An Automated System for Rapid and Secure Device Sanitization

By Ralph S. LaBarge

B.S. in Electrical Engineering, May 1979, University of Delaware
M.S.in Computer Science, May 1984, Johns Hopkins University
M.S. in Electrical Engineering, May 1988 Johns Hopkins University
M.S. in Information Assurance, December 2012, Johns Hopkins University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

May 18, 2014

Dissertation directed by

Shahram Sarkani
Professor of Engineering Management and Systems Engineering

and

Thomas Andrew Mazzuchi
Professor of Engineering Management and Systems Engineering & of Decision Sciences

The School of Engineering and Applied Science of the George Washington University certifies that Ralph S. LaBarge has passed the Final Examination for the degree of Docton of Philosophy as of April 4, 2104. This is the final and approved form of the dissertation.

An Automated System for Rapid and Secure Device Sanitization

Ralph S. LaBarge

Dissertation Research Committee:

Shahram Sarkani, Professor of Engineering Management and Systems Engineering, Dissertation Co-Director

Thomas Mazzuchi, Professor of Engineering Management and Systems Engineering & of Decision Sciences, Dissertation Co-Director

John Bischoff, Professorial Lecturer in Engineering Management and System Engineering, Committee Member

Jason Dever, Professorial Lecturer in Engineering Management and Systems Engineering, Committee Member

E. Lile Murphree, Professor of Engineering Management, Committee Member

Dedication

This dissertation is dedicated to my wife Jan and my three daughters Kim, Jen and Cindy. Without their understanding and support I would never have been able to complete this effort. I am sure they thought I was a crazy to enroll in a PhD program at the age of 57, but they always made it easy for me to attend class, work on assignments, write a paper, and finish this dissertation.

I would also like to dedicate this dissertation to Dr. James Zalucki, Dr. Jon Minford and Dr. Armando Sardi. Without their skill and dedication to helping cancer patients I would probably not be here today.

Finally, I would like to dedicate this dissertation to Razi Dianat, a GWU EMSE classmate who passed away before he could complete his PhD.

Razi Dianat – February 12, 1959 - March 14, 2013. A wonderful mind, a caring soul, and a great classmate. You have been missed!

Acknowledgments

Dr. Shahram Sarkani, Professor of Engineering Management and Systems Engineering; Academic Director, EMSE Off-Campus Programs

Dr. Thomas Mazzuchi, Professor of Engineering Management and Systems Engineering & of Decision Sciences; Chair, Department of Engineering Management and Systems Engineering

Dr. John Bischoff, Professorial Lecturer in Engineering Management and System Engineering, Committee Member

Dr. Jason Dever, Professorial Lecturer in Engineering Management and Systems Engineering, Committee Member

Dr. E. Lile Murphree, Professor of Engineering Management, Committee Chair

This work was sponsored by the Defense Advanced Research Projects Agency Strategic Technology Office (STO), Program: National Cyber Range, ARPA Order No. Y044/02, Program Code: 9B30, Issued by DARPNCMO under Contract No. HR0011-10-C-0039. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressly or implied, of the Defense Advanced Research Projects Agency, the National Security Agency/Central Security Service, or the U.S. Government.

Abstract

An Automated System for Rapid and Secure Device Sanitization

Public and private organizations face the challenges of protecting their networks from cyber-attacks, while reducing the amount of time and money spent on Information Technology. Organizations can reduce their expenditures by reusing server, switch and router hardware, but they must use reliable and efficient methods of sanitizing these devices before they can be redeployed. The sanitization process removes proprietary, sensitive or classified data, as well as persistent malware from a device prior to reuse. The Johns Hopkins University Applied Physics Laboratory has developed an automated, rapid, and secure method for sanitizing servers, switches and routers. This sanitization method was implemented and tested on several different types of network devices during the Cyber Measurement & Analysis Center project, which was funded under Phases I and II of the DARPA National Cyber Range program. The performance of the automated sanitization system was excellent with an order of magnitude reduction in the time required to sanitize servers, routers and switches, and a significant improvement in the effectiveness of the sanitization process through the addition of persistent malware removal.

