a WSUS instance to facilitate patching. Certain products such as System Center Configuration Manager leverage a WSUS instance to accomplish many of the same functions that a standalone deployment would do with the difference being that SCCM expands upon and obfuscates this native functionality by leveraging the mechanisms available within WSUS without ever allowing the user to directly interact with it. While this approach can be powerful, these configurations are hindered by the same limitations and maintenance demands of a standalone WSUS infrastructure.

1.3 Contribution to Knowledge

The two primary goals of this thesis are to ascertain the relative real-world impact of different Windows operating system patching methodologies within an enterprise and to analyze the relative efficacy of a custom solution that was developed without a dependence on Windows Server Update Services. The existence of an external mechanism for patching could have value as a backup patching plan in the event that Windows Server Update Services fail given WSUS itself lacks options for achieving true high-availability with regard to the synchronization of metadata that drives the process.

The research is comprised of 3 scenarios that represent an individual Windows patching methodology with the intention to reveal where the optimal utilization of each methodology resides. Furthermore, the thesis explores where the viability of the Custom Patch Mechanism (CPM) falls in relation to traditional patching methodologies as both a supplementary tool and as a potential replacement. The metrics that were captured during the research are as follows:

- Time to Compliance
- Endpoint CPU Utilization
- Network Utilization

1.4 Hypothesis

Windows Server Update Services is a common mainstay of businesses that support Windows-based devices but all supported designs for it are hindered by a single point of failure in the form of the top-level server that synchronizes and downloads metadata and content from Microsoft. If this top-level server encounters issues, downstream WSUS servers cannot serve the endpoints that are reliant upon that metadata to evaluate the applicability of updates.

Leveraging Windows Update is another possibility but exposes organizations to different problems in the form of excessive bandwidth consumption and a reduction in the ability to curate the patches. The viability of allowing this communication diminishes at scale not just because of the bandwidth consumption but also because of support concerns. For example, users whose endpoints were allowed to access to Windows Update in the first week of October 2018 may have downloaded the initial release of Windows 10 1809 that has been rescinded due to the deletion of user files (Bowden 2018).

The hypothesis being put forth is that there is a potential middle-ground that can satisfy the needs of an enterprise and circumvent the most significant limitations of either of the aforementioned methodologies detailed above. The proof-of-concept that was utilized for the purposes of this thesis was also created with the expectation that some businesses lack the funding for a Windows Server license that would be required to utilize WSUS. The Custom Patch Manager (CPM) proof-of-concept and subsequent management scripts developed for the purposes of collecting data provide an automated, repeatable, and expandable platform whereby an administrator can remotely invoke the process and simply await the completion of the process. CPU utilization and Time to Completion were measured to provide some insight into the relative efficiency of each approach.

CHAPTER 2 – LITERATURE REVIEW

2.1 Overview

This chapter begins with a brief note about the origins of operating system patching. The mechanisms that the Information Technology industry adopted in the past have increased in sophistication, scope, and usability but the underlying need that the aforementioned mechanisms addressed largely remain the same. Next, there is an overview of the advent of Windows Update and the current standard format for Windows patches themselves. Finally, the literary review concludes with an overview of the design, capabilities, and limitations of Windows Server Update Services as it is a common requirement for businesses with a heavy Windows footprint.

2.2 Origins of System Patching

The need to formally integrate changes to either fix previously submitted code or expand upon its functionality is nearly as old as the concept of computing itself. The programmers of previous generations were no less fallible in their efforts to operate within and expand the realm of computing and it is that fallibility that made the concept of system patching a natural inevitability.

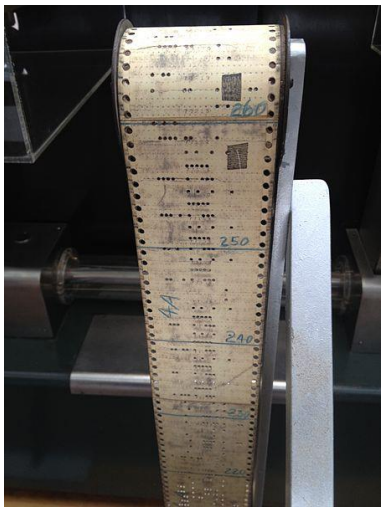


Figure 1: Harvard Mark 1 program tape with visible patches. Source: Arnold Reinhold, Wikimedia Commons

System updates, as the colloquial phrase ‘patch’ implies, originally came in the form of actual patches that were placed over specific blocks on punch card code to alter the functionality of the program that it contained. The delivery mechanisms of patches became more sophisticated alongside the industry itself, transitioning through periods of tape delivery, CD-ROM delivery, and ultimately arriving at the current standard of Internet-based delivery mechanisms.

2.3 Windows Update

Windows Update was launched alongside Windows 98 when it was released in May of 1998 and initially offered non-security items on an a la carte basis. Subsequent iterations grew to become more embedded in the operating system and more aggressively user-facing as the Internet became more heavily utilized as a vector for malware. This aggressive user interaction eventually culminated in what implementation of Automatic Updates in Windows XP. Greater flexibility and control were given via the Windows Update Agent that was paired with Windows Vista/Windows Server 2008 and continued into subsequent releases such as Windows 7/Server 2008 R2 and through to Windows 8.1/Server 2012 R2. The current trend that began with Windows 10 and Server 2016 is a movement away from providing users with fine-grained control of update delivery when Microsoft transitioned to the Windows as a Service model.

2.3.1 Advantages

The primary strength of Windows Update is that it is ubiquitous across all modern Windows offerings regardless of whether or not they’re consumer oriented, business oriented, or devoid of a graphical user interface. This ubiquity ensures that there is always an available avenue that can be leveraged in order to keep the operating system up-to-date with patches. Internet access is the core requirement for leveraging Windows Update.

2.3.2 Disadvantages

Organizations that opt to leverage Windows Update rather than other methodologies must contend with the ramifications of allowing every endpoint to download content directly from Microsoft. Smaller organizations may deem this a negligible disadvantage but organizations with fleets in the hundreds, thousands, or tens of thousands would understandably be wary of allowing every endpoint to download hundreds of megabytes of content every month. That said, newer Windows operating systems have the ability to allow for P2P sharing to local and remote resources that are connected via the Internet using a feature called Delivery Optimization. This feature is intended to lighten the bandwidth burden of larger updates such as feature updates and cumulative updates. Delivery Optimization is not available for operating systems that predate Windows 10/Server 2016.

The lack of fine-grained control in the selection of updates and the lack of an opportunity to perform rounds of quality assurance on updates that other organizations must consider if and when they decide to allow their endpoints to have access to Windows Update. In September and October of 2018, there have been two well-publicized issues in the form of the Windows 1809 release that deleted user data and the more recent KB4464330 issue that is creating blue screens of death in some HP and Dell systems (Claburn, 2018). Disruptions such as these would be problematic for any workforce but especially so for a mobile workforce that takes their laptops home from the office on a regular basis.

2.4 Windows Server Update Services

The trajectory of Windows patch management for businesses has remained relatively flat since the initial release of Software Update Services. The demands for fine-grained control over which updates are delivered, what endpoints the updates are delivered to, and the behavior presented to users as a result of that delivery emerged quickly as the need for system patching grew. In service of these growing requirements, Microsoft released Software Update Services (SUS) which was later renamed to Windows Server Update Services (WSUS). The core operating philosophy of the software remained the same in the transition from SUS to WSUS. These technologies functionally create an arbiter between intranet

clients and Windows Update. This arbiter synchronizes metadata with Windows Update and Microsoft Update, downloads the necessary update content to a local network share, and manages the delivery of the aforementioned content to clients that are configured to utilize that functionality. WSUS servers can be configured in a parent-child-grandchild relationship with one parent being capable of possessing multiple children and the children themselves being capable of possessing children. This delineation of service responsibility helps administrators prevent the resource exhaustion of any single WSUS server but does have structural limitations as Microsoft encourages administrators not to go more than 3 levels deep in their hierarchy guidance (Poggemeyer et al., 2018). The responsibility for the metadata synchronization with Microsoft's update servers cannot be shared and stands as a potential single point of failure.

2.4.1 Architecture and Management Style

WSUS infrastructure is relatively straightforward but warrants coverage as its simplicity is ultimately the hindrance that encourage me to pursue the exploration of a viable alternative. WSUS offers two management styles that can be individually or simultaneously utilized depending on the needs of an organization.

- Centralized: A centralized, multi-server hierarchy consists of one upstream server with subordinate WSUS replica instances. Computer groups and update approval decisions made at the upstream server flow down to downstream replica servers. Computer groups replicate to downstream servers but group membership does not so administrators must address that limitation by either automating the population of group membership or by manually populating the computer groups of each downstream replica server where applicable.

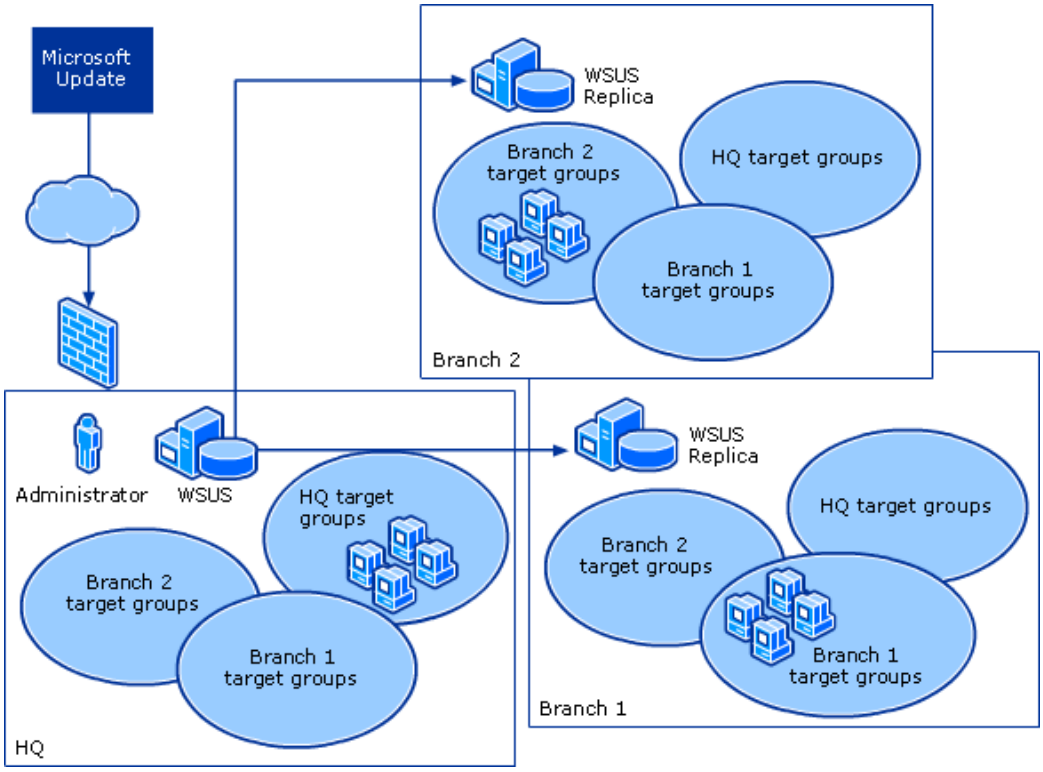


Figure 2: Centralized Management design. Source: Microsoft.com

- Distributed: The distributed, multi-server hierarchy model allows downstream servers to be fully configurable by administrators that are local to the branch that the server is located in.

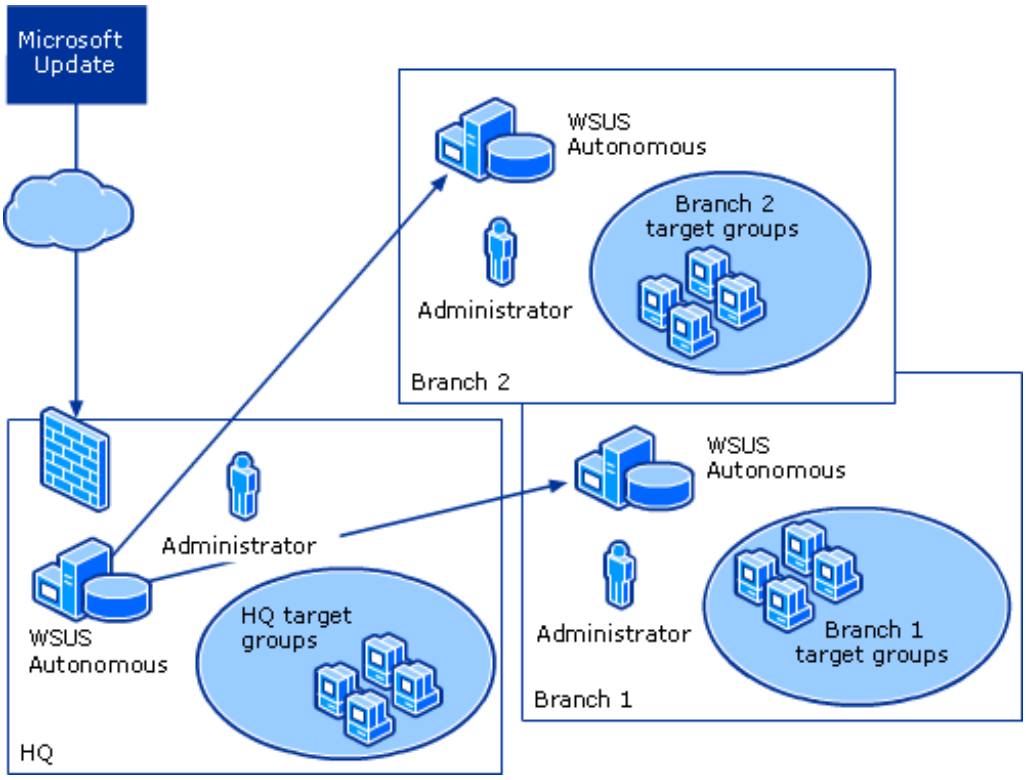


Figure 3: Distributed Management design. Source: Microsoft.com

2.4.2 Advantages

First and foremost, Windows Server Update Services is a simple technology to deploy even if one has not had the experience of managing it in the past. WSUS is available as a role in Windows Server operating systems and can be installed with the graphical user interface or via PowerShell. All that is required to configure a WSUS instance is a lightly provisioned Windows-based server with the following minimum specifications:

- Processor: 1.4 gigahertz (GHz) x64 processor (2 Ghz or faster is recommended)
- Memory: WSUS requires an additional 2 GB of RAM more than what is required by the server and all other services or software.
- Available disk space: 10 GB (40 GB or greater is recommended)
- Network adapter: 100 megabits per second (Mbps) or greater

- .NET Framework 4.0 or greater

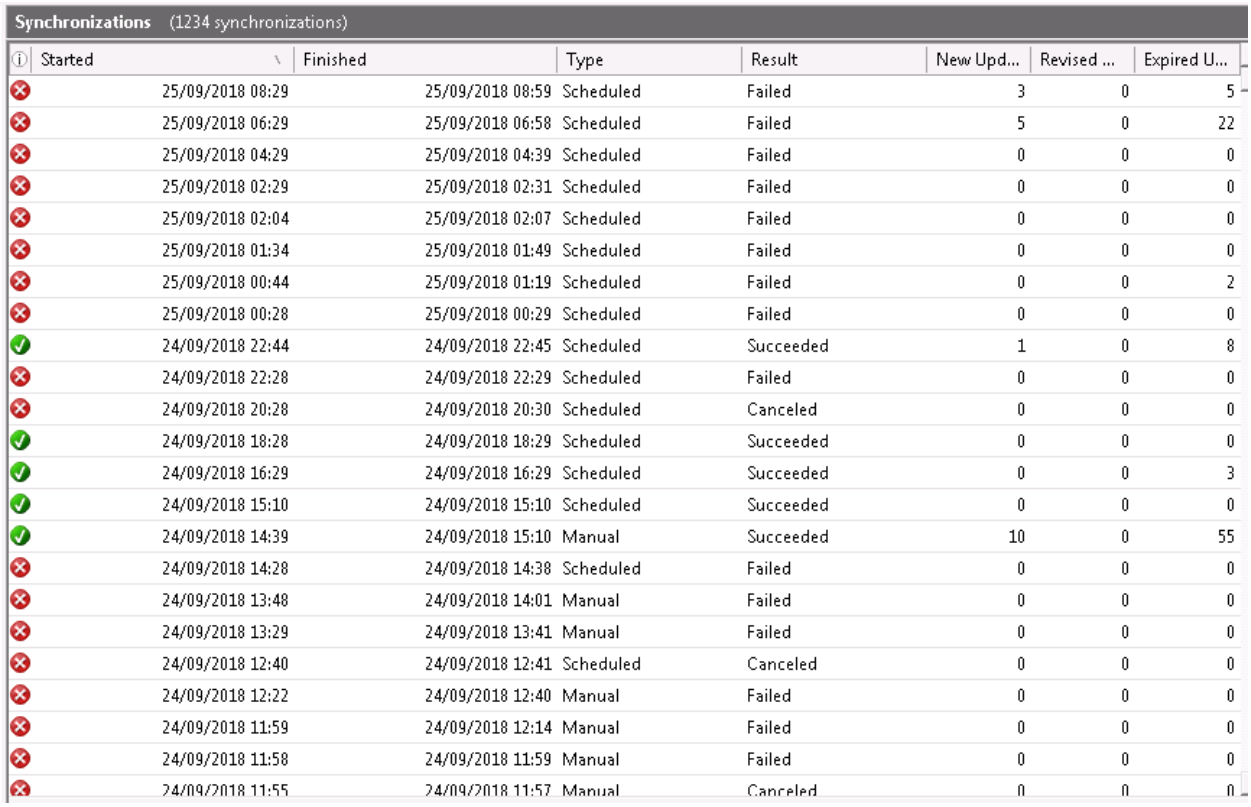
WSUS can be configured to utilize a Microsoft SQL Server database but does not require one as it natively supports the utilization of a Windows Internal Database.

The core purpose of WSUS is to provide software update services to organizations that utilize Microsoft products. To that end, the service provides broad support for hundreds of first-party products (Appendix L). The support list is too extensive to provide here but the list can be found in the appendix and ranges from products as recent as Server 2016 to products as old as Windows XP. This list was pulled directly from the WSUS instance that was utilized to create the data for this document.

The cost, usability, and product support list are the foundation of what has made Windows Server Update Services a common presence in Windows-based enterprises worldwide. Moreover, the adoption of WSUS provides a considerable savings in the form of bandwidth for content that would have otherwise needed to be delivered to each endpoint over the WAN. The formula for this reduction of bandwidth is realized as (N-1)*S with N representing the number of endpoints in an organization and S being the size of the update content that would otherwise have been downloaded. The actual implications of this formula will be explored further in the following chapter.

2.4.3 Disadvantages

Despite its many advantages, Windows Server Update Services has a flaw that has persisted since its inception. No matter what management style or architectural decisions you make, WSUS still suffers from the limitation that there may only be one top-level server in a WSUS hierarchy that can synchronize updates with Microsoft. As such, this synchronization process becomes a single point of failure that can and frequently does run into issues. This problem is a familiar one for any administrator who has needed to administer the product.