Table of Contents

Dedication ……...……..iii
Acknowledgments …….……iv
Abstract ……...v
Table of Contents…….……...vi
List of Figures……...viii
List of Tables ……ix
List of Acronyms ……x
Chapter 1 – Introduction & Working Hypotheses ……..1
1.1 Dissertation Organization ……...1
1.2 An Introduction to Device Sanitization ……..2
1.3 Sanitize to Protect Sensitive, Proprietary or Classified Data …….5
1.4 Sanitize to Remove Persistent Malware ……6
1.5 Sanitize to Restore a Device to a Trusted State ……..7
Chapter 2 – Literature Review ……9
2.1 Summary of Device Sanitization Literature……9
2.2 Limitations of Device Sanitization Literature……19
2.3 Improving Device Sanitization …….20
Chapter 3 –Sanitization of Server, Switch and Router Components ……21
3.1 Sanitization of Volatile Memory……...21
3.2 Sanitization of Non-Volatile Memory……...21
3.3 Sanitization of Hard Disk Drives……...22
3.4 Sanitization of Flash Memory…….....24
3.5 Slack Space……24
Chapter 4 –Sanitizing a Single Server, Switch or Router …….25
4.1 Step 1: Establish a Root of Trust……...25
4.2 Step 2: Sanitize the Device Components in the Proper Order……...28
4.3 Step 3: Restore the Device to a Trusted State……29
4.4 Step 4: Validate the Sanitization Results……...30
4.5 Step 5: Sanitize any Equipment Used During the Sanitization Process……30

Chapter 5 – An Automated Sanitization System Architecture ……32
Chapter 6 – An Implementation of the Automated Sanitization System Architecture….36
6.1 CMAC Sanitization System Design ……36
6.2 CMAC Sanitization Controller ……38
6.2.1 Dell Server BIOS Sanitization ……...45
6.2.2 Dell Server iDRAC Sanitization ……45
6.2.3 Dell Server iDRAC Authentication ……...46
6.2.4 Dell Server LOM Sanitization ……...46
6.2.5 Dell Server RAID Sanitization ……..47
6.2.6 Dell Server TPM Sanitization ……47
6.3 Dell Server Sanitization Modes of Operation……47
6.4 Dell Server Hard Drive Sanitization ……49
6.5 Dell Server Sanitization Modes of Operation Details ……..50
6.6 Cisco Switch & Router Sanitization ……52
6.7 Raritan Serial Console Manager Switch Sanitization ……...53
6.8 Sanitize the Sanitization System Devices ……54
Chapter 7 – CMAC Sanitization System Performance ……56
Chapter 8 – Summary & Conclusions ……..58
8.1 The CMAC Sanitization System Met Its Requirements ……...59
8.2 Issues Related to Implementing the Automated Sanitization System Architecture..59
8.2 Dissertation Hypotheses Conclusions……61
Chapter 9 – Areas for Future Research ……62
References ……64
Appendix A – Sample Statement of Volatility ……68
Appendix B – Dell Server Sanitization Software Design Diagrams ……70
Appendix C – Sample Dell Server Sanitization Script ……73
Appendix D – Sample Dell R610 Back Up Log ……...92
Appendix E – Sample Dell R610 Sanitization Log ……..95

List of Figures

Figure 1-1: NIST Sanitization and Disposition Flow Chart……4
Figure 2-1: Fast Secure Erase Hard Disk Sanitization Process…….17
Figure 4-1: Trusted Platform Module Block Diagram……..27
Figure 5-1: Automated Sanitization System Architecture…….33
Figure 6-1: CMAC Sanitization System……37
Figure 6-2: Expedite XES7330 Single Board Computer Block Diagram…….39
Figure 6-3: PXE Boot Process……...40
Figure 6-4: Dell R610 Server Internal View ……42
Figure B-1: Dell Server Sanitization Software State Diagram …….70
Figure B-2: Dell Server Sanitization Software Backup Mode Flow Chart ……..71
Figure B-3: Dell Server Sanitization Software Flow Chart ……..72