Synchronizations (1234 synchronizations)

Started	Finished	Type	Result	New Upd...	Revised ...	Expired U...
25/09/2018 08:29	25/09/2018 08:59	Scheduled	Failed	3	0	5
25/09/2018 06:29	25/09/2018 06:58	Scheduled	Failed	5	0	22
25/09/2018 04:29	25/09/2018 04:39	Scheduled	Failed	0	0	0
25/09/2018 02:29	25/09/2018 02:31	Scheduled	Failed	0	0	0
25/09/2018 02:04	25/09/2018 02:07	Scheduled	Failed	0	0	0
25/09/2018 01:34	25/09/2018 01:49	Scheduled	Failed	0	0	0
25/09/2018 00:44	25/09/2018 01:19	Scheduled	Failed	0	0	2
25/09/2018 00:28	25/09/2018 00:29	Scheduled	Failed	0	0	0
24/09/2018 22:44	24/09/2018 22:45	Scheduled	Succeeded	1	0	8
24/09/2018 22:28	24/09/2018 22:29	Scheduled	Failed	0	0	0
24/09/2018 20:28	24/09/2018 20:30	Scheduled	Canceled	0	0	0
24/09/2018 18:28	24/09/2018 18:29	Scheduled	Succeeded	0	0	0
24/09/2018 16:29	24/09/2018 16:29	Scheduled	Succeeded	0	0	3
24/09/2018 15:10	24/09/2018 15:10	Scheduled	Succeeded	0	0	0
24/09/2018 14:39	24/09/2018 15:10	Manual	Succeeded	10	0	55
24/09/2018 14:28	24/09/2018 14:38	Scheduled	Failed	0	0	0
24/09/2018 13:48	24/09/2018 14:01	Manual	Failed	0	0	0
24/09/2018 13:29	24/09/2018 13:41	Manual	Failed	0	0	0
24/09/2018 12:40	24/09/2018 12:41	Scheduled	Canceled	0	0	0
24/09/2018 12:22	24/09/2018 12:40	Manual	Failed	0	0	0
24/09/2018 11:59	24/09/2018 12:14	Manual	Failed	0	0	0
24/09/2018 11:58	24/09/2018 11:59	Manual	Failed	0	0	0
24/09/2018 11:55	24/09/2018 11:57	Manual	Canceled	0	0	0

Figure 4: WSUS synchronization errors Source: reddit.com

The list of potential causes for a WSUS synchronization failure are too numerous and varied to cover within the scope of this thesis but there is a recent example that underscores the sensitivity of the product. In early September of 2018, synchronization errors were widely reported among websites that cater to administrators of Windows updates (Leonhard, 2018).

There were two rounds of synchronization problems in September 2018 that were reflected via feedback to this particular blog. The first problem began in early September when malformed metadata in a non-security update for Microsoft Office broke the synchronization mechanism of WSUS (Leonhard, 2018). There was no consensus as to a manual fix for the problem and some administrators found themselves waiting idly for Microsoft to correct whatever had ultimately caused the problem in the first place. The second such synchronization issue began around the third week of September of 2018 with similar confusion and a lack of consensus on what steps administrators could take to remediate the broken

process (Leonhard, 2018). The aforementioned metadata synchronization issues eventually resolved on their own within a couple of days according to the discussions that were associated with those articles.

The second most prominent issue with the product stems from a lack of transparency in the disclosure, support, and remediation of the aforementioned problems. This thesis does not cover the September 2018 synchronization issues in great detail because Microsoft has not publicly acknowledged or explained them. The WSUS Product Team Blog is directly maintained by the team that develops, releases, and supports the product and their last article was released in March of 2018. This lack of transparency leaves many organizations at the mercy of others who have enterprise support agreements with Microsoft and can leak private communications and recommendations to the general public. For a feature that is as critical to the core maintenance of the Windows operating system as WSUS is, one could be forgiven for finding this level of support to be unacceptable. For comparison, another prominent problem arose in Microsoft's ecosystem in September whereby upgrades to Windows 10 1809 resulted in the deletion of user files. Microsoft publicly acknowledged the error, pulled the version from all distribution channels, corrected the problem, validated the problem internally, and rereleased the update to their early-release customers (Cable, 2018). More importantly, Microsoft publicly posted a detailed technical explanation for what happened and what cues they utilized to see where it originated.

2.5 Structural Changes to Windows Updates

The current state of the updates that are delivered by the aforementioned mechanisms is of interest as well because Microsoft made a significant departure from decades-long practices in 2016. Prior to Fall of 2016, Microsoft followed a predictable pattern of releasing individual patches for Windows on a monthly cycle that is referred to as Patch Tuesday. On the 2nd Tuesday of each month, Microsoft would publish the individual updates for each product they support and supersede or expire older updates as necessary. Given the ever-expanding state of discovered vulnerabilities and Microsoft's ever-expanding product portfolio, this list of updates grew to numbers that became unwieldy for

administrators to handle. In an effort to simply this process, Microsoft introduced two new update mechanisms for Windows administrators to utilize.

In Fall of 2016, Microsoft transitioned to an update aggregation model via two distinct mechanisms. The Security-Only update provides the current month's security fixes for the product it is applicable to, serving as an ongoing delta that can be applied by organizations that have otherwise kept their infrastructure up to date. The Monthly Rollup update contain all security and non-security patches released since the introduction of the new model. With this update, administrators can utilize a single update to get a system current if it has been patched since the introduction of this new model in 2016.

The aggregation of updates in this fashion has simplified patching in some respects while complicating it in others. At face value, the idea behind the transition is admirable but has been a source of frustration due to the all-or-none approach. The overall reliability of Microsoft patches has come into question due to internal shifts in the way that Microsoft handles the regression testing of updates and OS releases. Industry leaders have taken note and have begun to speak out about the increasing frequency of these issues and are now going so far as to poll administrators for feedback. Susan Bradley, a moderator for the prominent PatchManagement.org mailing list, polled their users and sent a formal letter to Microsoft in an attempt to raise awareness and compel them to address the increasing frequency of problems that are making their way to production environments worldwide (Keizer 2018). In the feedback that Bradley gathered, 69% of respondents indicated that they were either "not satisfied" or "very much not satisfied" in the quality of Windows updates and servicing that had been foisted upon the industry by Microsoft.

CHAPTER 3 - METHOD

3.1 Custom Patch Manager: Finding Middle Ground

The Custom Patch Manager was developed with four primary objectives in mind. First and foremost, the CPM was designed to patch Windows operating systems without being dependent upon the metadata synchronization process that can prevent Windows Server Update Services from acquiring current patch data and content from Microsoft. In addition to circumventing the WSUS dependency, the CPM was written to download content once and share it with other resources in order to avoid wasting bandwidth and potentially exhausting WAN resources. The third objective was to ensure that the CPM can whitelist or blacklist particular patches to conform to the individual needs of an organization. The fourth and final objective was to ensure that the CPM was capable of being invoked via command-line so that it could be integrated into other automation mechanisms and executed remotely at the administrator's discretion.

In the initial pursuit of these objectives, my research led me to a promising body of work created by Alejandro Gómez Galindo. The script that was published on his blog was broken in some areas, dependent upon deprecated functionality from older Windows operating systems, and in need of some additional functionality in order to meet the objectives detailed above but was an excellent foundation to build upon. I discussed the matter with Alejandro and have included his unabridged work as Appendix K so as to provide proper attribution and so that it may be referenced against the modifications that were made to it.

The testing of the CPM, Windows Update, and Windows Server Update Services series was conducted in a simulated environment that consisted of 30 endpoints that were updated to be current as of September 2018 with the exception of the two updates that each method would be installing. Three snapshots were created for each virtual machine so that they could be reset to the proper configuration for the upcoming method simulation that they were intended to execute.

The research revolved around three unique series of Windows update methodologies: Windows Update, Windows Server Update Services, and the Custom Patch Manager. Five iterations were executed for every series and each iteration was comprised of 30 endpoints. The goal of every series was to install two specific updates:

- 2018-09 Security Monthly Quality Rollup for Windows Server 2012 R2 for x64-based Systems (KB4457129, **352.6 MB**)
- Microsoft .NET Framework 4.7.2 for Windows Server 2012 R2 for x64 (KB4054566, **68.6 MB**)

The updates listed above were selected to serve as a least-common denominator for vulnerabilities in Windows-based environments. The Security Monthly Quality Rollup released every month contains the security patches for that month alongside the fixes from previous months. The Microsoft .NET Framework patch was included because that particular technology cannot be fully uninstalled from the Windows operating system. These two patches collectively represented 421.2 MB of data that needed to be distributed and installed on very endpoint.

3.1.1 Prerequisite Evaluation

The first stage of the Custom Patch Manager is to determine if the prerequisites needed for successful operation are locally available to the device and, if not, download the prerequisites from a network share. The prerequisites must be available in at least one of the two places or the solution will not function properly. These prerequisites are detailed below:

- Wsusscn2.cab: This cab file serves as an offline substitute for WSUS and contains metadata for security updates, update rollups, and service packs.
- Mbsacli.exe: The Microsoft Baseline Security Analyzer (MBSA) is a free tool distributed and maintained by Microsoft Corporation to help administrators assess vulnerabilities and insecure settings in their environment. The mbsacli.exe component allows the execution of certain options via command line and without the amount of dependencies that the graphical user interface has.

- Wusscan.dll: This dll is a module that is provided as part of the MBSA installation and is required for the mbsacli.exe to evaluate the vulnerabilities on endpoints.

3.1.2 WSUS Offline CAB Validation (Optional)

The CAB must be downloaded each month in order to ensure that the metadata that it contains is up to date. To that end, there is an optional step whereby the CAB age can be validated and the latest CAB can be downloaded if the one that is locally available is older than 15 days.

3.1.3 Vulnerability Assessment

After ensuring that the prerequisites are met and that the WSUS Offline CAB are valid, the Mbsacli.exe program is executed in conjunction with the CAB in order to determine the current security state of the machine. Upon completion, the executable will generate an XML file that provides the details of any missing patches and the URLs that can be sourced to download their respective content. Alternately, the script can be instructed to import previously generated XML to prevent an unnecessary evaluation from taking place.

3.1.4 Patch Content Acquisition

Once the XML with the vulnerability posture has been created, the CPM parses the update metadata and determines if the patch is allowed or disallowed. If the update is allowed and missing, the CPM then scans a local network share for the requisite content. The CPM will copy the content to the local C:\ drive if it is available on the local network share or it will download the file from the Internet and then create a copy on the network share so that the content does not have to be downloaded by other endpoints that require it.

3.1.5 Patch Installation

After the content acquisition has been completed, the CPM transitions into the installation phase. Windows Updates can come in the form of MSU files, EXE files, or CAB files. The CPM gathers a list

of all files to be installed and then installs the update with the mechanism that is appropriate for the file format that it was provided in.

3.2 Utilization

3.2.1 Series Breakdown

The research revolved around three distinct series representing different patching methodologies. Each series was invoked as an iteration and every iteration was invoked five times on all 30 endpoints. Each of these series methods are detailed below:

- wu_series: This series is the Windows Update series. Every iteration of this series automatically configured and initialized the endpoints in a way that resulted in each endpoint reaching out to Windows Update, downloading the updates from Microsoft, and installing the updates in an automated fashion.
- wsus_series: This series is the Windows Server Update Services series. Every iteration of this series automatically configured and initialized the endpoints reaching out to a local WSUS instance, copying the required update files across the WAN, and installing the updates in an automated fashion.
- cpm_series: This series is the Custom Patch Management series. Every iteration of this series automatically configured and initialized the Custom Patch Management automation. This automation will be detailed further along in this chapter.

The CPM was designed in such a way that it could be invoked by external automation. The external automation that was written to drive the research came in the form of 10 PowerShell functions of varying purposes that are detailed below:

1. Start-SeriesManager: This function served as the catalyst for all of the subordinate automation that made each iteration of a series run possible. The series manager would make calls to connect to the Netlab vSphere instance, reset the environment by reverting all virtual machines to the snapshots that were appropriate for the desired series, power on the virtual machines, monitor for

successful Windows startup, invoke an iteration of the specified series, monitor for completion, and then begin another iteration for as many iterations as it was instructed to perform (Appendix A).

2. Connect-LabVIServer: Ensured that connectivity to the VMware vSphere server was established and maintained for the proper execution of all subordinate automation (Appendix B).
3. Revert-LabToSnapshot: Reverted the virtual machines to snapshots that were created to appropriately fit the desired circumstances and configurations of each series (Appendix C).
4. Start-LabMachines: Powered on the lab machines for each iteration (Appendix D).
5. Invoke-LabCommand: Invoked a command in the guest operating system that it was instructed to target (Appendix G).
6. Start-LabTests: Managed the specific invocation of every iteration on the endpoints, parsing input passed from Start-SeriesManager and carrying out the specific series that was requested on the endpoints it was instructed to target (Appendix E).
7. Invoke-LabCPM: Kicked off the Custom Patch Manager on endpoints. The functionality will be detailed later in this chapter (Appendix I).
8. Start-PowerCLIJob: A community script that allowed for the parallel invocation of vSphere jobs via PowerCLI such as reversions to snapshots (Appendix H).
9. Stop-LabMachines: Powered down the lab endpoints in a controlled fashion (Appendix F).
10. Upload-LabData: Copied lab data from the network share to OneDrive (Appendix K).

3.2.2 Invoke-LabCPM Function Parameters

The Invoke-LabCPM function accepts a variety of parameters that can change the behavior depending on the preferences of the user. The parameters that may be passed to this function are as follows:

- SourceFolder: This is the source folder for the mandatory prerequisite files. This folder defaults to C:\temp.

- Iteration: Designates the current iteration of the cpm_series that's being ran so that logs may be stored with the appropriate name.
- UseNetwork: Indicates whether or not the network share should be searched for matching updates prior to downloading them from Microsoft.
- NetworkSource: Designates the path of the network share to be searched when used in conjunction with the UseNetwork parameter.
- ValidateCab: When invoked, this parameter evaluates the age of the CAB and downloads another if it is out of date by more than 15 days.
- Download: Indicates whether or not updates should be downloaded. Without this parameter, the function will generate XML output of missing updates and go no further.
- HomeTest: This parameter was utilized to designate what pathing to use for a network share as the one at home was lengthy and obnoxious to look up.
- PriorXMLPath: If XML has been generated by a previous run, it may be imported with this parameter to save time and forego the execution of the MBSA evaluation.

3.3 Lab Technology

The lab environment that was utilized to conduct the research, development, and testing of all scenarios utilized the same overarching infrastructure. The details of the infrastructure technologies that were employed are as follows:

- Netlab: NetDevGroup (NDG) partners with campuses worldwide to provide easy access to fully containerized, hands-on lab environments so that students may learn within a realistic environment that they can administer themselves. The Netlab environment utilized for this research was tailored for VMware and EMC technologies due to the educational offerings of Georgia Southern University.
- VMware technologies: VMware is an industry-leader in the virtualization space and the lab environment was based upon their offerings.

- ESXi: A type-1 hypervisor that runs directly atop hardware and abstracts those resources in such a way that they can be leveraged by many "guest" operating systems.
- vCenter Server: The vCenter Server suite is a central management system that can manage one or many ESXi "host" servers and enhance them with additional functionality such as vMotion (Move a virtual machine between hosts), storage vMotion (Move a virtual machine's file system from one storage system to another), high-availability (Automatically move a virtual machine to one host if another fails), fault-tolerance (Maintain uninterrupted service of a virtual machine if a host fails), and more.
- PowerCLI: This technology is a module built in PowerShell that allows users of that scripting language to manage VMware technologies.

3.4 Lab Specifications

- Client Specifications:
  - OS: Server 2012 R2
  - CPU: Dual-Core CPU
  - 4GB RAM
- Active Directory Server Specifications
  - OS: Server 2012 R2
  - CPU: Dual-Core CPU
  - 4GB RAM
- WSUS Server Specifications
  - OS: Server 2012 R2
  - CPU: Dual-Core CPU
  - 4GB RAM
- VMware Host Specifications
  - OS: VMware ESXi 5.5
  - CPU: 8-core Intel Xeons (16 logical processors)

- 192GB of RAM

3.5 Advantages

The CPM borrows a number of advantages from both of the standard enterprise patching solutions. Like Windows Update, the proposed solution has no external dependencies that change from month to month aside from the CAB file that is updated on a monthly basis. If a system using CPM must download an update, it copies that update to the network share so that other endpoints may source it locally rather than also having to utilize WAN bandwidth. In addition, CPM mirrors functionality found in WSUS in that it can be used to blacklist updates, whitelist updates, and source updates from the network rather than downloading them from the Internet. The blacklisting and whitelisting functionality are hardcoded into the script as it was written so future works would be required to make this functionality dynamic and useful outside of the context of this proof-of-concept.

3.7 Disadvantages

The largest disadvantage comes in the form of limitations to the WSUS offline CAB itself as it only provides metadata for security updates, update rollups, and service packs. Critical updates and non-security updates do not fit into those classifications so those patches cannot be deployed with this solution.

The dependence upon the Microsoft Baseline Security Analyzer can also be considered a potential disadvantage. While the MBSA is free and publicly available from Microsoft, there is nothing to prevent them from assigning a price to this software or discontinuing the utility altogether. Future works would include circumventing this mechanism altogether in favor of a PowerShell-based solution.

CHAPTER 4 – PERFORMANCE ANALYSIS

The data collection that was undertaken as part of this research took place between October 4, 2018 and October 7th, 2018 during off-hours when the NDG lab environment was not being utilized by other students. Every endpoint for every iteration of each series was reset to a snapshot that had been captured with the appropriate configuration for the applicable series. Prior to the beginning of each iteration, every endpoint was powered on fresh from the applicable snapshot and the monitoring portion of the research automation would wait until every endpoint was responsive before initializing the desired series of testing. Each iteration of each series was executed in isolation so as to avoid resource contention of the host servers from skewing the results of the guest virtual machines.

There were three incomplete endpoints found throughout the entirety of the dataset and they are as follows:

- cpm_series, Iteration 2, Client 4
- cpm_series, Iteration 2, Client 13
- wu_series, Iteration 4, Client 25

The problem was found to have been environmental for each of these incomplete endpoints. The NDG lab operates based on reservations that cannot be programmatically extended and those reservations ended prior to full completion in two iterations that contained the incomplete endpoints.

One other detail of note that should be considered while evaluating the upcoming results is the quality of connectivity. The Time to Compliance metric on the Windows Update methodology is more sensitive to the Internet connectivity of the endpoints than any of the other methodologies due to each endpoint having to download the content on an individual basis. While the specific connection driving the NDG lab is unknown as of this writing, several speed tests were undertaken via speedtest.net that showed results of 250+ Mbps down and 150+ Mbps up. The United States has an average fixed Internet speed of 101.95 Mbps according to the Speedtest Global Index. The efficacy of the Windows Update methodology will also be affected by the number of endpoints: while 30 endpoints was not enough to

exhaust the network resources of the research lab, that observation will not hold true in real-world scenarios of hundreds of devices on a slower connection.

4.2 Time to Compliance

Time to Compliance is a simple measurement of the time between the invocations of a method on an endpoint to the time that the endpoint reported completion. The automation was not invoked until the endpoint had completed its Windows startup processes and was responsive to the automation that orchestrated the research.

The following chart is an overview of the averages of the individual endpoint outcomes for all iterations of each method and provides an overall look at the trends that each method took over the course of the research.