List of Tables

Table 2-1: Multiple Pass Overwriting Technique ……10
Table 2-2: NIST Special Publication 800-88 Sanitization Recommendations ……12
Table 2-3: Wiping Experiment Results ……13
Table 2-4: Defense Security Service Sanitization Matrix ……16
Table 2-5: Hard Drive Sanitization Key Performance Metrics……17
Table 2-6: SSD ATA-SE Test Results ……..18
Table 6-1: Dell R610 Non-Volatile Memory Areas ……43
Table 6-2: Dell R610 Sanitization Scripts & Programs……44

List of Acronyms

ACPI - Advanced Configuration and Power Interface

AES - Advanced Encryption Standard

ATA - Advanced Technology Attachment

ATA-SE – Advanced Technology Attachment Secure Erase

BIOS - Basic Input/Output System

CMAC - Cyber Measurement & Analysis Center

CMOS - Complementary metal–oxide–semiconductor

DARPA - Defense Advanced Research Projects Agency

DHCP - Dynamic Host Configuration Protocol

DHHS – US Department of Health and Human Services

DOD – US Department of Defense

DOS – Disk Operating System

DDR SDRAM - Double Data Rate Synchronous Dynamic Random Access Memory

DRAM - Dynamic Random Access Memory

DSS – Defense Security Service

EEPROM - Electrically Erasable Programmable Read-Only Memory

EPROM - Erasable Programmable Read-Only Memory

FIPS – Federal Information Processing Standards

FRU - Field Replaceable Unit

iDRAC – Integrated Dell Remote Access Controller

IP – Internet Protocol

IPMI - Intelligent Platform Management Interface

IOS - Internetwork Operating System

JHU/APL – Johns Hopkins University Applied Physics Laboratory

KVM – Keyboard/Video/Mouse

LAN – Local Area Network

LOM – LAN on Motherboard

MAC – Media Access Control

MBR - Master Boot Record

MFM – Modified Frequency Modulation

NCR – National Cyber Range

NFS – Network File System

NISPOM - National Industrial Security Program Operating Manual

NIST – National Institute of Standards and Technology

NSA – National Security Agency

NVRAM - Non-Volatile Random Access Memory

PCI - Peripheral Component Interconnect

PCR - Platform Configuration Registers

PDU – Power Distribution Unit

PERC – PowerEdge Raid Controller

PROM - Programmable Read-Only Memory

PXE - Preboot Execution Environment

RAID - Redundant Array of Independent Disks

RLL – Run Length Limited

ROM - Read-Only Memory

SAS – Serial Attached SCSI

SCM – Serial Console Management

SCSI - Small Computer System Interface

SED - Self-Encrypting Drives

SRAM - Static Random Access Memory

STO - Strategic Technology Office

TCP – Transmission Control Protocol

TFTP – Trivial File Transfer Protocol

TPM - Trusted Platform Module

TS//SCI – Top Secret - Sensitive Compartmented Information

UDP - User Datagram Protocol

USB - Universal Serial Bus

Chapter 1 – Introduction & Working Hypotheses

This dissertation focuses on the issue of sanitization of information technology devices, such as servers, routers and switches, from a system perspective. Prior to this dissertation the majority of the published literature related to device sanitization discussed how to sanitize a single device component such as a hard drive, random access memory, or non-volatile memory. Recognizing that modern information technology devices are composed of many different forms of volatile and non-volatile memory components, and operate in large scale systems that include many different types of servers, switches and routers is the key to understanding the sanitization problem from a systems engineering perspective.

The working hypotheses for this dissertation are:

H_1: All sensitive, proprietary or classified information, as well as persistent malware can be removed from servers, switches and routers through the use of a well-defined sanitization process.

H_2: The amount of time required to sanitize servers, switches and routers can be significantly reduced by automating and parallelizing a well-defined sanitization process within a sanitization system.

The null hypothesis for this dissertation is:

H_0: Some sensitive, proprietary or classified information, or persistent malware will remain on servers, switches and routers after the completion of a well-defined sanitization process.