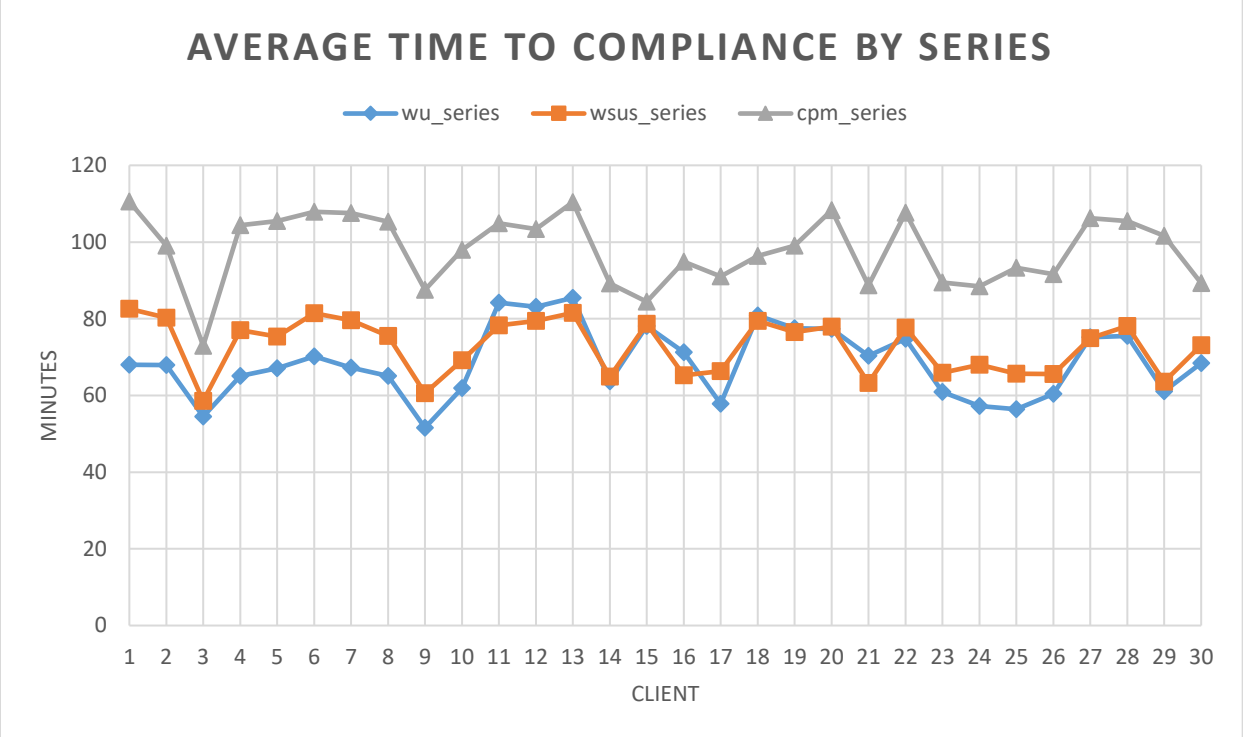


Figure 5: Average Time to Compliance by Series

4.2.2 Iteration 1 TTC

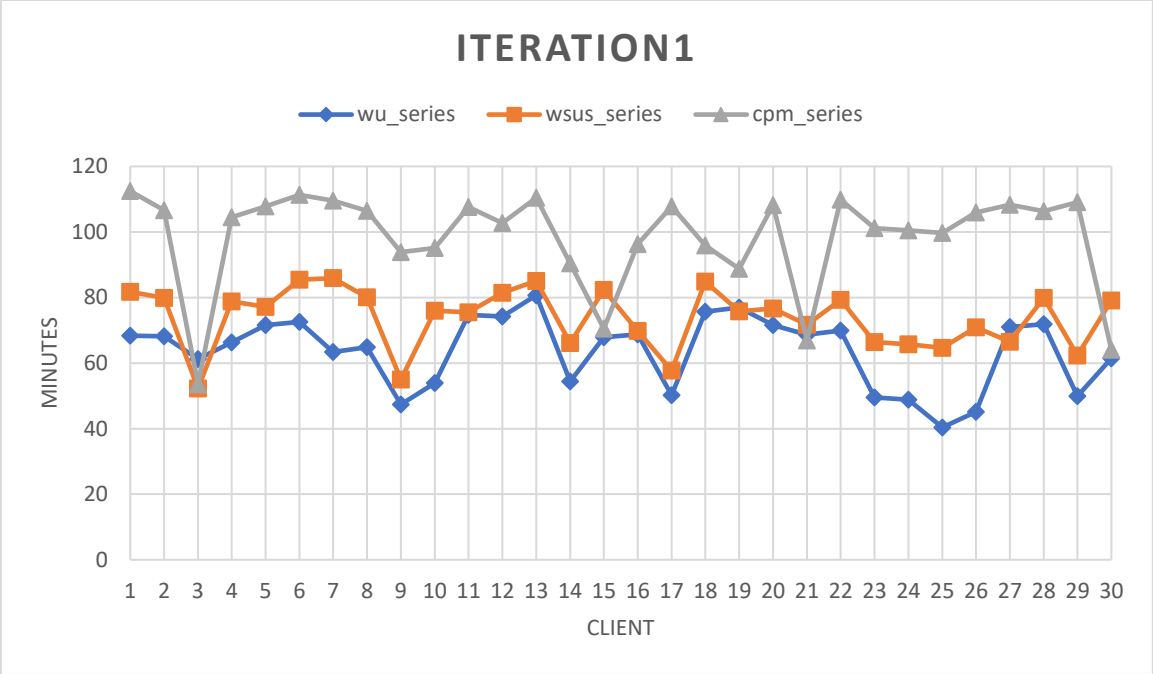


Figure 6: Iteration 1 Completion Results

Series	Iteration	Minimum	Maximum	Range	Mean	Median	Standard Deviation	Samples
wu_series	1	40.38	80.7	40.32	63.65	68.005	10.93	30
wsus_series	1	52.27	85.9	33.63	73.79	76.325	9.29	30
cpm_series	1	53.53	112.47	58.94	98.38	105.21	15.34	30

Table 1: Iteration 1 completion statistics

The Windows Update series possessed the fastest single data point with Client 25 completing its patch cycle in 40.38 minutes. However, the WSUS series proved to be slightly more consistent overall with a Standard Deviation of 9.29 minutes. On the opposite end of the spectrum, the Custom Patch Manager had the least consistency as demonstrated with a Standard Deviation of 15.34 minutes and proved to be the slowest solution on average with a mean completion of 98.38 minutes and a range of 58.94 minutes between the quickest and slowest endpoint patching duration. Table 1 serves as a reference for the statistical metrics of all series data that was collected throughout iteration 1.

4.2.3 Iteration 2 TTC

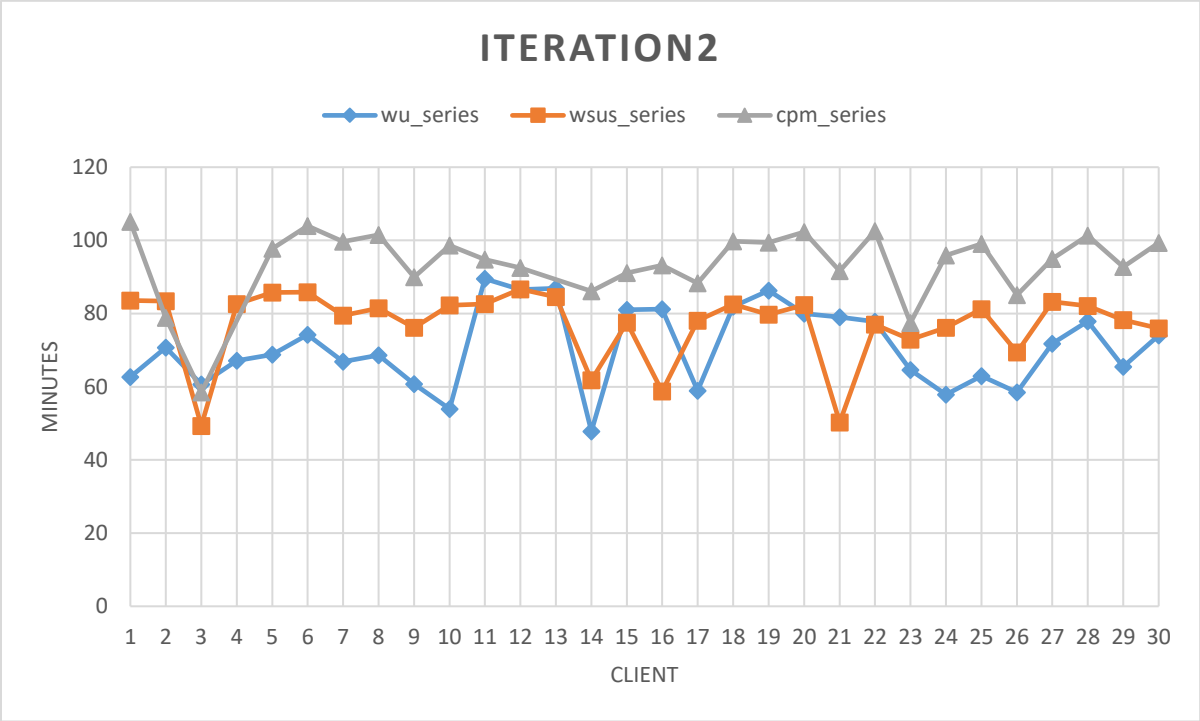


Figure 7: Iteration 2 Completion Results

Series	Iteration	Minimum	Maximum	Range	Mean	Median	Standard Deviation	Samples
wu_series	2	47.82	89.52	41.7	70.82	69.765	10.86	30
wsus_series	2	49.33	86.68	37.35	77.04	80.465	9.79	30
cpm_series	2	58.47	105.1	46.63	93.62	95.41	9.88	28

Table 2: Iteration 2 completion statistics

The disparity between each series was notably smaller with the most significant overall improvement coming from the Custom Patch Manager series results. The Custom Patch Manager series demonstrated a 5.46-minute improvement in the Standard Deviation metric. Nevertheless, the Windows Update series still narrowly possessed both the fastest individual completion time of 47.82 minutes and the lowest mean time to compliance with 70.82 minutes. The Windows Update series narrowly proved to be the least consistent with a Standard Deviation of 10.86 minutes. The Custom Patch Manager again proved to be the slowest solution in possessing the highest minimum TTC, the highest maximum TTC,

the highest mean TTC, and the highest median TTC. Table 2 serves as a reference for the statistical metrics of all series data that was collected throughout iteration 2.

4.2.4 Iteration 3 TTC

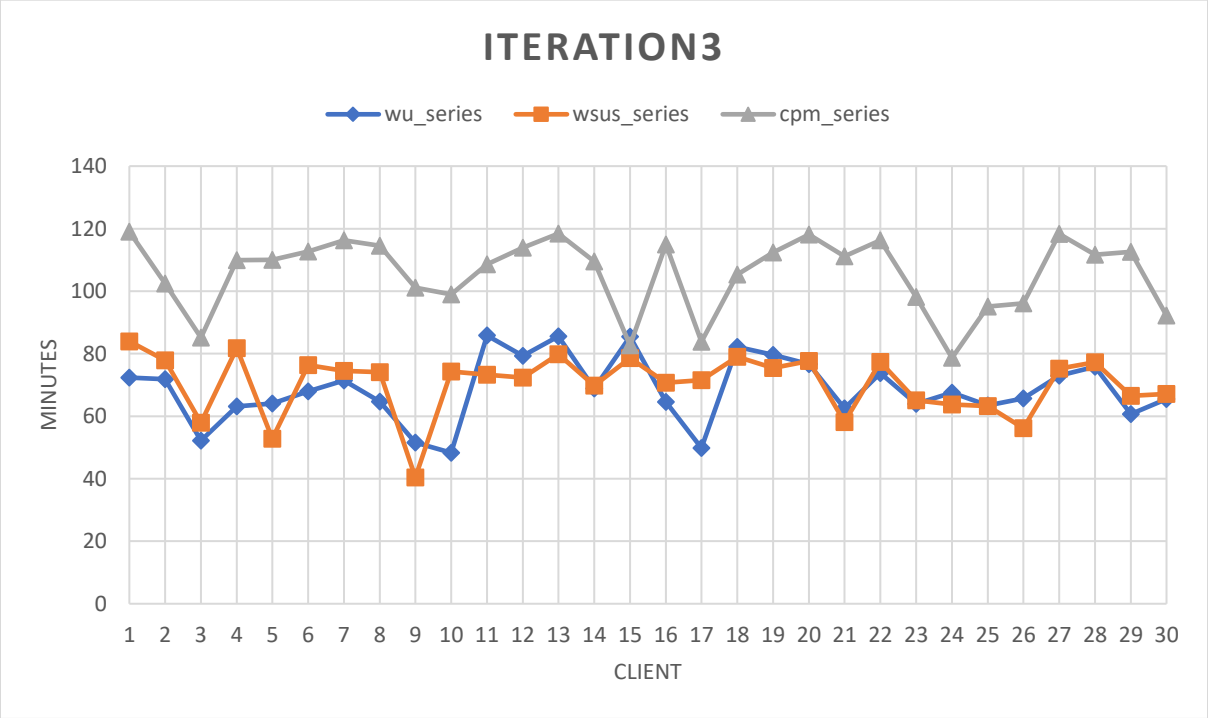


Figure 8: Iteration 3 Completion Results

Series	Iteration	Minimum	Maximum	Range	Mean	Median	Standard Deviation	Samples
wu_series	3	48.32	85.82	37.5	68.58	67.75	10.27	30
wsus_series	3	40.43	83.95	43.52	70.40	73.66	9.75	30
cpm_series	3	78.65	119.03	40.38	105.60	110	11.82	30

Table 3: Iteration 3 completion statistics

Iteration 3 demonstrates the first time during testing that the Windows Update series did not possess the fastest individual completion, losing out on that particular distinction by a margin of 7.89 minutes to Client 9 during its run of the WSUS series. The WSUS series of iteration 3 also had the distinction of possessing the lowest maximum TTC of any individual endpoint. As was the case with previous iterations, the Custom Patch Manager featured the most consistently lengthy metrics in all categories excluding the range between minimum and maximum times to completion. The delta between

the mean times to completion of the other series and CPM during iteration 3 was the most exaggerated of all iterations at 35.2 minutes (WSUS) and 37.02 minutes (Windows Update) respectively. Table 3 serves as a reference for the statistical metrics of all series data that was collected throughout iteration 3.

4.2.5 Iteration 4 TTC

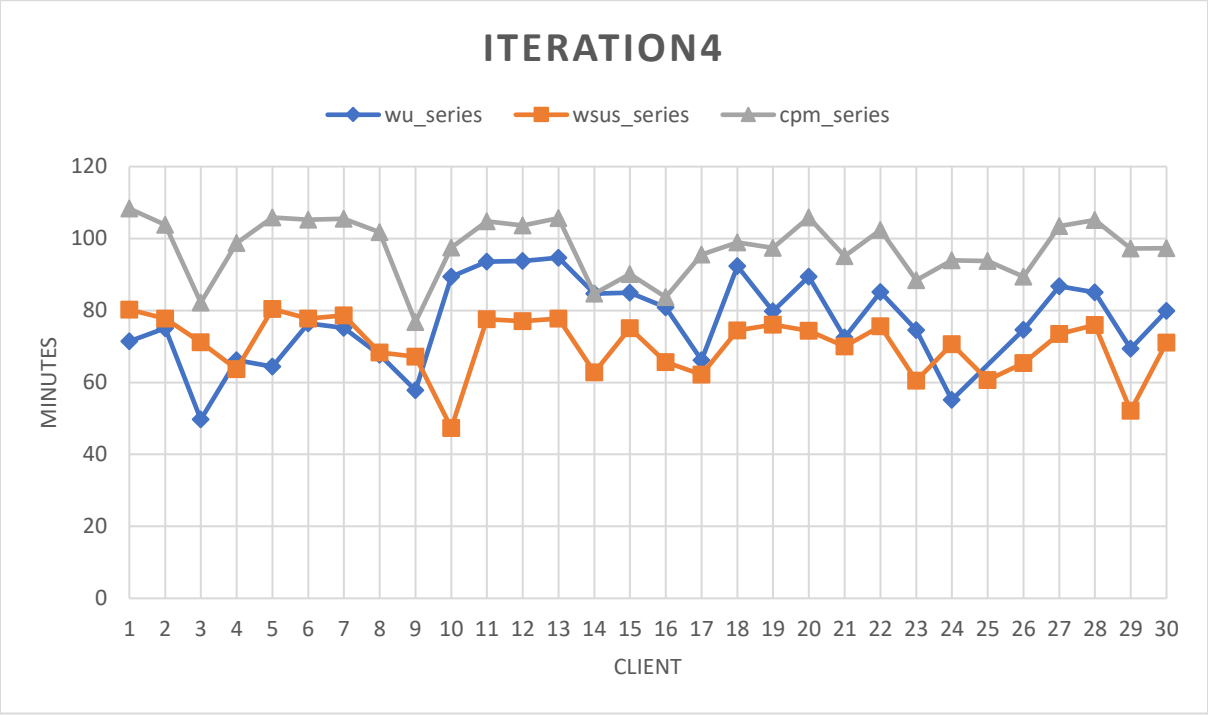


Figure 9: Iteration 4 Completion Results

Series	Iteration	Minimum	Maximum	Range	Mean	Median	Standard Deviation	Samples
wu_series	4	49.7	94.67	44.97	77.13	76.45	11.93	29
wsus_series	4	47.33	80.4	33.07	70.36	72.325	8.23	30
cpm_series	4	76.8	108.32	31.52	97.41	98.115	8.24	30

Table 4: Iteration 4 completion statistics

Iteration 4 possessed the narrowest gap between two standard deviations in any iteration with the WSUS series coming in at 8.23 minutes and the CPM series measuring out to 8.24 minutes. On that note, the standard deviation of 8.24 minutes shown by the Custom Patch Manager was the best result of that metric that the methodology earned across all iterations. The WSUS series once again had the distinction of holding the quickest single time to completion with Client 10 finishing in 47.33 minutes. Windows

Update was the least consistent during iteration 4, possessing a standard deviation of 11.93 minutes. Table 4 serves as a reference for the statistical metrics of all series data that was collected throughout Iteration 4.

4.2.6 Iteration 5 TTC

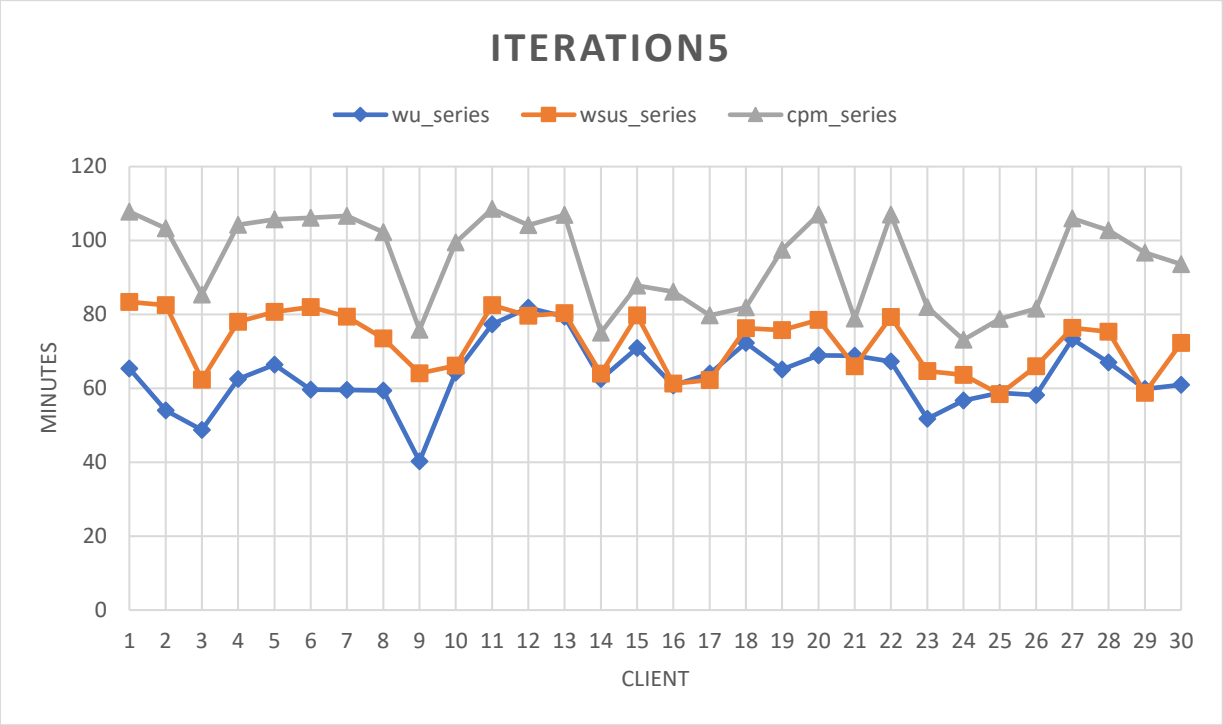


Figure 10: Iteration 5 Completion Results

Series	Iteration	Minimum	Maximum	Range	Mean	Median	Standard Deviation	Samples
wu_series	5	40.25	81.8	41.55	63.52	63.26	8.84	30
wsus_series	5	58.42	83.37	24.95	72.44	75.54	8.27	30
cpm_series	5	73.12	108.55	35.43	94.40	98.48	12.30	30

Table 5: Iteration 5 completion statistics

The final iteration was consistent with the overall trends shown previously in that Windows Update possessed the single fastest data point while WSUS proved to be slightly more consistent due to a Standard Deviation of 8.27. The range of values for the WSUS series was the smallest in iteration 2 of any iteration in the research, coming in at 24.95 minutes. The CPM series remained the slowest in most respects but did notably have a lower range of values than Windows Update. That said, CPM had the

highest Standard Deviation of the iteration by a fairly large margin in comparison to the more traditional methodologies. Table 5 serves as a reference for the statistical metrics of all series data that was collected throughout iteration 5.

4.3 Endpoint CPU Utilization

The CPU utilization of every endpoint was captured throughout the patching process to get an idea of how demanding each methodology could be over time. The CPU utilization data for each endpoint of each iteration was parsed by statistical scripts to gather the mean and median values.