1.1 Dissertation Organization

This dissertation includes nine chapters, a References section and five Appendices. Chapter 1 provides an introduction and overview of the topic of device sanitization. Chapter 2 provides a review of the literature related to device sanitization. Chapter 3 discusses how individual components within a server, switch or router can be properly sanitized. Chapter 4 discusses how all of the components within a server, switch or router can be properly sanitized. Chapter 5 presents a framework for an Automated Sanitization System that can be used to sanitize a variety of server, router and switch device types with limited or no human interaction. Chapter 6 presents the Cyber Measurement and Analysis Center (CMAC) Sanitization System, which is an implementation of the Automated Sanitization System framework presented in Chapter 5. Chapter 7 reviews the performance of the CMAC sanitization system during system integration and testing efforts. Chapter 8 provides a summary of the device sanitization process and conclusions about how the CMAC sanitization system was able to meet its requirements. Chapter 9 discusses areas for future research within the field of device sanitization. Appendix A provides a sample Statement of Volatility. Appendix B provides a state diagram and flow charts for the Dell Server Sanitization process software design. Appendix C provides a sample shell script used to automate the Dell server sanitization process. Appendix D provides a sample log from an automated Dell server backup process. Appendix E provides a sample log from an automated Dell server sanitization process.

1.2 An Introduction to Device Sanitization

The National Institute of Standards and Technology (NIST) defines sanitization as "the general process of removing data from storage media, such that there is reasonable assurance that the data may not be easily retrieved and reconstructed" (Kissel et al. 2006). Similarly the Department of Defense (DoD) National Industrial Security Program Operating Manual (NISPOM) defines sanitization as "the process of removing the data from media before reusing the media in an environment that does not provide an acceptable level of protection for the data that was in the media before sanitizing" (Deutch 2004).

NIST defines four classes of sanitization techniques: *disposal*, *clearing*, *purging*, and *destroying* as shown in Figure 1-1. *Disposal* is "the act of discarding media with no other sanitization considerations. This is most often done by paper recycling containing non-confidential information but may also include other media."; *clearing* is "a level of media sanitization that would protect the confidentiality of information against a robust keyboard attack. Simple deletion of items would not suffice for clearing. Clearing must not allow information to be retrieved by data, disk, or file recovery utilities. It must be resistant to keystroke recovery attempts executed from standard input devices and from data scavenging tools"; *purging* is "a media sanitization process that protects the confidentiality of information against a laboratory attack. For some media, clearing media would not suffice for purging. However, for ATA disk drives manufactured after 2001 (over 15 GB) the terms clearing and purging have converged"; and *destruction* is "is the ultimate form of sanitization. After media are destroyed, they cannot be reused as originally intended. Physical destruction can be accomplished using a variety of methods,

including disintegration, incineration, pulverizing, shredding, and melting". (Kissel et al. 2006)