The following chart is an overview of the averages of the individual endpoint outcomes for all iterations of each method and provides an overall look at the trends that each method took over the course of the research.

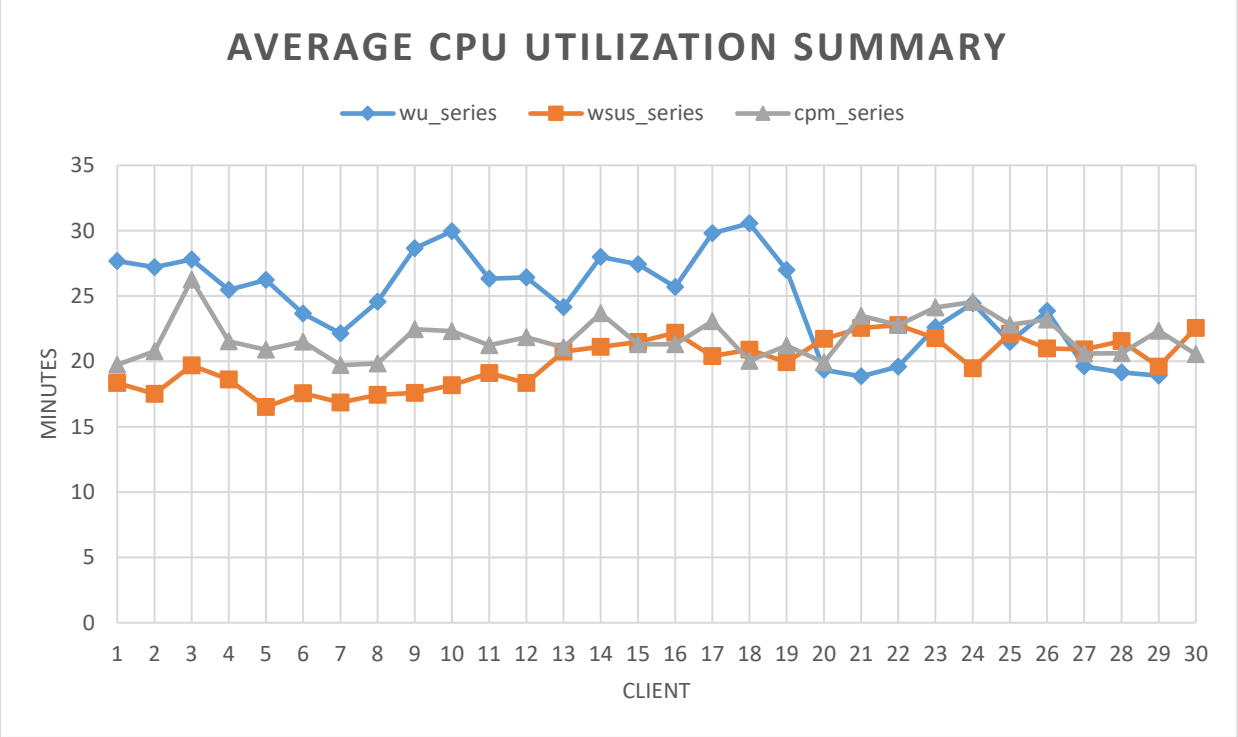


Figure 11: Average CPU Utilization by Series

4.3.2 Iteration 1 CPU Utilization

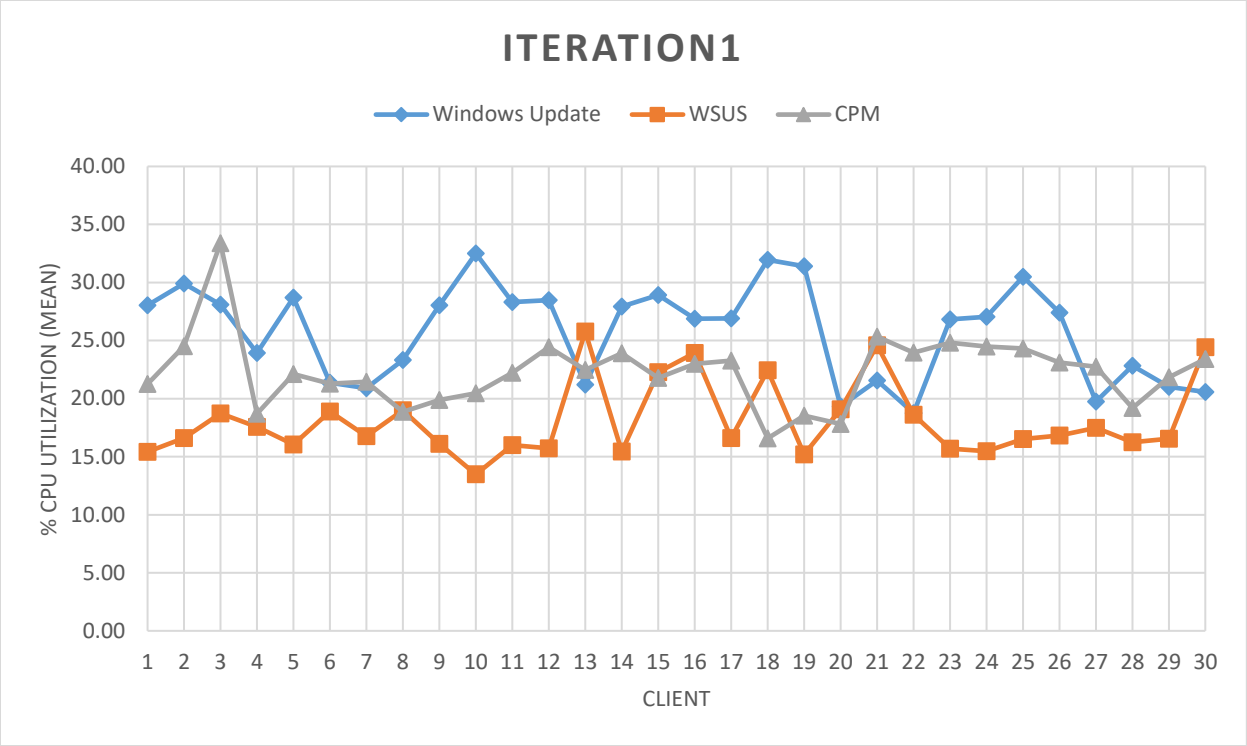


Figure 12: Iteration 1 CPU averages

Series	Iteration	Average Mean	Average Median
Windows Update	1	22.55	10.92
WSUS	1	21.98	8.74
CPM	1	21.76	9.51

Table 6: Iteration 1 Mean/Median CPU utilization averages

Iteration 1 sees the establishment of a general trend that holds true across all subsequent iterations in that the WSUS series had the least CPU utilization and the Windows Update series had the most CPU utilization with the CPM series hovering in the middle. While the deltas between the averages of averages vary slightly, the pattern holds across all iterations.

4.3.3 Iteration 2 CPU Utilization

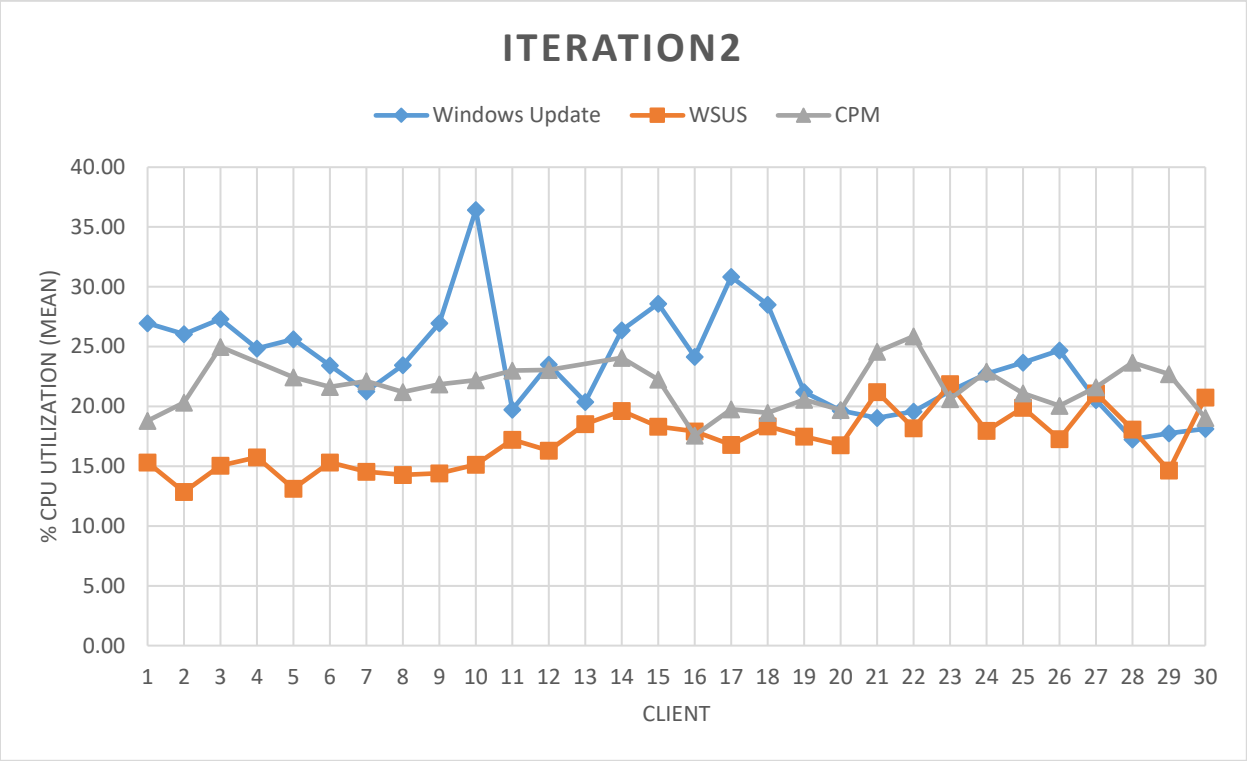


Figure 13: Iteration 2 CPU averages

Series	Iteration	Average Mean	Average Median
Windows Update	2	24.46	16.18
WSUS	2	19.98	8.80
CPM	2	21.93	8.94

Table 7: Iteration 2 Mean/Median CPU utilization averages

The Windows Update endpoints for Iteration 2 demonstrated the highest CPU utilization numbers while the WSUS endpoints demonstrated the least. Again, the CPM remained in the middle. Iteration 2 shows the beginning of a trend that holds true for the remainder of the iterations in that the Windows Update average of median values is nearly double that of the WSUS value for that iteration.

4.3.4 Iteration 3 CPU Utilization

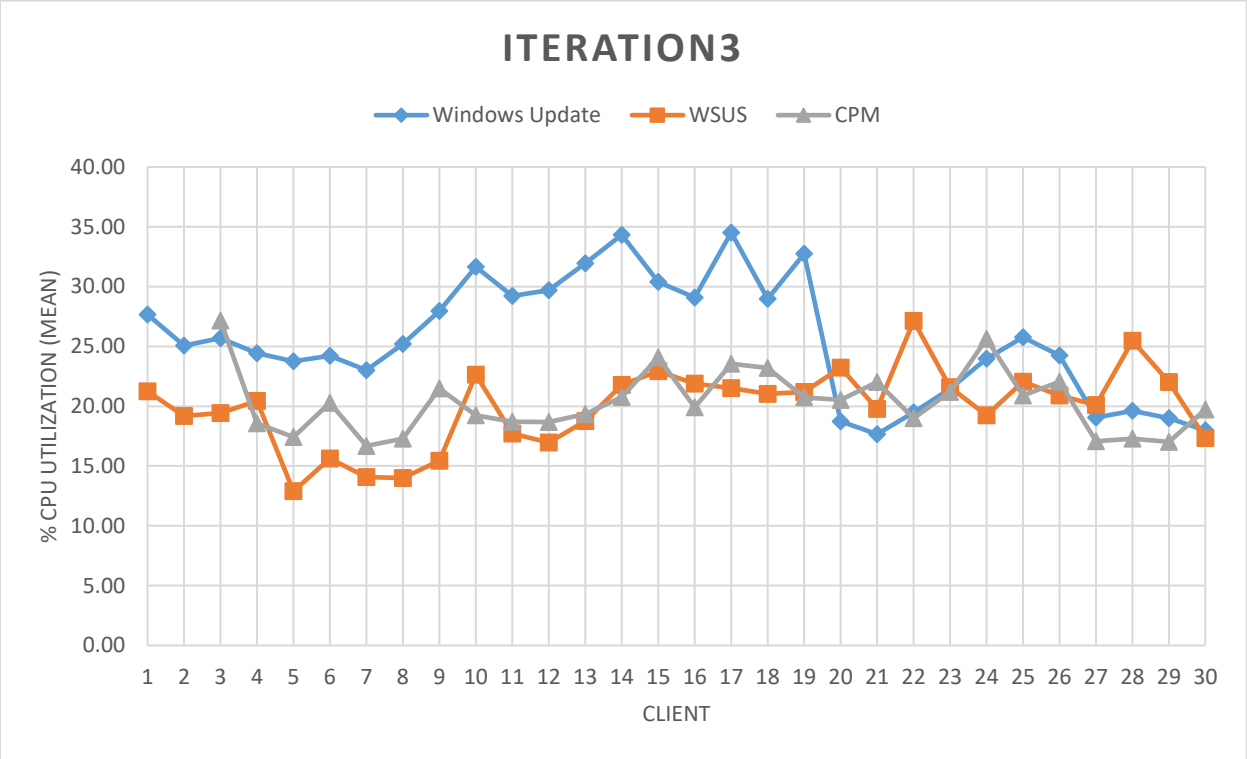


Figure 14: Iteration 3 CPU averages

Series	Iteration	Average Mean	Average Median
Windows Update	3	24.62	16.39
WSUS	3	19.94	8.71
CPM	3	21.81	8.88

Table 8: Iteration 3 Mean/Median CPU utilization averages

The trends established in the previous iterations continue to hold with WSUS showing the best performance and Windows Update showing the worst. Subsequent tables and figures are provided without commentary as there is little difference between them and the first 3 iterations that have been discussed.

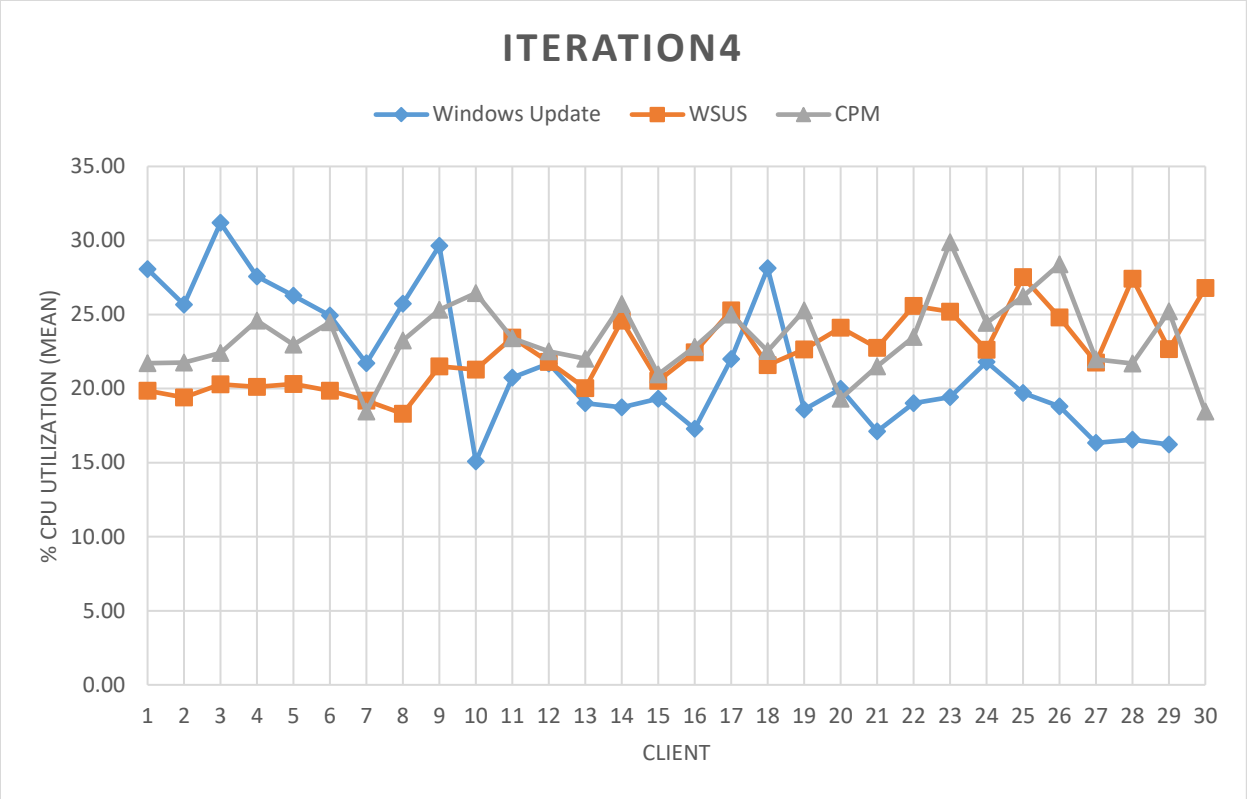


Figure 15: Iteration 4 CPU averages

Series	Iteration	Average Mean	Average Median
Windows Update	4	24.57	16.29
WSUS	4	20.02	8.81
CPM	4	21.78	8.86

Table 9: Iteration 4 Mean/Median CPU utilization averages

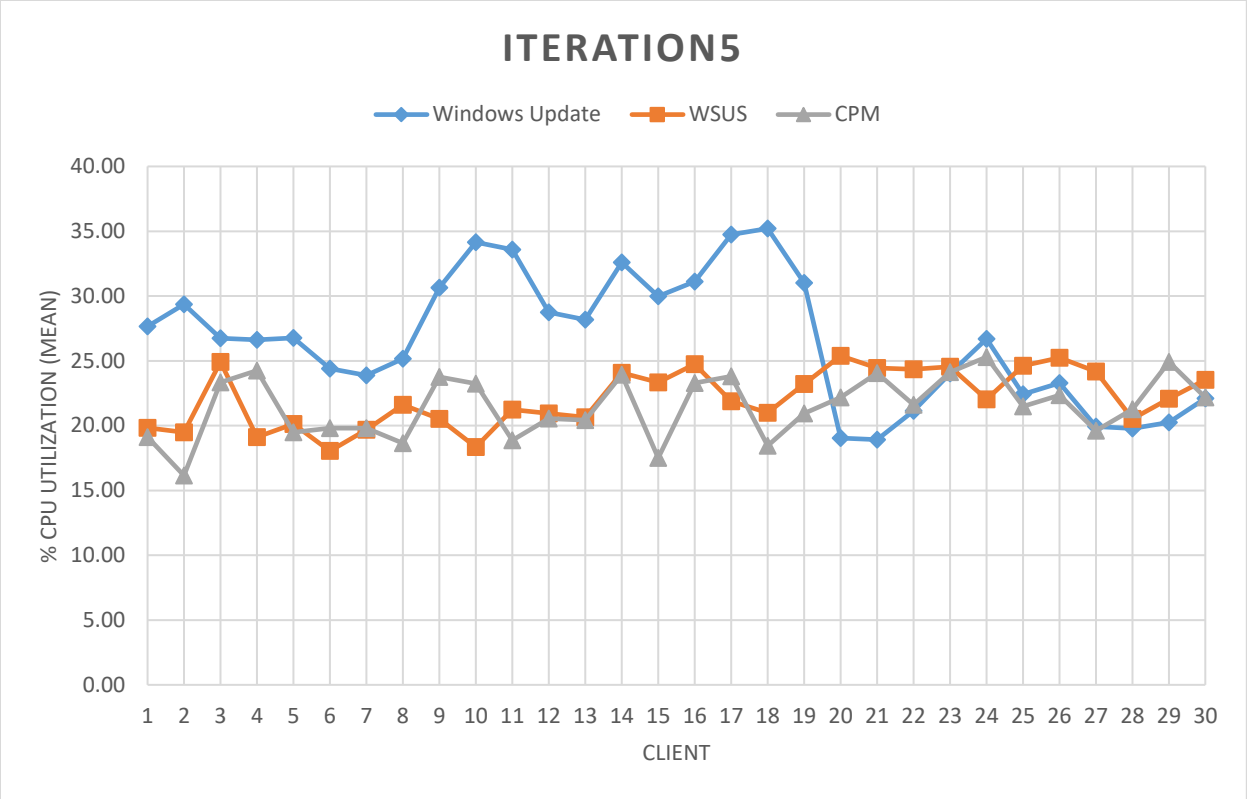


Figure 16: Iteration 5 CPU averages

Series	Iteration	Average Mean	Average Median
Windows Update	5	24.54	16.26
WSUS	5	20.01	8.81
CPM	5	21.82	8.89

Table 10: Iteration 5 Mean/Median CPU utilization averages

4.4 Bandwidth Considerations

While the Time to Completion and CPU utilization metrics are interesting, the bandwidth considerations have more dramatic implications when it comes to which methodology an organization decides to utilize. The WSUS and CPM methodologies utilize locally cached update content where possible; if content is not locally available, a single endpoint will download that content and store it locally so that other endpoints on the network do not have to individually pull that content down on their own. As was discussed in Chapter 2, the formula for this reduction of bandwidth is realized in the simple

equation of (N-1)*S with N representing the number of endpoints in an organization and S being the size of the update content that would otherwise have been downloaded.

The research for this thesis was conducted in an environment with 30 endpoints and an Internet connection that is approximately 2.5 times faster than the average fixed Internet speed of the United States. Organizations with larger fleets and slower connections would be unlikely to achieve similar Time to Completion outcomes. The two updates that served as the test candidates for each methodology collectivity represent 421.2 MB of data and that is the total amount of data that either of the WSUS and CPM methodologies would have to download to serve their fleets. The table and chart below represent calculations based upon that amount of data and the equation for bandwidth savings that was previously established.

Fleet Size	Content Size (MB)	Bandwidth Savings (MB)	Bandwidth Savings (GB)
30	421.2	12214.8	12.21
100	421.2	41698.8	41.70
500	421.2	210178.8	210.18
1000	421.2	420778.8	420.78
5000	421.2	2105578.8	2105.58
10000	421.2	4211578.8	4211.58
20000	421.2	8423578.8	8423.58

Table 11: Bandwidth Calculations

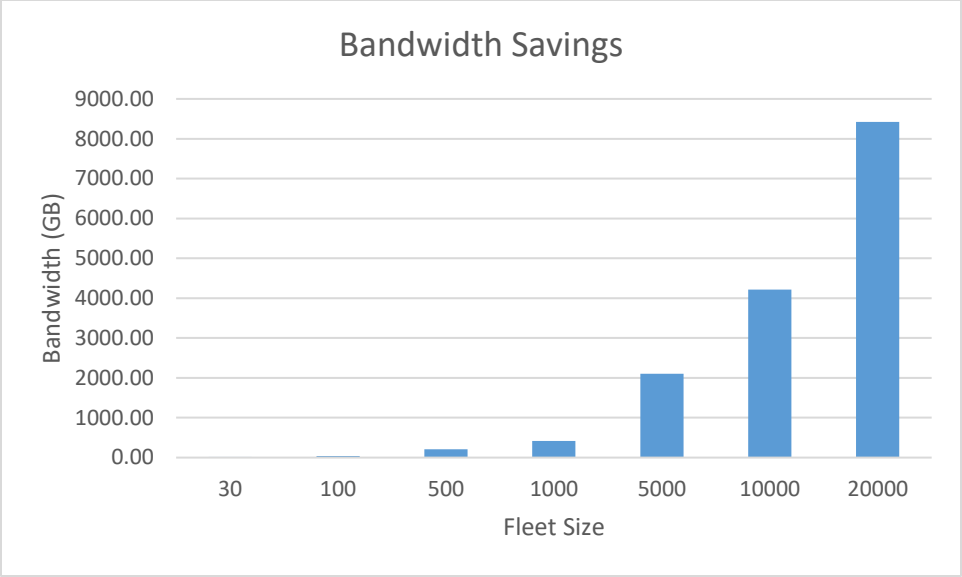


Figure 17: Bandwidth Savings for WSUS/CPM utilization

The data above portrays the serious ramifications that can come of choosing a methodology that is not appropriate for the unique circumstances of an organization. In the research conducted for this thesis, the data shows that every iteration of the Windows Update methodology consumed 12.21 GB of data across the WAN that could have otherwise been saved with competing methodologies. Based on that figure, 61.05 GB of data was consumed by the Windows Update iterations throughout the entirety of the research.

The contrasts become starker as the fleet sizes increase. For a fleet of 500 endpoints that deployed the patches utilized in the research, 210 GB would be consumed unnecessarily on a monthly basis. A fleet of 5000 endpoints would consume over 2 TB of data. The sharp increase in data consumption conveys a growth pattern that would likely prove to be unsustainable for medium to large businesses. The patches that were selected represent what could be considered the bare minimum coverage in terms of monthly patching: patching other common products such as Internet Explorer, Adobe Flash, and Microsoft Office would exacerbate the consumption of data to an even greater extent.

CHAPTER 5 – CONCLUSIONS AND RECOMMENDTIONS

From the onset of this research, the primary goal was to develop a functional solution for patching the Windows operating system that does not require WSUS or any additional form of licensing. To that end, the CPM proof-of-concept can be considered a success. With that being said, there are a variety of improvements that could improve the efficiency of the CPM methodology. The competing methodologies have their own vulnerabilities that will be explored subsequently.

5.1 Conclusions

5.1.1 Time to Completion

The research conclusively demonstrates that the CPM is the slowest methodology on the whole and could benefit from some improvements in this area. The WSUS methodology proved to be consistent and competitive in completions across the board. Windows Update regularly outperformed both of the aforementioned methodologies but that performance would not hold in environments with inferior Internet connectivity and a greater number of endpoints.

5.1.2 CPU Utilization

The WSUS and CPM methodologies were close in terms of their utilization of CPU resources with Windows Update requiring slightly higher CPU resources and generally hovered between 4 and 5 percent higher in CPU utilization than the other methodologies. All of the solutions saw utilization spikes that consumed all of the available CPU resources on the endpoints which suggests that the behavior is likely unavoidable no matter what methodology an organization opts to utilize.

5.1.3 Bandwidth Considerations

Basic calculations revolving around the bandwidth consumptions of each individual method paint a bleak picture for the viability of Windows Update in mid-to-large organizations with 500 or more clients. WSUS and the CPM both download content once and ensure that it is locally available for any clients that require it from that point forward whereas clients configured to utilize Windows Update must

each download the content to themselves. Using the (N-1)*S formula, we saw that an organization with a fleet of 500 devices would consume 210 GB of bandwidth just by downloading the two patches from September 2018 that were utilized for research.

5.1.4 Future Works

The research that was conducted on behalf of this thesis demonstrates that the CPM is a viable solution that could benefit from some performance enhancements along with changes in process to reduce the impact of its current deficiencies. The most prominent issue that needs to be addressed is the Time to Completion as it was the area that was the most problematic when compared to the more common enterprise patching methodologies.

- MBSA Scanning Options: The Microsoft Baseline Security Analyzer can selectively exclude certain classifications of vulnerabilities from scanning. By default, the MBSA scans search for vulnerabilities in the OS, SQL, IIS, and Updates categories.

- Procedural Efficiency: The XML that contains the vulnerability results could be generated prior to the intended patch time so that the endpoint does not have to evaluate vulnerability applicability, copy updates, and install those updates at the same time.

- Content Prestaging: The patching content could easily be prestaged as an addendum to the procedural change suggested above.

- Dependency Elimination: While the Microsoft Baseline Security Analyzer is currently free for enterprise use, there is nothing that obligates Microsoft to refrain from licensing or discontinuing the product in the future. I believe it would be prudent to design a WSUS fallback solution that possesses the same functionality but is entirely PowerShell-based. The same problem is true of the WSUS Offline CAB that provides the metadata for the MBSA to use.

- Management Console: Much of the approachability and ease-of-use attributed to Windows Server Update Services would be lost if it were entirely driven from a command-line interface in

the way that the CPM currently functions. The development of a PowerShell-based management console would be beneficial to administrators who lack the technical expertise to work with a CLI-based utility.

- Supported Products: The WSUS Offline CAB only supports security updates, update rollups, and service packs. Decoupling the concept from the MBSA and WSUS CAB dependencies would allow for greater support of an even broader number of update classifications and allow administrators to provide a more comprehensive offering to their environment.
- Error Handling: As a proof-of-concept in a controlled research environment, the solution has a bare minimum of error handling. This would need to be remedied for production applications so as to be easily auditable in the event of failures.

5.1.5 Recommendations

I feel that a supplemental tool to WSUS is critical despite the impressive performance and consistency that was demonstrated throughout the research. The catalyst for the exploration of an alternative patching methodology stemmed from an architectural limitation of all WSUS environments and a lack of outage communication for that product from Microsoft Corporation, two problems which could potentially manifest at inopportune times. The CPM proof-of-concept was shown to be capable of patching endpoints despite being slower than the preferred enterprise methodology of WSUS.

The Custom Patch Manager would be best suited as a supplemental tool that could be called upon in an emergency to ensure that the most critical patches are deployable in an instance where Windows Server Update Services is down. There is little downside in having the CPM as a backup solution given that the dependencies that drive it are freely available to any organization. That said, I feel like functionality proposed in the 5.1.4 Future Works would need to be completed before it could truly be considered as an outright replacement for WSUS.

REFERENCES

Higgins, K. (2018). Unpatched Vulnerabilities the Source of Most Data Breaches. [online] Dark Reading. Available at: https://www.darkreading.com/vulnerabilities---threats/unpatched-vulnerabilities-the-source-of-most-data-breaches/d/d-id/1331465 [Accessed 11 Sep. 2018].

Docs.microsoft.com. (2017). *Issues with Synchronization*. [online] Available at: https://docs.microsoft.com/de-de/security-updates/windowsupdateservices/18127563 [Accessed 28 Oct. 2018].

"A Five-Year Analysis Of Reported Windows Vulnerabilities - Help Net Security." Help Net Security. N. p., 2018. Web. 12 Sept. 2018.

Rains, Tim. "Industry Vulnerability Disclosures Trending Up." Cloudblogs.microsoft.com. N. p., 2014. Web. 12 Sept. 2018.

Keizer, Gregg. "Windows 7 Update Guide: How 'Security-Only' And 'Monthly Rollups' Differ." Computerworld. N. p., 2018. Web. 8 Oct. 2018.

Keizer, Gregg. "Patch Expert Calls On Microsoft Execs To Fix Windows Updating." Computerworld. N. p., 2018. Web. 8 Oct. 2018.

Bowden, Z. (2018). *Microsoft pulls the Windows 10 October 2018 Update as it investigates user files going missing*. [online] Windows Central. Available at: https://www.windowscentral.com/microsoft-has-pulled-windows-10-october-2018-update-users-files-go-missing [Accessed 8 Oct. 2018].

Leonhard, W. (2018). *Born: September non-security Office updates blamed for WSUS synchronization failures @ AskWoody*. [online] Askwoody.com. Available at: https://www.askwoody.com/2018/born-september-non-security-office-updates-blamed-for-wsus-synchronization-failures/ [Accessed 10 Oct. 2018].

Leonhard, W. (2018). *More WSUS Sync failures @ AskWoody*. [online] Askwoody.com. Available at: https://www.askwoody.com/2018/more-wsus-sync-failures/ [Accessed 10 Oct. 2018].

Cable, J. (2018). *Updated version of Windows 10 October 2018 Update released to Windows Insiders - Windows Experience Blog*. [online] Windows Experience Blog. Available at: https://blogs.windows.com/windowsexperience/2018/10/09/updated-version-of-windows-10-october-2018-update-released-to-windows-insiders/ [Accessed 10 Oct. 2018].

Claburn, T. (2018). *Microsoft Windows 10 October update giving HP users BSOD*. [online] Theregister.co.uk. Available at: https://www.theregister.co.uk/2018/10/11/microsoft_windows_10_october_update_fix_still_stumbling/ [Accessed 13 Oct. 2018].

SANS Institute Internet Storm Center (2003). *Windows XP: Surviving the First Day*. [online] Obviousmag.org. Available at: http://obviousmag.org/archives/uploads/2004/2004081900_Surving_the_first_day.pdf [Accessed 25 Oct. 2018].

Shields, G. (2016). *Geek of All Trades: 6 Tips for 100 Percent WSUS Compliance*. [online] Docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/previous-versions/technet-magazine/gg537354(v=msdn.10) [Accessed 25 Oct. 2018].

Poggemeyer, L., Decker, J., Kirkland, D. and Plett, C. (2018). *Plan Your WSUS Deployment*. [online] Docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/windows-server/administration/windows-server-update-services/plan/plan-your-wsus-deployment [Accessed 26 Oct. 2018].

Appendix A: START-SERIESMANAGER POWERSHELL FUNCTION

```
<#
    .SYNOPSIS
        Begin the automated execution of multiple series and iterations.

    .DESCRIPTION
        This function will automatically revert VMs to baseline, power them on, begin the
    execution of a test series/iteration, monitor for completion,
        upload the statistical data that is collected, and begin the process for as many cycles it can
    while the NETLAB reservation is active.

    .PARAMETER Begin
        The beginning integer suffix of a block of machines you wish to begin a series/iteration
    on.

    .PARAMETER End
        The ending integer suffix of a block of machines you wish to begin a series/iteration on.

    .PARAMETER Series
        Please select the type of test series you wish to run.

    .PARAMETER SeriesIterations
        This parameter defines how many iterations of the chosen series that you intend to
    perform.
```

```
        .PARAMETER Iterations
            Please indicate how many iterations to attempt.

        .EXAMPLE
            PS C:\> Start-SeriesManager -Begin $value1 -End $value2 -Series wu_manager -
        SeriesIterations 5

        .NOTES
            Additional information about the function.
#>
function Start-SeriesManager
{
        [CmdletBinding()]
        param
        (
            [Parameter(Position = 1)]
            [ValidateRange(1, 30)]
            [int]$Begin = 1,
            [Parameter(Position = 2)]
            [ValidateRange(1, 30)]
            [int]$End = 30,
            [Parameter(Mandatory = $true,
                                Position = 3,
                                HelpMessage = 'Valid values are wu_series (Windows Update),
        wsus_series (WSUS update), sccm_series (SCCM update), or cpm_series (Custom Patch
        Method update)')]
```

```
        [ValidateSet('wu_series', 'wsus_series', 'sccm_series', 'cpm_series')]
        [string]$Series,
        [Parameter(Mandatory = $false,
                                Position = 4)]
        [ValidateRange(1, 10)]
        [int]$SeriesIterations = 1
    )

$nextIteration = [int]$(Get-Content -Path "\\192.168.1.98\data\$Series\iteration.txt")

    if ($nextIteration -eq $null) {
        $nextIteration = 1
    }
    else {
        $nextIteration = $nextIteration + 1
    }

$psdir = "C:\Users\bhenderson\OneDrive\Documents\Education\Thesis Work\scripts"
$range = "$Begin".."$End"
$runCount = 0

    Get-ChildItem "${psdir}\*.ps1" | Where-Object { $_ -notlike "*TempPoint*" } | Select-
    Object -ExpandProperty Fullname | ForEach-Object{ . "$_" }

do
{
```

```
Connect-LabVIServer | Out-Null

Revert-LabToSnapShot -Begin $Begin -End $End -Series $Series

Write-Host "Sleeping 60 seconds for snapshot reversion."

Start-Sleep -Seconds 30

Start-Machines  -Begin $Begin -End $End

do
{
   Write-Host 'Waiting for VMs to power on...'
   Start-Sleep -Seconds 2
}
until ($($range | ForEach-Object {Get-VM -Name ('HendClient' + "$_") | Where-Object
    {$_.PowerState -eq 'PoweredOn'}}).Count -eq $End)

do
        {
            Write-Host 'Waiting for Windows to complete startup on endpoints'
            Start-Sleep 2
        }
        until ($($range | ForEach-Object {Test-Connection -ComputerName ('HendClient' + $_)
    -Count 1 -ErrorAction SilentlyContinue}).Count -eq $End)
```

```
Start-LabTests -Begin $Begin -End $End -Series "$Series"

$trackerMaster = [System.Collections.ArrayList]@{}
$dropDeadTime = $(Get-Date).AddHours(2)
$range | ForEach-Object {$trackerMaster.Add('HendClient' + "$_")} | Out-Null

    do
    {
  $count = $(Get-ChildItem -Path "\\192.168.1.98\data\$Series" -Include (-join
   ('iteration',"$nextIteration",'_','startFinish','_','HendClient','*','.csv')) -Recurse).Count
        Write-Host "$count endpoints have completed the current iteration.  There are $($End
   - $count) more pending."
        Start-Sleep 3
    }
    until ($count -eq $End -or $(Get-Date) -ge $dropDeadTime)

$nextIteration ++
$runCount ++

Restart-Computer -ComputerName HendSCCM -Verbose -Force

do
    {
        Write-Host 'Waiting for Windows to complete startup on endpoints'
        Start-Sleep 2
    }
```

```
            until ([bool]$(Test-Connection -ComputerName HendSCCM -Count 1 -ErrorAction
        SilentlyContinue) -eq $true)


    }
    until ($runCount -eq $SeriesIterations)
}
```

Appendix B: CONNECT-LABVISERVER POWERSHELL FUNCTION

```
function Connect-LabVIServer
{
    Connect-VIServer -Server '141.165.201.182' -User "bhenderson@vsphere.local" -Password $pwd
}
```

Appendix C: REVERT-LABTOSNAPSHOT POWERSHELL FUNCTION

```
<#
        .SYNOPSIS
            Revert-LabToSnapshot

        .DESCRIPTION
            This function shall revert one, many, or all lab machines to a former state
      by invoking the snapshot functionality provided by VMware vSphere.

        .PARAMETER Begin
            The beginning integer suffix of a block of machines you wish to revert.

        .PARAMETER End
            The ending integer suffix of a block of machines you wish to revert.

        .EXAMPLE
            PS C:\> Revert-LabToSnapshot -Begin $value1 -End $value2

        .NOTES
            Additional information about the function.
#>
function Revert-LabToSnapshot
{
        [CmdletBinding()]
        param
        (
```

```
        [Parameter(Position = 1)]
        [ValidateRange(1, 50)]
        [int]$Begin = 1,
        [Parameter(Position = 2)]
        [ValidateRange(1, 50)]
        [int]$End = 30,
        [Parameter(Mandatory = $true,
                        Position = 3,
                        HelpMessage = 'Valid values are wu_series (Windows Update),
    wsus_series (WSUS update), sccm_series (SCCM update), or cpm_series (Custom Patch
    Method update)')]
        [ValidateSet('wu_series', 'wsus_series', 'sccm_series', 'cpm_series')]
        [string]$Series
    )

    $range = "$Begin".."$End"

switch ( $Series) {

'wu_series'{ $snapShot = 'Stage0' }
'wsus_series'{ $snapShot = 'Stage1' }
'sccm_series'{ $snapShot = 'Stage2' }
'cpm_series'{ $snapShot = 'Stage0' }

}
```

```
        foreach ($item in $range)
        {
            $vm = 'HendClient' + $item
            $targetSnap = Get-Snapshot -VM $vm -Name $snapShot
    Set-VM -VM $vm -Snapshot $snapShot -RunAsync -Confirm:$false
        }
}
```

Appendix D: START-LABMACHINES POWERSHELL FUNCTION

```
<#
    .SYNOPSIS
        Power on laboratory machines for thesis project.

    .DESCRIPTION
        A detailed description of the Start-LabMachines function.

    .PARAMETER Begin
        The beginning integer suffix of a block of machines you wish to power on.

    .PARAMETER End
        The ending integer suffix of a block of machines you wish to power on.

    .PARAMETER All
        A description of the All parameter.

    .EXAMPLE
        PS C:\> Start-LabMachines -Begin $value1 -End $value2
    .EXAMPLE
        PS C:\> Start-LabMachines -All

    .NOTES
        Additional information about the function.
#>
function Start-LabMachines
```

```
{
    [CmdletBinding()]
    param
    (
        [Parameter(Position = 1)]
        [ValidateRange(1, 30)]
        [int]$Begin = 1,
        [Parameter(Position = 2)]
        [ValidateRange(1, 30)]
        [int]$End = 30
    )

    $range = "$Begin".."$End"

    foreach ($item in $range)
    {
        $rand = Get-Random -Minimum 3 -Maximum 5
        $vm = 'HendClient' + $item
        Start-PowerCLIJob -DefaultVIServer $DefaultVIServer -JobName $vm -ArgumentList
$vm -ScriptBlock { Start-VM $using:vm -Confirm:$false }
        Start-Sleep -Seconds $rand
    }
}
```