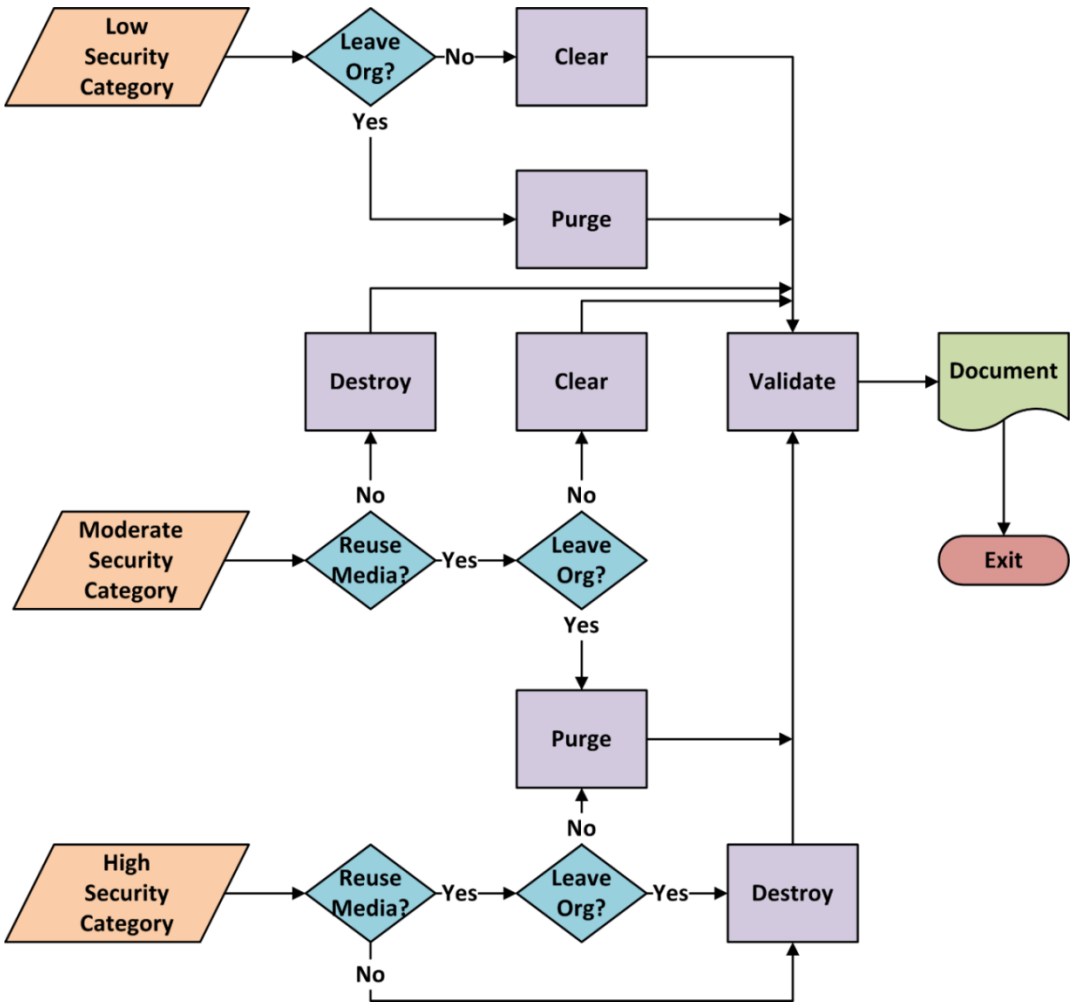


Figure 1-1: NIST Sanitization and Disposition Flow Chart (Kissel et al. 2006)

In general, higher levels of sanitization are recommended if a device will be leaving an organization's control. There are several other factors that should be considered before determining which sanitization method is appropriate for a particular type of device. These factors include the cost of sanitizing the device, the amount of time required to sanitize the device, the need to reuse the device after sanitization, and the

risks associated with the disclosure of any sensitive, proprietary or classified data that may be stored on the device.

Network servers, switches and routers are complex devices. These devices often contain high capacity hard disk drives capable of storing one TByte or more of data, and flash memory capable of storing many GBytes of data. They also include many different areas of Non-Volatile Random Access Memory (NVRAM) used to store the device's Basic Input/Output System (BIOS) software, hard disk controller firmware, network controller firmware, and other types of firmware. A common characteristic shared by these device components is that the information or firmware they store is persistent, even after the device is rebooted, or power cycled.

There are several reasons for sanitizing a device before it is decommissioned or reused including: (1) to protect sensitive, proprietary or classified data; (2) to remove persistent malware; and (3) to restore a device to a trusted state.

1.3 Sanitize to Protect Sensitive, Proprietary or Classified Data

Several United States laws and regulations require that sensitive, proprietary or classified data stored on electronic devices be properly sanitized if the device is to be reused or decommissioned. These laws include the Health Insurance Portability and Accountability Act, the Sarbanes-Oxley Act, the Fair and Accurate Credit Transactions Act of 2003, and the Health Information Technology for Economic and Clinical Health Act (Hughes, Coughlin and Commins 2009). Failure to comply with the data protection and sanitization requirements of applicable laws and regulations could lead to civil and/or criminal penalties, damage to an organization's reputation, and the loss of sensitive, proprietary or classified information.

There have been many well documented failures to properly sanitize devices which have resulted in data loss. Improperly sanitized surplus computers sold by the State of Georgia during 2006 resulted in the disclosure of names, credit card numbers, birth dates and social security numbers of many Georgia citizens (Privacy Rights Clearinghouse 2006). In 2011 Lebanon Internal Medicine Associates disposed of an improperly sanitized network server resulting in the disclosure of 55,000 patient health records (US DHHS 2011). Device sanitization failures have also occurred at the United States Veterans Administration Medical Center, the Pennsylvania Department of Labor and Industry, Purdue University, and others (Garfinkel and Shelat 2003).

1.4 Sanitize to Remove Persistent Malware

McAfee Labs recently disclosed they have over 113 million unique samples in their malware database (McAfee Labs 2013). Many of these malware samples are rootkits which are designed to evade detection and persist on an infected computer for an indefinite period of time. Some malware is designed to infect a host computer's BIOS or firmware in order to remain persistent even after the hard disk drive has been removed or sanitized. For example the Mebroni virus, first detected in September 2011, can infect the system BIOS and hard drive's Master Boot Record (MBR) of a host computer that uses a Phoenix Technologies Award BIOS (Ge 2011). Properly sanitizing the computer's hard drive will remove the Mebroni virus from the Master Boot Record, but a copy of the virus will still be active in the system's BIOS.

Persistent malware can hide in computer firmware. John Heasman has demonstrated the viability of firmware rootkits in both Advanced Configuration and Power Interface (ACPI) and Peripheral Component Interconnect (PCI) expansion card firmware

(Heasman 2006). From a hacker's perspective, BIOS based malware and firmware rootkits are very attractive since they are difficult to detect, difficult to remove, can survive a system reboot or power cycle, do not leave a malware signature on the system's hard disk drive, and often have system or root level privileges.

Persistent malware can also hide in the firmware of embedded devices like network routers, switches and firewalls. Cisco has released over 600 Security Advisories, Responses, and Notices for their network devices, most of which run the IOS operating system (Cisco 2013). Rootkits that can take advantage of vulnerabilities in Cisco IOS were demonstrated at the Black Hat conference in 2005 (Lynn 2005) and at the EuSecWest conference in 2008 (Muñiz 2008). Historically malware designed to take advantage of IOS vulnerabilities has been difficult to implement since there are so many variations of this embedded operating system. Recently a two stage attack against Cisco routers has been proposed that does not require a-priori knowledge of the target device's IOS version (Cui, Kataria and Stolfo 2011), making it easier to attack these types of devices.

1.5 Sanitize to Restore a Device to a Trusted State

Ideally devices that are based on the same hardware (i.e. the same motherboard, add-in cards, etc.) should also share the same BIOS, firmware, operating system software, and patch levels. Having homogeneous device configurations provides a solid foundation upon which to implement an organization's information assurance and information security policies and procedures. A well designed sanitization process can be used to restore a device to a trusted state, thus making it easier to implement an effective configuration management process within an organization.

Another reason to restore a device to a trusted state is to mitigate supply chain attacks where malware can be introduced into a device's BIOS, firmware, or operating system either at the factory or at some point between the factory and the end user. In 2010 several Dell PowerEdge R410 server motherboards were discovered to have the W32.Spybot worm resident in flash memory (Narine 2010). Dell discovered the malware, and removed the infected motherboards from their supply chain; however some of these motherboards had already been installed in systems connected to a variety of networks.

While supply chain attacks are not as well-known as more traditional forms of malware, they represent a real threat that must be addressed. Since 2008 there have been dozens of reported supply chain attacks resulting in pre-infected servers, laptops, routers, USB memory sticks, telephones, cameras and even credit card terminals (Sachs 2010). Some examples of malware delivered via supply chain attacks include the Mariposa botnet client, the Confiker worm, the Sality worm, the Arugizer Trojan, the Aspxor worm, and the Perlovga worm.

Chapter 2 – Literature Review

2.1 Summary of Device Sanitization Literature

Over the past three decades a number of papers and standards have been published that are directly related to the sanitization of servers, switches and routers.

In 1995 the National Industrial Security Program Operating Manual, also known as the DOD 5220.22.M standard, recommended that hard drives and random access memory be sanitized using a four step process, often referred to as the DOD 5220 Block Erase. During this sanitization process every byte of data is overwritten with a specific character, overwritten again with the inverse of that character, overwritten a third time with random data, and then verified (Deutch 2004).