Appendix E: START-LABTESTS POWERSHELL FUNCTION

```
<#
        .SYNOPSIS
            Begin the execution of a test series and iteration.

        .DESCRIPTION
            A detailed description of the Start-LabTests function.

        .PARAMETER Begin
            The beginning integer suffix of a block of machines you wish to begin a series/iteration
        on.

        .PARAMETER End
            The ending integer suffix of a block of machines you wish to begin a series/iteration on.

        .PARAMETER Series
            Please select the type of test series you wish to run.

        .EXAMPLE
            PS C:\> Start-LabTests -Begin $value1 -End $value2

        .NOTES
            Additional information about the function.
#>
function Start-LabTests
{
```

```
[CmdletBinding()]
param
(
    [Parameter(Position = 1)]
    [ValidateRange(1, 30)]
    [int]$Begin = 1,
    [Parameter(Position = 2)]
    [ValidateRange(1, 30)]
    [int]$End = 30,
    [Parameter(Mandatory = $true,
                Position = 3,
                HelpMessage = 'Valid values are wu_series (Windows Update),
wsus_series (WSUS update), sccm_series (SCCM update), or cpm_series (Custom Patch
Method update)')]
    [ValidateSet('wu_series', 'wsus_series', 'sccm_series', 'cpm_series')]
    [string]$Series
)

$range = "$Begin".."$End"
$pwd = ConvertTo-SecureString "password" -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential "bhenderson", $pwd
$global:masterObject = [System.Collections.ArrayList]@()

$iteration = Get-Content -Path "\\192.168.1.98\data\$Series\iteration.txt"

if ($iteration -eq $null) {
```

```
        Set-Content -Path "\\192.168.1.98\data\$Series\iteration.txt" -Value '1' -Force
$iteration = Get-Content -Path "\\192.168.1.98\data\$Series\iteration.txt"
    }
    else {
        Set-Content -Path "\\192.168.1.98\data\$Series\iteration.txt" -Value ([int]$iteration + 1)
    -Force
        $iteration = Get-Content -Path "\\192.168.1.98\data\$Series\iteration.txt"
    }

    $iterationString = 'iteration' + $iteration

    if ($Series -eq 'wu_series' -or $Series -eq 'wsus_series') {
        $completionText = 'Installed [2] Updates'
$problemText = 'Found [0] Updates in pre search criteria'
$progressText = 'Accepted [2] Updates ready to Download'

        foreach ($item in $range) {
                $vm = 'HendClient' + $item
                $psWindowsUpdateLog = "\\$vm\c$\PSWindowsUpdate.log"

   #Stagger execution of every 10 items by 5 minutes

   if ($Series -eq 'wu_series') {
      if ($item % 10 -eq 0 -and $item -gt 9) {
         Start-Sleep -Seconds 600
      }
```

```
}
else {
   if ($item % 10 -eq 0 -and $item -gt 9) {
      Start-Sleep -Seconds 180
   }
}

            #Create array to track overall status

            $tempObject = New-Object psobject
            $tempObject | Add-Member -MemberType NoteProperty -Name 'Hostname' -
Value $vm
            $tempObject | Add-Member -MemberType NoteProperty -Name 'PatchStatus' -
Value 'Unpatched'
            $tempObject | Add-Member -MemberType NoteProperty -Name 'Start' -Value
(Get-Date)
            $tempObject | Add-Member -MemberType NoteProperty -Name 'End' -Value
$null
            $tempObject | Add-Member -MemberType NoteProperty -Name 'Iteration' -
Value $iteration

            #Begin job to monitor patching status of endpoint

            Start-Job -Name ($vm + '-Monitor') -ArgumentList
$psWindowsUpdateLog,$completionText,$tempObject,$Series,$iteration,$vm,$iterationStrin
g -ScriptBlock {
```

```
                $psWindowsUpdateLog = $using:psWindowsUpdateLog
  $completionText = $using:completionText
  $tempObject = $using:tempObject
  $Series = $using:Series
  $iteration = $using:iteration
  $iterationString = $using:iterationString

  Invoke-Command -ComputerName $using:vm -ArgumentList
$psWindowsUpdateLog,$completionText,$tempObject,$Series,$iteration,$iterationString -
ScriptBlock {
    $cmd = “net use Y: \\192.168.1.98\data /persistent:Yes /user:lab\bhenderson $pwd”
    $cmd2 = 'shutdown /a'
    if ([bool](Test-Path -Path 'Y:\') -eq $false) { Invoke-Expression -Command $cmd |
Out-Null }
    $tempObject = $using:tempObject

    do {

                $patchSuccess = Get-Content $using:psWindowsUpdateLog
-ErrorAction SilentlyContinue | Select-String -SimpleMatch $using:completionText -Quiet

                if ($patchSuccess -ne $true)
                {
        Start-Sleep -Milliseconds 250
                }
                else
```

```
                {
                    Invoke-Expression -Command $cmd2
        $tempObject[0].PatchStatus = 'Patched'
                    $tempObject[0].End = Get-Date
        $tempObject | Export-Csv -Path ("C:\$using:Series\$env:COMPUTERNAME\" +
"$using:iterationString" + "_startFinish_$env:COMPUTERNAME.csv") -
NoTypeInformation -Force
        $tempObject | Export-Csv -Path ("C:\" + "$using:iterationString" +
"_startFinish_$env:COMPUTERNAME.csv") -NoTypeInformation -Force
        Copy-Item -Path $using:psWindowsUpdateLog -Destination
("Y:\$using:Series\$env:COMPUTERNAME\" + "$using:iterationString" +
"PSWindowsUpdateLog_$env:COMPUTERNAME.log") -Force
        Copy-Item -Path ("C:\" + "$using:iterationString" +
"_startFinish_$env:COMPUTERNAME.csv") -Destination
("Y:\$using:Series\$env:COMPUTERNAME\" + "$using:iterationString" +
"_startFinish_$env:COMPUTERNAME.csv" ) -Force
        Copy-Item -Path ("C:\" + "$using:iterationString" +
"_$env:COMPUTERNAME.csv") -Destination
("Y:\$using:Series\$env:COMPUTERNAME\" + "$using:iterationString" +
"_$env:COMPUTERNAME.csv" ) -Force
        Start-Sleep -Seconds 2
        Restart-Computer -Force
                }
            }
            until ($(Test-Path -Path $using:psWindowsUpdateLog) -eq $true -and
$patchSuccess -eq $true)
```

```
  }
                }

                #Begin performance monitoring of endpoint

                Invoke-Command -ComputerName $vm -ArgumentList $iterationString,$Series
-AsJob -ScriptBlock {
                        $iterationString = $using:iterationString
  $Series = $using:Series
  $cmd = net use Y: \\192.168.1.98\data /persistent:Yes /user:lab\bhenderson $pwd”
                        if ([bool](Test-Path -Path 'Y:\') -eq $false) { Invoke-Expression -
Command $cmd | Out-Null }
                        #Start-Job -ArgumentList $Series,$iterationString -ScriptBlock { Get-
Counter -ComputerName $env:COMPUTERNAME -Counter "\processor(_total)\%
processor time", "\Network Adapter(*intel*)\Bytes Total/sec" -Continuous | Export-Counter -
Path (-join
("Y:\","$using:series","\","$env:COMPUTERNAME","\","$using:iterationstring","_","$env:
COMPUTERNAME",".csv")) -FileFormat CSV -Force}
  Get-Counter -ComputerName $env:COMPUTERNAME -Counter "\processor(_total)\%
processor time", "\Network Adapter(*intel*)\Bytes Total/sec" -Continuous | Export-Counter
-Path (-join ("C:\","$iterationstring","_","$env:COMPUTERNAME",".csv")) -FileFormat
CSV -Force
                }

                #Execute update installation
```

```
            Invoke-Command -Credential $creds -ComputerName $vm -ArgumentList
$psWindowsUpdateLog,$completionText,$problemText -AsJob -ScriptBlock {

  try
  {
    Get-WindowsUpdate -Install -NotKBArticleID "KB4457133" -AcceptAll -IgnoreReboot
-ScheduleJob ($(Get-Date).AddMinutes(1)) -Verbose
  }
  finally
  {
    do
    {
      $patchProblem = Get-Content $using:psWindowsUpdateLog -ErrorAction
SilentlyContinue | Select-String -SimpleMatch $using:problemText -Quiet
                                $patchProgress = Get-Content $using:psWindowsUpdateLog
-ErrorAction SilentlyContinue | Select-String -SimpleMatch $using:progressText -Quiet

      if ($patchProblem -eq $true -or $patchProgress -eq $null)
                                {
                                        Get-WindowsUpdate -Install -NotKBArticleID
"KB4457133" -AcceptAll -IgnoreReboot -ScheduleJob ($(Get-Date).AddMinutes(1)) -
Verbose
                                }

      Start-Sleep -Seconds 300
    }
```

```
      until ($patchProblem -eq $false -and $patchProgress -eq $true)
  }
                }
    }
}
elseif ($Series -eq 'cpm_series')
{
    foreach ($item in $range)
    {
                $vm = 'HendClient' + $item

                Start-Job -Name ($vm + '-Monitor') -ArgumentList
$creds,$iteration,$iterationString,$Series,$vm -ScriptBlock {
                        $creds = $using:creds
                        $iteration = $using:iteration
                        $iterationString = $using:iterationString
                        $Series = $using:Series
                        $vm = $using:vm
                        $remoteSession = New-PSSession -ComputerName $vm -Credential
$creds

                        Invoke-Command -Session $remoteSession -FilePath
"C:\Users\bhenderson\OneDrive\Documents\Education\Thesis Work\scripts\Invoke-
LabCPM.ps1"

  #Used for Iterations 1 through 5
```

```
            #Invoke-Command -Session $remoteSession -ArgumentList $iteration -ScriptBlock
        {Invoke-LabCPM -Iteration $using:iteration -UseNetwork -Download}

            #Used for iterations 5 through 10
            Invoke-Command -Session $remoteSession -ArgumentList $iteration -ScriptBlock
        {Invoke-LabCPM -Iteration $using:iteration -UseNetwork -Download -PriorXMLPath
        'C:\temp\results.xml'}
                        }
                }
        }
}
```

Appendix F: STOP-LABMACHINES POWERSHELL FUNCTION

```
<#
    .SYNOPSIS
        Power off laboratory machines for thesis project.

    .DESCRIPTION
        A detailed description of the Stop-LabMachines function.

    .PARAMETER Begin
        The beginning integer suffix of a block of machines you wish to power on.

    .PARAMETER End
        The ending integer suffix of a block of machines you wish to power on.

    .PARAMETER All
        A description of the All parameter.

    .EXAMPLE
        PS C:\> Stop-LabMachines -Begin $value1 -End $value2
    .EXAMPLE
        PS C:\> Stop-LabMachines -All

    .NOTES
        Additional information about the function.
#>
```

```
function Stop-LabMachines
{
	[CmdletBinding()]
	param
	(
	    [Parameter(Position = 1)]
	    [ValidateRange(1, 50)]
	    [int]$Begin = 1,
	    [Parameter(Position = 2)]
	    [ValidateRange(1, 50)]
	    [int]$End = 50
	)

	$count = "$Begin".."$End"

	foreach ($item in $count)
	{
	    $vm = 'HendClient' + $item
	    #Start-PowerCLIJob -DefaultVIServer $DefaultVIServer -JobName $vm -ArgumentList
	$vm -ScriptBlock { Start-VM $using:vm -Confirm:$false }
	    Start-Job -Name "$vm Stop" -ArgumentList $vm -ScriptBlock { Stop-Computer -
	ComputerName $using:vm -Force -Verbose }
	}
}
```

Appendix G: INVOKE-LABCOMMAND POWERSHELL FUNCTION

```
<#
        .SYNOPSIS
            Execute a scriptblock on one, many, or all of the lab endpoints as needed.

        .DESCRIPTION
            A detailed description of the Invoke-LabCommand function.

        .PARAMETER Begin
            The beginning integer suffix of a block of machines you wish to execute a scriptblock on.

        .PARAMETER End
            The ending integer suffix of a block of machines you wish to execute a scriptblock on.

        .PARAMETER ScriptBlock
            The command/script that you intend to be executed on the targeted endpoints.

        .EXAMPLE
            PS C:\> Invoke-LabCommand -Begin $value1 -End $value2

        .NOTES
            Additional information about the function.
#>
function Invoke-LabCommand
{
```

```
        [CmdletBinding()]
        param
        (
            [Parameter(Position = 1)]
            [ValidateRange(1, 50)]
            [int]$Begin = 1,
            [Parameter(Position = 2)]
            [ValidateRange(1, 50)]
            [int]$End = 50,
            [Parameter(Mandatory = $true,
                                Position = 3)]
            [string]$ScriptBlock
        )

        $ScriptBlock = [ScriptBlock]::Create($ScriptBlock)

    $range = "$Begin".."$End"

        foreach ($item in $range)
        {
            $vm = 'HendClient' + $item
       Invoke-Command -ComputerName $vm -ScriptBlock {Invoke-Expression
        $using:ScriptBlock} -AsJob
        }
}
```

Appendix H: START-POWERCLIJOB POWERSHELL FUNCTION

```
Function Start-PowerCLIJob {


param(
   [parameter(mandatory=$True)]
   [VMware.VimAutomation.ViCore.Impl.V1.VIServerImpl]
   $DefaultVIServer,
   [string]
   $JobName,
   [parameter(mandatory=$True)]
   [scriptblock]
   $ScriptBlock,
   [object[]]
   $ArgumentList,
   [psobject]
   $InputObject,
   [string[]]
   $Modules = "VMware.VimAutomation.Core"
)


$ScriptBlockPrepend = {import-module $using:Modules | out-null;
Set-PowerCLIConfiguration -DisplayDeprecationWarnings:$false -Scope Session -confirm:$False |
        out-null;
Connect-ViServer -Server $using:DefaultVIServer.name -session
        $using:DefaultVIServer.SessionSecret | out-null;
}
```

```
$ScriptBlock = [ScriptBlock]::Create($ScriptBlockPrepend.ToString() + $ScriptBlock.ToString())


$params = @{scriptblock=$ScriptBlock}
if ($JobName) {$params.Add('name',$JobName)}
if ($ArgumentList) {$params.Add('ArgumentList',$ArgumentList)}
if ($InputObject) {$params.Add('InputObject',$InputObject)}

Start-Job @params


}
```

Appendix I: INVOKE-LABCPM POWERSHELL FUNCTION

<#

.SYNOPSIS

Download and install missing patches utilizing offline WSUS CAB file.

.DESCRIPTION

This function will scan system, detect missing updates, download the updates, and install the updates on an endpoint. The function requires the following files:

-wsusscn2.cab

-mbsacli.exe

-wusscan.dll

.PARAMETER SourceFolder

This is the source folder for the mandatory files indicated in the Synopsis. Defaults to C:\temp.

.PARAMETER Iteration

A description of the Iteration parameter.

.PARAMETER UseNetwork

This parameters determines whether or not the function searched for update files on a local network share before downloading them from the Internet.