In 1996 Peter Gutmann argued that using an automated Magnetic force microscopy system it could be "possible to collect an image of the entire disk by changing the MFM controller software" (Gutmann 1996). He proposed that a multiple pass overwrite technique was required to properly sanitize hard drives and random access memory. Gutmann's recommended overwrite technique is provided in Table 2-1. His proposed method was designed to properly sanitize hard drives that used two popular forms of run-length-limited (RLL) encoding, including RLL (1, 7) and RLL (2, 7), as well as drives that used modified FM (MFM) encoding. At the time Gutmann published his paper these were the three most common encoding techniques used to store data on hard disk drives. Gutmann recommended only using those overwrite passes that are directly applicable to the type of encoding used on a specific hard drive. Thus not all 35 passes are required to properly sanitize a hard disk drive. One of Gutmann's primary conclusions was that

"good scrubbing with random data will do about as well as can be expected" (Gutmann 1996).

Table 2-1: Multiple Pass Overwriting Technique (Gutmann, 1996)

Overwrite Data				
Pass	**Data Written**	**Encoding Scheme Targeted**		
1	Random			
2	Random			
3	Random			
4	Random			
5	01010101 01010101 01010101 0x55	(1,7) RLL		MFM
6	10101010 10101010 10101010 0xAA	(1,7) RLL		MFM
7	10010010 01001001 00100100 0x92 0x49 0x24		(2,7) RLL	MFM
8	01001001 00100100 10010010 0x49 0x24 0x92		(2,7) RLL	MFM
9	00100100 10010010 01001001 0x24 0x92 0x49		(2,7) RLL	MFM
10	00000000 00000000 00000000 0x00	(1,7) RLL	(2,7) RLL	
11	00010001 00010001 00010001 0x11	(1,7) RLL		
12	00100010 00100010 00100010 0x22	(1,7) RLL		
13	00110011 00110011 00110011 0x33	(1,7) RLL	(2,7) RLL	
14	01000100 01000100 01000100 0x44	(1,7) RLL		
15	01010101 01010101 01010101 0x55	(1,7) RLL		MFM
16	01100110 01100110 01100110 0x66	(1,7) RLL	(2,7) RLL	
17	01110111 01110111 01110111 0x77	(1,7) RLL		
18	10001000 10001000 10001000 0x88	(1,7) RLL		
19	10011001 10011001 10011001 0x99	(1,7) RLL	(2,7) RLL	
20	10101010 10101010 10101010 0xAA	(1,7) RLL		MFM
21	10111011 10111011 10111011 0xBB	(1,7) RLL		
22	11001100 11001100 11001100 0xCC	(1,7) RLL	(2,7) RLL	
23	11011101 11011101 11011101 0xDD	(1,7) RLL		
24	11101110 11101110 11101110 0xEE	(1,7) RLL		
25	11111111 11111111 11111111 0xFF	(1,7) RLL	(2,7) RLL	
26	10010010 01001001 00100100 0x92 0x49 0x24		(2,7) RLL	MFM
27	01001001 00100100 10010010 0x49 0x24 0x92		(2,7) RLL	MFM
28	00100100 10010010 01001001 0x24 0x92 0x49		(2,7) RLL	MFM
29	01101101 10110110 11011011 0x6D 0xB6 0xDB		(2,7) RLL	
30	10110110 11011011 01101101 0xB6 0xDB 0x6D		(2,7) RLL	
31	11011011 01101101 10110110 0xDB 0x6D 0xB6		(2,7) RLL	
32	Random			
33	Random			
34	Random			
35	Random			

In 2001 the Advanced Technology Attachment (ATA) hard drive standard was modified to include a low level secure erase function called ATA-SE. The ATA-SE function overwrites all data on a hard drive in a single pass, including blocks that have been marked as bad (Hughes and Coughlin 2006).