.PARAMETER NetworkSource

```
        Used in conjunction with the UseNetwork parameter, this parameter identifies the path of
        the network share to be utilized.

        .PARAMETER ValidateCab
            Determine whether or not the WSUS Offline CAB is up to date and download the
        updated version if it is outdated.

        .PARAMETER Download
            A description of the Download parameter.

        .PARAMETER HomeTest
            A description of the HomeTest parameter.

        .PARAMETER PriorXMLPath
            This parameter is the path for previously generated XML.

        .EXAMPLE
            PS C:\> Invoke-LabCPM -SourceFolder 'Value1'

        .NOTES
            Additional information about the function.
#>
function Invoke-LabCPM
{
        [CmdletBinding()]
        param
```

```
(
    [Parameter(Position = 1,
                    HelpMessage = 'This is the source folder for the mandatory files
indicated in the Synopsis.  Defaults to C:\temp.')]
    [string]$SourceFolder = 'C:\temp',
    [Parameter(Position = 2)]
    [int]$Iteration = 0,
    [Parameter(Position = 3)]
    [switch]$UseNetwork = $true,
    [Parameter(Position = 4)]
    [string]$NetworkSource = '\\192.168.1.98\data\_storage',
    [switch]$ValidateCab = $false,
    [switch]$Download = $true,
    [switch]$HomeTest = $false,
    [string]$PriorXMLPath
)

#Begin performance monitoring

Start-Job -ArgumentList $Iteration -ScriptBlock { Get-Counter -ComputerName
$env:COMPUTERNAME -Counter "\processor(_total)\% processor time", "\Network
Adapter(*intel*)\Bytes Total/sec" -Continuous | Export-Counter -Path (-join
('C:\','iteration',"$using:Iteration", "_", "$env:COMPUTERNAME", ".csv")) -FileFormat
CSV -Force}

#Establish network pathing
```

```
if ($HomeTest -eq $true) { $NetworkSource =
'C:\Users\Brent\OneDrive\Documents\Education\Thesis Work\source_files' }

$cmd = 'net use Y: \\192.168.1.98\data /persistent:Yes /user:lab\bhenderson Spec5bh5'
if ([bool](Test-Path -Path 'Y:\') -eq $false) { Invoke-Expression -Command $cmd | Out-
Null }

if (!(Test-Path $SourceFolder)) { New-Item -type directory -Force -Path $SourceFolder |
Out-NUll }
else { Write-Host "Detected: $SourceFolder" }
if (!(Test-Path ("$SourceFolder\patches"))) { New-Item -type directory -Force -Path
("$SourceFolder\patches") | Out-NUll }
else { Write-Host ("Detected: " + "$SourceFolder\patches") }

#Validate existence of prerequisites

$prerequisites = Get-ChildItem -Path $SourceFolder -Recurse -Include 'mbsacli.exe',
'wsusscn2.cab', 'wusscan.dll' -ErrorAction SilentlyContinue

if ($($prerequisites).Count -ne '3')
{
    Copy-Item -Path "$NetworkSource\*" -Include 'mbsacli.exe', 'wsusscn2.cab',
'wusscan.dll','results.xml' -Destination $SourceFolder -Force
}
```

```
$wsusscn2_path = Get-ChildItem -Path $SourceFolder -Recurse -Include 'wsusscn2.cab' -
ErrorAction SilentlyContinue | Select-Object -ExpandProperty Fullname
$mbsacli_path = Get-ChildItem -Path $SourceFolder -Recurse -Include 'mbsacli.exe' -
ErrorAction SilentlyContinue | Select-Object -ExpandProperty Fullname

#Variable Declarations

$tempObject = New-Object psobject
$tempObject | Add-Member -MemberType NoteProperty -Name 'Hostname' -Value
$env:COMPUTERNAME
$tempObject | Add-Member -MemberType NoteProperty -Name 'PatchStatus' -Value
'Unpatched'
$tempObject | Add-Member -MemberType NoteProperty -Name 'Start' -Value (Get-Date)
$tempObject | Add-Member -MemberType NoteProperty -Name 'End' -Value $null
$tempObject | Add-Member -MemberType NoteProperty -Name 'Iteration' -Value $iteration
$Series = 'cpm_series'

#Clean up prior runs

Get-ChildItem -Path $SourceFolder -Include "iteration*.xml",
"$env:COMPUTERNAME*.xml", 'install_patches.bat' -Recurse | Remove-Item -Force

Write-Host "Information and content generated from this function will be kept in:
$SourceFolder"

#WSUS CAB Validation
```

```
if ($ValidateCab -eq $true)
{
    ##Check if wsusscn2.cab is up to date
    $wsusscn2_url = "http://go.microsoft.com/fwlink/?LinkID=74689"
    $system_date = Get-Date
    $wsus_date = [datetime]((Get-ItemProperty -Path $wsusscn2_path -Name
LastWriteTime).lastwritetime)
    $Days = (New-TimeSpan -Start $system_date -End $wsus_date).Days

    if ($Days -lt -15)
    {
        Write-Host "wsusscn2 has not been updated whithin 15 days and could be out of
date"
        $in = Read-Host "Do you want to update the wsusscn2.cab file right now [Y|n]:"
        if ($in -eq "Y" -or $in -eq '')
        {
            $wc = New-Object System.Net.WebClient
            Write-host "Downloading file, this might take a while..."
            $wc.DownloadFile($wsusscn2_url, $wsusscn2_path)
            Write-Host "File download successfully"
        }
        else
        {
            Write-Host "skipping wsusscn2.cab update"
        }
```

```
    }
}

if (!($PriorXMLPath))
{
    Write-Host "Invoking mbsacli to gather vulnerability information.  Please be patient."
    $cmd = -join ('cmd.exe /c ', $mbsacli_path, ' /catalog ', $wsusscn2_path, ' /xmlout >
C:\temp\', 'iteration', "$iteration", '_results_', $env:COMPUTERNAME, '.xml')
    Invoke-Expression -Command "$cmd" -ErrorAction SilentlyContinue | Out-Null
    $UpdateXML = -join ('C:\temp\', 'iteration', "$iteration", '_results_',
$env:COMPUTERNAME, '.xml')
}
else
{
    $UpdateXML = $PriorXMLPath
}

#Collecting update information from XML.

$Updates = [xml](Get-Content $UpdateXML)

foreach ($Check in $Updates.XMLOut.Check | Where-Object { $_.Name -like 'Windows
Security Updates' })
{
    Write-Host "Checking for", $Check.Name
    Write-Host $Check.Advice.ToString()
```

```
    #Checking for files to download
    foreach ($UpdateData in $($Check.Detail.UpdateData | Where-Object {$_.ID -eq
'4457129 -and -ne '4457133"}))
    {
        if ($UpdateData.IsInstalled -eq $false)
        {
            $PatchID = $updateData.ID.ToString()

            Write-Host "$PatchID is not installed on the system"

            if ($Download -eq $true)
            {
                $url = [URI]$UpdateData.References.DownloadURL
                $file = $url.Segments[$url.Segments.Count – 1]
                $networkPatches = Get-ChildItem -Path "Y:\_storage\patches" -
Recurse -Include "*.exe", "*.cab", "*.msu"
                if ($networkPatches.Name -contains $file)
                {
                    Write-Host $UpdateData.Title, " has been located on
the network share.  Copying to local drive for installation."
                    Copy-Item -Path "Y:\_storage\patches\$file" -
Destination "$SourceFolder\patches" -Force
                }
                else
                {
```

```
                                    #Initialize webclient for downloading files
                                    $webclient = New-Object Net.Webclient
                                    $webClient.UseDefaultCredentials = $true

                                    Write-Host "Download the file for KB",
    $UpdateData.KBID
                                    Write-Host "Starting download ", $UpdateData.Title,
    "."
                                    $url = [URI]$UpdateData.References.DownloadURL
                                    $file = $url.Segments[$url.Segments.Count – 1]
                                    Write-Host $file
                                    $toFile = $SourceFolder + "\patches\" + $file
                                    $webClient.DownloadFile($url, $toFile)
                                    Write-Host "Download complete."
                                    Copy-Item -Path (-join
    ("$SourceFolder",'\','patches','\',"$file")) -Destination "Y:\_storage\patches" -Force
                                }
                            }
                        }
                    }
    }

Write-Host "Copying kb4054566 to local cache for installation."

Copy-Item -Path "Y:\_storage\patches\windows8.1-kb4054566-x64.cab" -Destination
    "$SourceFolder\patches" -Force
```

```
$installationfiles = Get-ChildItem -Path "$SourceFolder\patches" -Recurse -Include
        "*.cab","*.exe","*.msu" | Select-Object -ExpandProperty Name


foreach ($file in $installationfiles)
{
   Write-host ("Beginning installation of " + $file)
            if ($file.EndsWith(".msu"))
            {
                $exe = 'C:\windows\system32\wusa.exe'
                $arg = -join ("$SourceFolder",'\','patches','\',"$file", ' /quiet /norestart')
                Start-Process -Wait -FilePath $exe -ArgumentList $arg
            }
            elseif ($file.EndsWith(".cab"))
            {
                $exe = 'C:\Windows\system32\Dism.exe'
                $arg = -join ('/Online /Add-Package /PackagePath:', "$SourceFolder\patches\$file", '
        /quiet /NoRestart')
                Start-Process -Wait -FilePath $exe -ArgumentList $arg
            }
            elseif ($file.EndsWith(".exe"))
            {
                $exe = ($SourceFolder + '\patches\' + $file)
                $arg = -join (' /passive /norestart')
                Start-Process -Wait -FilePath $exe -ArgumentList $arg
            }
```

```
    else
    {
        Write-Host 'Unexpected file type found.  Please investigate.'
    }
}

do
{
    Start-Sleep -Milliseconds 500
}
until ($(Get-HotFix KB4457129, KB4054566).Count -eq '2')

$tempObject.PatchStatus = 'Patched'
$tempObject.End = Get-Date
$tempObject | Export-Csv -Path (-join
('C:\','iteration',"$Iteration",'_startFinish_',"$env:COMPUTERNAME",'.csv')) -
NoTypeInformation -Force
Copy-Item -Path (-join ("$SourceFolder", '\','iteration',"$Iteration",'_results_',
"$env:COMPUTERNAME", '.xml')) -Destination (-join ('Y:\', "$Series", '\',
"$env:COMPUTERNAME")) -Force

Copy-Item -Path (-join ('C:\', 'iteration', "$Iteration", '_startFinish_', "$env:COMPUTERNAME",
'.csv')) -Destination (-join ('Y:\', "$Series", '\', "$env:COMPUTERNAME")) -Force

Copy-Item -Path (-join ('C:\', 'iteration', "$Iteration", '_', "$env:COMPUTERNAME", '.csv')) -
Destination (-join ('Y:\', "$Series", '\', "$env:COMPUTERNAME")) -Force
```

```
    Start-Sleep -Seconds 2
}
```

Appendix J: GETMISSINGPATCHES POWERSHELL SCRIPT

```
<#
.Synopsis
   Install missing patches on the system
.DESCRIPTION
   This script will check for missing patches on the system using mbsacli and create a .bat file to
        automate the installation of those patches
   The script requires the following mbsa files:
      -Mbsacli.exe
      -wsusscn2.dll
      -wsusscn2.cab
   The files MUST be stored on the same folder
.EXAMPLE
   This command will check for missing patches and list them. Additionally it will generate .bat script
        to patch the system automatically
   ./GetMissingPatches -mbsaFolder C:\temp -outputFolder C:\temp
.EXAMPLE
   The -download switch will download missing patches automatically and strore them in
        C:\temp\patches folder
   ./GetMissingPatches -mbsaFolder C:\temp -outputFolder C:\temp -download
.EXAMPLE
   The -importResultsXML will import results.xml files generated on isolated environments that
        doesn't allow to download updates
   ./GetMissingPatches -importResultsXML C:\temp\results.xml
```

```
#>


Param (

        [string]$importResultsXML,
        #[Parameter(Mandatory=$False,Position=1)]

        [string]$mbsaFolder = "C:\temp",
        #[Parameter(Mandatory=$False,Position=2)]

        [string]$outputFolder = "C:\temp",
        #[Parameter(Mandatory=$False,Position=3)]

        [switch]$download

)

Write-Host "All information generated by the script will be stored in C:\temp"
Write-Host ""

#Checking parameters
if ($PSBoundParameters.ContainsKey('download'))
{
        $download = $True
        Write-Host "**********Download mode active**********"
```

```
}



$patchesFolder = $outputFolder + "\patches\"
$installFile = $outputFolder + "\Install_patches.bat"
New-Item -type directory -Force -Path $outputFolder | Out-NUll
New-Item -type directory -Force -Path $patchesFolder | Out-Null



##Check if wsusscn2.cab is up to date
$wsusscn2_url = "http://go.microsoft.com/fwlink/?LinkID=74689"
$wsusscn2_path = $mbsaFolder + "\wsusscn2.cab"
$mbsacli_path = $mbsaFolder + "\mbsacli.exe"
$system_date = Get-Date
$wsus_date = [datetime]((Get-ItemProperty -Path $wsusscn2_path -Name
        LastWriteTime).lastwritetime)
$Days = (New-TimeSpan -Start $system_date -End $wsus_date).Days

if ($Days -lt -15)
{
        Write-Host "wsusscn2 has not been updated whithin 15 days and could be out of date"
        $in = Read-Host "Do you want to update the wsusscn2.cab file right now [Y|n]:"
        if ($in -eq "Y" -or $in -eq '')
        {
            $wc = New-Object System.Net.WebClient
```

```
            Write-host "Downloading file, this might take a while..."
            $wc.DownloadFile($wsusscn2_url, $wsusscn2_path)
            Write-Host "File download successfully"
        }
        else
        {
            Write-Host "skipping wsusscn2.cab update"
        }
}


#if not importing previous results, then execute mbsacli
if (!$PSBoundParameters.ContainsKey('importResultsXML'))
{
        Write-Host "Checking for missing patches, this may take a while..."
        $cmd = "cmd.exe /c $mbsacli_path /catalog $wsusscn2_path /xmlout > C:\temp\results.xml"
        Invoke-Expression -Command "$cmd"
        #cmd.exe /c $mbsacli_path /catalog $wsusscn2_path /xmlout > C:\temp\results.xml
        $UpdateXML = "C:\temp\results.xml"
}
else
{
        $UpdateXML = $importResultsXML
}

#Initialize webclient for downloading files
```

```
$webclient = New-Object Net.Webclient
$webClient.UseDefaultCredentials = $true

#Get the content of the XML file
$Updates = [xml](Get-Content $UpdateXML)

"@Echo Off" | Out-File $installFile
"REM This script will install missing patches on the system" | Out-File $installFile -Append

# for each patch check if it's installed on the system

foreach ($Check in $Updates.XMLOut.Check)
{
        Write-Host "Checking for", $Check.Name
        Write-Host $Check.Advice.ToString()

        #Checking for files to download
        foreach ($UpdateData in $Check.Detail.UpdateData)
        {
            if ($UpdateData.IsInstalled -eq $false)
            {
                    $PatchID = $updateData.ID.ToString()
                    Write-Host "The patch $PatchID is not installed on the system"
                    if ($PSBoundParameters.ContainsKey('download'))
                    {
                            Write-Host "Download the file for KB", $UpdateData.KBID
```

```
                    Write-Host "Starting download ", $UpdateData.Title, "."
                    $url = [URI]$UpdateData.References.DownloadURL
                    $fileName = $url.Segments[$url.Segments.Count – 1]
                    Write-Host $fileName
                    $toFile = $outputFolder + "\patches\" + $fileName
                    $webClient.DownloadFile($url, $toFile)
                    Write-Host "Done downloading"

                    "@ECHO Starting installing " + $fileName | Out-File $installFile -
Append
                    if ($fileName.EndsWith(".msu"))
                    {
                            "wusa.exe " + $fileName + " /quiet /norestart
/log:%SystemRoot%\Temp\KB" + $UpdateData.KBID + ".log" | Out-File $installFile -
Append
                    }
                    elseif ($fileName.EndsWith(".cab"))
                    {
                            "start /wait pkgmgr.exe /ip /m:" + $fileName + " /quiet /nostart
/l:%SystemRoot%\Temp\KB" + $UpdateData.KBID + ".log" | Out-File $installFile -Append
                    }
                    else
                    {
                            $fileName + " /passive /norestart /log
%SystemRoot%\Temp\KB" + $UpdateData.KBID + ".log" | Out-File $installFile -Append
                    }
```

```
                    "@ECHO Installation returned %ERRORLEVEL%" | Out-File
        $installFile -Append
                    "@ECHO." | Out-File $installFile -Append
                    Write-Host
                }
            }
        }

        Write-Host
}

Write-Host "Job done!"
```

Appendix K: UPLOAD-LABDATA

```
function Upload-LabData
{
   Copy-Item -Path "\\hendsccm\data\" -Destination
        "C:\Users\bhenderson\OneDrive\Documents\Education\Thesis Work\" -Recurse -Container -
        Force
}
```

Appendix L: WINDOWS SERVER UPDATE SERVICES PRODUCT SUPPORT LIST

Active Directory Rights Management Services Client 2.0	Active Directory Rights Management Services Client 2.0 (AD RMS Client 2.0) is software designed for your client computers to help protect access to and usage of information flowing through applications that use AD RMS whether installed on your premises or in a Microsoft datacenter.
Active Directory	Active Directory Product Family Category
Antigen for Exchange/SMTP	Defines the category for Antigen Updates. This will make sure that the update with this category will be offered only when the applicability rule is met.
Antigen	Antigen Product Family Category

ASP.NET Web and Data Frameworks	ASP.NET Web and Data Frameworks product family
ASP.NET Web Frameworks	ASP.NET Web Framework
Azure File Sync agent updates for Windows Server 2012 R2	Azure File Sync agent updates for Windows Server 2012 R2
Azure File Sync agent updates for Windows Server 2016	Azure File Sync agent updates for Windows Server 2016
Azure File Sync	Azure File Sync
Bing Bar	Get quick access to Bing and MSN, as well as handy tools for online safety and productivity.
Bing	Live Search Product Family Category
BizTalk Server 2002	Category for BizTalk 2002. It requires SP1 as the minimum version.
BizTalk Server 2006R2	BizTalk Server 2006R2
BizTalk Server 2009	BizTalk Server 2009

BizTalk Server 2013	BizTalk Server 2013
BizTalk Server	BizTalk Server Product Family Category
CAPICOM	CAPICOM
Category for System Center Online Client	System Center Online Client is the client software for Microsoft Asset Inventory Service (AIS). Updates offered in this category will apply only to computers running the System Center Online Client software and will contain updates including security updates. AIS is an online service that translates inventory data into business intelligence. AIS is accessible through the Microsoft Desktop Optimization

	Pack for Software Assurance. For more information, see: http://www.microsoft.com/sconline For tips on how to configure AIS client computer behavior by using Group Policy, see: (http://technet.microsoft.com/en-us/sconline/bb847943.aspx).
Compute Cluster Pack	The Compute Cluster Pack product category will include updates for the Microsoft? Compute Cluster Pack, including service packs, optional updates, and critical or security updates. Updates offered through this category will apply

	only to computers running Compute Cluster Pack software.
Data Protection Manager 2006	Data Protection Manager 2006(DPM)is designed specifically for disk-based backup. DPM, the newest member of the Microsoft Windows Server System focuses on disk-based data protection and recovery. DPM installs on Microsoft Windows Server 2003 SP1 and protects servers running Microsoft Windows 2000 Server, Microsoft Windows Server 2003, and Windows Storage Server 2003 to deliver best-in-class data protection services.

Developer Tools, Runtimes, and Redistributables	Developer Tools, Runtimes, and Redistributables
Device Health	This is a category for Device Health. Device Health is a windows service to provide the device?s health information. By installing this software, you are encouraged to adopt secure practice in software usage, and the certified ecommerce and online banking partners can provide better protection based on the information gotten from Device Health.
Device Health	Device Health products family
Dictionary Updates for Microsoft IMEs	Contains the dictionary updates for Microsoft IMEs. Dictionary update is a data file which IME uses to fix issues in the

	corresponding dictionary.
Exchange 2000 Server	For Exchange 2000 Products
Exchange Server 2003	For Exchange 2003 Products
Exchange Server 2007 and Above Anti-spam	Microsoft Exchange Server 2007 and Above Anti-spam
Exchange Server 2007	Exchange Server 2007
Exchange Server 2010	Exchange Server 2010 Category
Exchange Server 2013	Exchange Server 2013 Category
Exchange Server 2016	Exchange Server 2016 Category
Exchange	Exchange
Expression Design 1	Category for checking whether Expression Design 1 is installed.
Expression Design 2	Category for checking whether Expression Design 2 is installed
Expression Design 3	Category for checking whether Expression Design 3 is installed

Expression Design 4	Category for checking whether Expression Design V4 is installed
Expression Media 2	Category for checking whether Expression Media 2 is installed.
Expression Media V1	Expression Media V1
Expression Web 3	Category for checking whether Expression Web V3 is installed
Expression Web 4	Category for checking whether Expression Web V4 is installed
Expression	Expression Product Family
Firewall Client for ISA Server	This category includes updates of Microsoft Firewall Client.
Forefront Client Security	Microsoft Forefront Client Security provides unified malware protection for business desktops, laptops, and server operating systems that is

easier to manage and control. Built on the same highly successful Microsoft protection technology already used by millions of people worldwide, Forefront Client Security helps guard against emerging threats such as spyware and rootkits, as well as traditional threats such as viruses, worms, and Trojan horses. By delivering simplified administration through central management and by providing critical visibility into threats and vulnerabilities, Forefront Client Security helps you protect your business with greater confidence and efficiency. Forefront Client Security integrates

	with your existing infrastructure software, such as Active Directory, and complements other Microsoft security technologies for better protection and greater control.
Forefront Endpoint Protection 2010	Forefront Endpoint Protection is a single product that delivers unified security management and reporting with comprehensive, coordinated protection across clients, server applications, and the network edge.
Forefront Identity Manager 2010 R2	FIM 2010 R2 Category.
Forefront Identity Manager 2010	FIM2010 Category
Forefront Protection Category	Product Family category for FPE & FPSP

Forefront Server Security Category	Product Family Category for Forefront
Forefront Threat Management Gateway, Definition Updates for HTTP Malware Inspection	Forefront Threat Management Gateway, Definition Updates for HTTP Malware Inspection is the location for malware inspection definition updates for Forefront Threat Management Gateway.
Forefront TMG MBE	Main category for the TMG updates
Forefront TMG	Main category for the TMG updates (post MBE)
Forefront	Forefront
HealthVault Connection Center Upgrades	Microsoft HealthVault Connection Center Upgrades
HealthVault Connection Center	Microsoft HealthVault Connection Center
Host Integration Server 2000	Category for Host Integration Server 2000 release. It

	requires SP2 as the minimum version
Host Integration Server 2004	Category for Host Integration Server 2004 release. It requires RTM as the minimum version
Host Integration Server 2006	Category for Host Integration Server 2006 release.
Host Integration Server 2009	Category for Host Integration Server 2009 release.
Host Integration Server 2010	Category for Host Integration Server 2010 release.
HPC Pack 2008	The HPC Pack 2008 product category will include updates for the Microsoft HPC Pack 2008 client, server, and SDK. Updates may include service packs, optional updates, critical updates, or security updates.
HPC Pack	The HPC Pack product family will include updates for

	all HPC Pack products Category
Internet Security and Acceleration Server 2004	This category contains updates for ISA Server 2004.
Internet Security and Acceleration Server 2006	Main category for ISA 2006.
Internet Security and Acceleration Server	Internet Security and Acceleration Server Product Family
Microsoft Advanced Threat Analytics	Product Family for Microsoft Advanced Threat Analytics
Microsoft Advanced Threat Analytics	
Microsoft Application Virtualization 4.5	Microsoft Application Virtualization 4.5
Microsoft Application Virtualization 4.6	Microsoft Application Virtualization 4.6
Microsoft Application Virtualization 5.0	The category of updates for Microsoft Application Virtualization 5.0 Client and Server

Microsoft Application Virtualization	Microsoft Application Virtualization Product Family
Microsoft Azure Information Protection Client	Microsoft Azure Information Protection helps you classify and label your data at the time of creation, based on a simple and intuitive interface
Microsoft Azure Information Protection	Product Family for Microsoft Azure Information Protection
Microsoft Azure Site Recovery Provider	Windows Azure Hyper-V Recovery Manager Provider Category for Product Code :6ccc483c-ad9e-468d-83f6-ad7fba2b310b
Microsoft Azure	Microsoft Azure product family
Microsoft BitLocker Administration and Monitoring v1	Microsoft BitLocker Administration and Monitoring v1

Microsoft BitLocker Administration and Monitoring	Microsoft BitLocker Administration and Monitoring products family
Microsoft Dynamics CRM 2011 SHS	Microsoft Dynamics CRM 2011 SHS
Microsoft Dynamics CRM 2011	Microsoft Dynamics CRM 2011
Microsoft Dynamics CRM 2013	Microsoft Dynamics CRM 2013
Microsoft Dynamics CRM 2015	Microsoft Dynamics CRM 2015
Microsoft Dynamics CRM 2016 SHS	Microsoft Dynamics CRM 2016 SHS
Microsoft Dynamics CRM 2016	Microsoft Dynamics CRM 2016
Microsoft Dynamics CRM	Microsoft Dynamics CRM
Microsoft HealthVault	Microsoft HealthVault Product Family
Microsoft Lync 2010	Category for Microsoft Lync 2010
Microsoft Lync Server 2010	This is the product category for Microsoft Lync Server 2010.

Microsoft Lync Server 2013	This is the product category for Microsoft Lync Server 2013.
Microsoft Lync Server and Microsoft Lync	Microsoft Communications Server and Microsoft Communicator
Microsoft Monitoring Agent (MMA)	Microsoft Monitoring Agent (MMA) product family
Microsoft Monitoring Agent	Category for releasing MMA update
Microsoft Online Services Sign-In Assistant	The Microsoft Online Services Sign-In Assistant provides end user sign-in capabilities to Microsoft Online Services, such as Office 365
Microsoft Online Services	Microsoft Online Services product family
Microsoft Research AutoCollage 2008	The 2008 version of MSR AutoCollage
Microsoft Research AutoCollage	Microsoft Research AutoCollage Product Family Category

Microsoft Security Essentials	Microsoft Security Essentials
Microsoft SQL Server 2008 R2 - PowerPivot for Microsoft Excel 2010	PowerPivot
Microsoft SQL Server 2012	Microsoft SQL Server 2012
Microsoft SQL Server 2014	Microsoft SQL Server 2014
Microsoft SQL Server 2016	Microsoft SQL Server 2016
Microsoft SQL Server 2017	Microsoft SQL Server 2017
Microsoft SQL Server Management Studio v17	Microsoft SQL Server Management Studio v17
Microsoft SQL Server PowerPivot for Excel	Microsoft SQL Server PowerPivot for Excel
Microsoft StreamInsight V1.0	Microsoft StreamInsight V1.0
Microsoft StreamInsight	Microsoft StreamInsight Product Family
Microsoft System Center Data Protection Manager	Microsoft System Center Data Protection Manager
Microsoft System Center DPM 2010	This category is for the products which are installed for the functioning of Microsoft System Center Data Protection Manager 2010

Microsoft System Center Virtual Machine Manager 2007	Virtual Machine Manager provides centralized administration of a virtual machine infrastructure and enables increased physical server utilization and rapid provisioning of new virtual machines by the administrator and authorized users.
Microsoft System Center Virtual Machine Manager 2008	Microsoft System Center Virtual Machine Manager 2008
Microsoft Works 8	Software Update for Microsoft Works 8
Microsoft Works 9	Software Update for Microsoft Works 9
Microsoft	Microsoft
MS Security Essentials	Microsoft Security Essentials helps protect your computer against security threats caused

	by viruses, spyware and other unwanted software.
Network Monitor 3	Category for all Network Monitor 3 updates (including 3.0, 3.1, 3.2 etc). Listed under the Network Monitor product family.
Network Monitor	Product Familty Category for Network Monitor
New Dictionaries for Microsoft IMEs	Contains the new dictionaries published by Microsoft for Microsoft IMEs, IME dictionary is a data file which usually contains the terms in a domain to help improve input accuracy in that domain.
Office 2002/XP	Office 2002/XP
Office 2003	Office 2003
Office 2007	Office 2007
Office 2010	Office 2010

Office 2013	Office 2013
Office 2016	Office 2016
Office 365 Client	Office 365 Client
Office Communications Server 2007 R2	This is the Product category for Office Communications Server 2007 R2
Office Communications Server 2007	This is for the Microsoft Office Communications Server 2007 Product family.
Office Communications Server And Office Communicator	Office Communications Server And Office Communicator Product Family Category
Office Communicator 2007 R2	This is the product category for Office Communicator 2007 R2
Office Live Add-in	Microsoft Office Live Add-in for Office is a small program that you install on your local computer to extend your Microsoft Office experience to the Web. The Office Live

	Add-in installs a new toolbar in Microsoft Office XP and Microsoft Office 2003, and new menu options in the 2007 Microsoft Office system.
Office Live	Office Live Product Family Category
Office	Office
OneCare Family Safety Installation	Contains installation files for Windows Live OneCare Family Safety, a program that allows parents to help protect their children from access to inappropriate website content and contacts.
OOBE ZDP	This category would be used by the updates that would meet the ZDP bar.
Photo Gallery Installation and Upgrades	Contains installation and upgrade files for Windows Live Photo Gallery, a tool for

	editing photos and organizing and sharing photos and videos.
Report Viewer 2005	Report Viewer 2005
Report Viewer 2008	Report Viewer 2008
Report Viewer 2010	Report Viewer 2010
SDK Components	SDK Components and Interfaces
Search Enhancement Pack	Microsoft Search Enhancement Pack helps users find exactly what they are looking for fast.
Security Essentials	Microsoft Security Essentials helps protect your computer against security threats caused by viruses, spyware and other unwanted software.
Service Bus for Windows Server 1.1	Service Bus for Windows Server 1.1
Silverlight	Microsoft Silverlight is a cross-browser plug-in for Microsoft Internet Explorer and Mozilla

	Firefox that enables simplified delivery of media and rich Internet applications blending animation, audio/video, and interactivity. Silverlight 1.0 adds interactivity to media-centric applications, including corporate communications and training applications, while maintaining compatibility and scalability with Windows Media Audio and Video for live and on-demand streaming up to HD quality.
Silverlight	Silverlight Product Family
Skype for Business Server 2015, SmartSetup	This is the product category for Skype for Business Server 2015, SmartSetup.

Skype for Business Server 2015	This is the product cateogry for Skype for Business Server 2015
Skype for Business	Skype for Business
SQL Server 2000	SQL Server Catergory Description
SQL Server 2005	SQL Server 2005
SQL Server 2008 R2	SQL Server 2008 R2
SQL Server 2008	SQL Server 2008
SQL Server 2012 Product Updates for Setup	This is the category used for updates that can be used by Product Update E2E. This category will be default category that Setup will search and will be the category that we'll ship with.
SQL Server 2014-2016 Product Updates for Setup	This is the category used for updates that can be used by Product Update E2E. This category will be default category that Setup will search and

	will be the category that we'll ship with.
SQL Server Feature Pack	The Feature Pack is a collection of standalone install packages that provide additional value for SQL Server. It includes: * Latest versions of redistributable components for SQL Server * Latest versions of add-on providers for SQL Server * Latest versions of backward compatibility components for SQL Server
SQL Server	SQL Server Product Family
System Center 1801 - Orchestrator	Product category for System Center 1801 - Orchestrator

System Center 2012 - App Controller	System Center 2012 - App Controller
System Center 2012 - Data Protection Manager	System Center 2012 - Data Protection Manager
System Center 2012 - Operations Manager	product category for System Center 2012 - Operations Manager.
System Center 2012 - Orchestrator	Create category for System Center 2012 Orchestrator
System Center 2012 - Virtual Machine Manager	System Center 2012 Virtual Machine Manager
System Center 2012 R2 - Data Protection Manager	System Center 2012 R2 - Data Protection Manager
System Center 2012 R2 - Operations Manager	Product Category for System Center 2012 R2 - Operations Manager
System Center 2012 R2 - Orchestrator	Category for System Center 2012 R2 - Orchestrator
System Center 2012 R2 - Virtual Machine Manager	Category for System Center 2012 R2 Virtual Machine Manager

System Center 2012 SP1 - App Controller	System Center 2012 SP1 - App Controller
System Center 2012 SP1 - Data Protection Manager	System Center 2012 SP1 - Data Protection Manager
System Center 2012 SP1 - Operation Manager	Product category for System Center 2012 SP1 - Operations Manager
System Center 2012 SP1 - Virtual Machine Manager	Category for System Center 2012 SP1 Virtual Machine Manager
System Center 2016 - Data Protection Manager	System Center 2016 - Data Protection Manager
System Center 2016 - Operations Manager	Product Category for System Center 2016 - Operations Manager
System Center 2016 - Orchestrator	Product category for System Center 2016 - Orchestrator
System Center 2016 - Virtual Machine Manager	Category for System Center 2016 - Virtual Machine Manager

System Center Advisor	The product category for Microsoft System Center Advisor
System Center Configuration Manager 2007	The System Center Configuration Manager 2007category will allow Service Packs and Updates to be offered to Configuration Manager product family
System Center Data Protection Manager	System Center Data Protection Manager
System Center Online	System Center Online Product Family Category
System Center Operations Manager 1807	Product Category for System Center 1801 - Operations Manager
System Center Version 1801 - Virtual Machine Manager	Product category for SCVMM Version 1801 release
System Center Virtual Machine Manager	System Center Virtual Machine Manager Product Family Category
System Center	System Center products family

Systems Management Server 2003	The Systems Management Server 2003 category will allow Service Packs 2 and Updates to be offered to Systems Management Server product family.
Systems Management Server	Systems Management Server Prodcut Family Category
Threat Management Gateway Definition Updates for Network Inspection System	Forefront TMG Network Inspection System (NIS) helps guard against intrusion attempts targeting known and newly discovered vulnerabilities in network protocols. As a security best practice, NIS signatures should be kept up to date.
TMG Firewall Client	Main category for TMG Firewall Client (RTM)
Virtual PC	Microsoft Virtual PC all versions till VPC 2007 SP1

Virtual Server	Virtual Server Product Family Category
Virtual Server	Category - app rule detects virtual server from registry key
Visual Studio 2005	Visual Studio 2005
Visual Studio 2008	Visual Studio 2008
Visual Studio 2010 Tools for Office Runtime	Visual Studio 2010 Tools for Office Runtime
Visual Studio 2010 Tools for Office Runtime	Visual Studio 2010 Tools for Office Runtime
Visual Studio 2010	Visual Studio 2010
Visual Studio 2012	Visual Studio 2012
Visual Studio 2013	Visual Studio 2013
Windows 10 and later drivers	Windows 10 and later drivers
Windows 10 and later upgrade & servicing drivers	Windows Drivers
Windows 10 Anniversary Update and Later Servicing Drivers	Windows 10 Anniversary Update and Later Servicing Drivers

Windows 10 Anniversary Update and Later Upgrade & Servicing Drivers	Windows 10 Anniversary Update and Later Upgrade & Servicing Drivers
Windows 10 Creators Update and Later Servicing Drivers	Windows 10 Creators Update and Later Servicing Drivers
Windows 10 Creators Update and Later Servicing Drivers	Windows 10 Creators Update and Later Servicing Drivers
Windows 10 Creators Update and Later Upgrade & Servicing Drivers	Windows 10 Creators Update and Later Upgrade & Servicing Drivers
Windows 10 Dynamic Update	Windows 10 Dynamic Update
Windows 10 Fall Creators Update and Later Servicing Drivers	Windows 10 Fall Creators Update and Later Servicing Drivers
Windows 10 Fall Creators Update and Later Upgrade & Servicing Drivers	Windows 10 Fall Creators Update and Later Upgrade & Servicing Drivers
Windows 10 Feature On Demand	This category would be used by theWindows 10 Feature On Demand content.

Windows 10 GDR-DU FOD	This category will be used for GDR-DU Feature On Demand updates
Windows 10 GDR-DU LP	This category will be used for GDR-DU Language Pack updates
Windows 10 GDR-DU	Windows 10 GDR-DU
Windows 10 Language Interface Packs	Category for Windows 10 Language Interface Packs
Windows 10 Language Packs	Category for Windows 10 Language Packs
Windows 10 LTSB	Windows 10 LTSB
Windows 10 S and Later Servicing Drivers	Windows 10 S and Later Servicing Drivers
Windows 10 S Version 1709 and Later Servicing Drivers for testing	Windows 10 S Version 1709 and Later Servicing Drivers for testing
Windows 10 S Version 1709 and Later Upgrade & Servicing Drivers for testing	Windows 10 S Version 1709 and Later Upgrade & Servicing Drivers for testing

Windows 10 S Version 1803 and Later Servicing Drivers	Windows 10 S Version 1803 and Later Servicing Drivers
Windows 10 S Version 1803 and Later Upgrade & Servicing Drivers	Windows 10 S Version 1803 and Later Upgrade & Servicing Drivers
Windows 10 version 1803 and Later Servicing Drivers	Windows 10 version 1803 and Later Servicing Drivers
Windows 10 Version 1803 and Later Upgrade & Servicing Drivers	Windows 10 Version 1803 and Later Upgrade & Servicing Drivers
Windows 10	Windows 10
Windows 2000	Windows 2000
Windows 7	Windows 7 Category
Windows 8 Dynamic Update	This category is used by Windows Setup and contains updates to the Windows 8 Setup binaries, as well as critical security updates and drivers for the operating system that is being installed. WSUS administrators do not

	need to select this category.
Windows 8 Embedded	Detects Windows 8 versions of Embedded
Windows 8 Language Interface Packs	Category for Windows 8 language interface packs
Windows 8 Language Packs	Windows 8 Language Packs
Windows 8.1 and later drivers	Windows 8.1 and later drivers
Windows 8.1 Drivers	Windows 8.1 Drivers
Windows 8.1 Dynamic Update	Windows 8.1 Dynamic Update
Windows 8.1 Language Interface Packs	Category for Windows 8.1 language interface packs
Windows 8.1 Language Packs	Category for Windows 8.1 Language Packs
Windows 8.1	Windows 8.1
Windows 8	Windows 8
Windows Admin Center	Product Family for Windows Admin Center
Windows Admin Center	Windows Admin Center
Windows Azure Pack - Web Sites	Windows Azure Pack - Web Sites

Windows Azure Pack: Admin API	Windows Azure Pack: Admin API
Windows Azure Pack: Admin Authentication Site	Windows Azure Pack: Admin Authentication Site
Windows Azure Pack: Admin Site	Windows Azure Pack: Admin Site
Windows Azure Pack: Configuration Site	Windows Azure Pack: Configuration Site
Windows Azure Pack: Microsoft Best Practice Analyzer	Windows Azure Pack: Microsoft Best Practice Analyzer
Windows Azure Pack: Monitoring Extension	Windows Azure Pack: Monitoring Extension
Windows Azure Pack: MySQL Extension	Windows Azure Pack: MySQL Extension
Windows Azure Pack: PowerShell API	Windows Azure Pack: PowerShell API
Windows Azure Pack: SQL Server Extension	Windows Azure Pack: SQL Server Extension
Windows Azure Pack: Tenant API	Windows Azure Pack: Tenant API
Windows Azure Pack: Tenant Authentication Site	Windows Azure Pack: Tenant Authentication Site

Windows Azure Pack: Tenant Public API	Windows Azure Pack: Tenant Public API
Windows Azure Pack: Tenant Site	Windows Azure Pack: Tenant Site
Windows Azure Pack: Usage Extension	Windows Azure Pack: Usage Extension
Windows Azure Pack: Web App Gallery Extension	Windows Azure Pack: Web App Gallery Extension
Windows Azure Pack: Web Sites	Create high-scale, multi-tenant website hosting services
Windows Azure Pack	Windows Azure Pack products family
Windows Defender	Windows Defender helps protect your computer against pop-ups, slow performance, and security threats caused by spyware and other unwanted software.
Windows Dictionary Updates	Windows Dictionary Updates for Simplified Chinese, Japanese Input Methods

	and English (US) Bing trendy words.
Windows Embedded Developer Update	Category for updates to WEDU client itself.
Windows Embedded Standard 7	Windows Embedded Standard 7
Windows Embedded	TLC for windows embedded team
Windows Essential Business Server 2008 Setup Updates	Non security updates applied to Windows Essential Business Server 2008 at MU phase 0, or critical security update released by EBS team
Windows Essential Business Server 2008	Updates for Windows Essential Business Server 2008.
Windows Essential Business Server Preinstallation Tools	Updates for the Windows Essential Business Server Preinstallation Tools.
Windows Essential Business Server	Windows Essential Business Server Product Family Category

Windows GDR-Dynamic Update	Windows GDR-Dynamic Update
Windows Internet Explorer 7 Dynamic Installer	The Window Internet Explorer 7 setup application uses this category to find updates to download and install during installation of Internet Explorer 7. More information about the updates in this category is available at http://support.microsoft.com/kb/924568. If this category is not selected, Internet Explorer 7 will install successfully but will not download and install any applicable updates during the installation.
Windows Internet Explorer 8 Dynamic Installer	The Window Internet Explorer 8 setup application uses this category to find updates to download and install during installation

	of Internet Explorer 8. More information about the updates in this category is available at http://support.microsoft.com/kb/948564. If this category is not selected, Internet Explorer 8 will install successfully but will not download and install any applicable updates during the installation.
Windows Live Toolbar	Contains the installation files for Windows Live Toolbar, an Internet Explorer extension that provides search, anti-virus protection, customizable buttons, and quick access to maps and other information.
Windows Live	Get quick access to Bing and MSN, as well as handy

	tools for online safety and productivity.
Windows Live	Contains updates and upgrades for all Windows Live programs, some of which may be distributed via Automatic Update. It is strongly recommended that you approve all categories and updates in the Windows Live product family as a group, because files may contain elements needed to successfully install or update multiple Windows Live programs at once.
Windows Live	Windows Live
Windows Media Dynamic Installer	Windows Media Setup Updates
Windows Next Graphics Driver Dynamic update	This hidden category will be scanned during 8.1 upgrade from Store by

	DU by ignoring currently installed drivers. Only Graphics drivers should be published to this category. The best graphics driver available for Blue for the users system will be downloaded and included in the upgrade image.
Windows RT 8.1	Windows RT 8.1
Windows RT	Windows RT
Windows Safe OS Dynamic Update	This category will be used to address the Safe OS DU scenario.
Windows Server 2003, Datacenter Edition	Windows Server 2003, Datacenter Edition
Windows Server 2003	Windows Server 2003
Windows Server 2008 R2	Windows Server 2008 R2 Category

Windows Server 2008 Server Manager Dynamic Installer	Windows Server 2008 Server Manager uses this category to find and download updates.
Windows Server 2008	Windows Server 2008
Windows Server 2012 Language Packs	Windows Server 2012 Language Packs
Windows Server 2012 R2 and later drivers	Windows Server 2012 R2 and later drivers
Windows Server 2012 R2 Drivers	Windows Server 2012 R2 Drivers
Windows Server 2012 R2 Language Packs	Category for Windows Server 2012 R2 Language Packs
Windows Server 2012 R2	Windows Server 2012 R2
Windows Server 2012	Windows Server 2012
Windows Server 2016 and Later Servicing Drivers	Windows Server 2016 and Later Servicing Drivers
Windows Server 2016	Windows 10 Server
Windows Server Drivers	Windows Server Drivers
Windows Server Manager ? Windows Server Update Services (WSUS) Dynamic Installer	Windows Server Manager uses this category to find and

	download updates for Windows Server Update Services (WSUS).
Windows Server Solutions Best Practices Analyzer 1.0	Windows Server Solutions Best Practices Analyzer scans and finds problems on Windows Small Business Server 2011, Windows Storage Server 2008 Essentials R2, and Windows Multipoint Server 2011.
Windows Server Technical Preview Language Packs	Category for Windows Server Technical Preview Language Packs
Windows Small Business Server 2003	This is the top level category from Windows Small Business Server 2003
Windows Small Business Server 2008 Migration Preparation Tool	Updates offered through this category will apply only to computers running Migration Preparation Tool for Windows Small Business Server 2008.

Windows Small Business Server 2008	Category for Windows Small Business Server 2008
Windows Small Business Server 2011 Standard	SBS7
Windows Small Business Server	Windows Small Business Server Product Family
Windows Ultimate Extras	Category for Windows Ultimate Extras
Windows Vista Dynamic Installer	Windows Vista Dynamic Updates category.
Windows Vista Ultimate Language Packs	Category for Windows Vista Ultimate Language Packs
Windows Vista	Windows Vista
Windows XP 64-Bit Edition Version 2003	Windows XP 64-Bit Edition Version 2003
Windows XP Embedded	Category for XP WEPOS and POSReady 2009 Embedded Machines for SP3
Windows XP x64 Edition	Windows XP x64 Edition
Windows XP	Windows XP

Windows	Windows
Works 6-9 Converter	Software Update for Microsoft Works 6-9 Converter
Works	Works Product Family Category
Writer Installation and Upgrades	Contains installation and upgrade files for Windows Live Writer, a program designed to make it easier to edit and publish rich content to your blog.