In 2001 NIST published the FIPS140-2 standard which defines security requirements for cryptographic modules used in information technology devices (NIST 2001). This standard defines four increasing levels of security (Levels 1, 2, 3 and 4). Security Level 1 is the lowest defined level which requires a basic cryptographic module but no physical security measures. Security Level 2 adds requirements for tamper-evidence which will provide evidence of an attempt to access or modify the cryptographic module. Security Level 3 add requirements for identify-based authentication mechanism to prevent unauthorized remote access to the cryptographic module. Security Level 4 adds requirements to protect the cryptographic module from changes in environmental conditions or fluctuations outside of the modules normal operating conditions.

In 2001 NIST also published FIPS 197, the Advanced Encryption Standard (AES), which defines standard algorithms used to encrypt electronic data (FIPS PUB 2001). The AES standard uses a symmetric block cipher with a block length of 128 bits, using 128-bit, 192-bit or 256-bit encryption keys. A brute-force attack against AES-128 encrypted data requires more than 1.8×10^{18} years using a supercomputer capable of performing more than 10^{15} floating point operations per second (Arora 2012). This amount of time is longer than the age of the universe, so it is impractical to attempt to break AES using brute-force methods. A biclique cryptanalysis based attack has reduced the amount of

time required to break AES-128 by a factor 5 for some versions of AES-128 (Bogdanov et al. 2011), however the length of time required to recover the key is still impractical using today's technology.

In 2003 the National Security Agency (NSA) announced that the AES-128 algorithm could be used to encrypt SECRET data and the AES-256 algorithm could be used to encrypt TOP SECRET data, provided the AES devices used had been approved by the NSA (Hathaway 2003).

In 2006 NIST approved the use of the ATA-SE function for sanitization of hard drives (Kissel et al. 2006). NIST Special Publication 800-88 Appendix A provides recommendations for the sanitization of different types of magnetic media. Table 2-2 provides a summary of the NIST recommendations for sanitizing hard disk drives, and memory devices.

Table 2-2: NIST Special Publication 800-88 Sanitization Recommendations (Kissel et al. 2006)

Media Type	Sanitization Method
ATA Hard Drives, USB Removable Media	Purge using Secure Erase. The Secure Erase software can be download from the University of California, San Diego (UCSD) CMRR site.
SCSI Hard Drives	Purge hard disk drives by either purging the hard disk drive in an NSA/CSS-approved automatic degausser or by disassembling the hard disk drive and purging the enclosed platters with an NSA/CSS-approve
Flash Drives	Overwrite media by using agency-approved and validated overwriting technologies/methods/tools
Dynamic Random Access Memory (DRAM)	Purge DRAM by powering off and removing the battery (if battery backed).
Electronically Erasable PROM (EEPROM)	Overwrite media by using agency approved and validated overwriting technologies/methods/tools
Flash Memory	Overwrite media by using agency approved and validated overwriting technologies/methods/tools

Non Volatile RAM (NVRAM	Overwrite media by using agency approved and validated overwriting technologies/methods/tools. Each overwrite must reside in memory for a period longer than the data resided. Remove all power to include battery power.
RAM	Purge functioning DRAM by powering off and removing the battery (if battery backed).

Also in 2006 Wright et al. argued that a single overwrite pass was sufficient to sanitize modern hard drives, even drives that did not provide an ATA-SE function (Wright, Kleiman, and Sundhar 2008). They disputed the claim that data can be recovered from a sanitized hard disk drive using Magnetic force microscopy or other methods. They performed a series of experiments using several different types of hard drives and several different overwriting methods. To sanitize or wipe a hard drive they used several different data patterns including all 0's, all 1's, a 01010101, a 00110011 and a 0000111 pattern. The results of their experiment using a new hard drive with one overwrite and a new hard drive with three overwrites are shown in Table 2-3.

Table 2-3: Wiping Experiment Results (Wright, Kleiman, and Sundhar 2008)

Probability of Recovery	**New Drive with one wipe**	**New Drive with three wipes**
1 bit	0.87	0.64
2 bits	0.7569	0.4096
4 bits	0.57289761	0.16777216
8 bits	0.328211672	0.028147498
16 bits	0.107722901	0.000792282
32 bits	0.011604223	6.2771E-07
64 bits	0.000134658	3.9402E-13
128 bits	1.81328E-08	1.55252E-25
256 bits	3.28798E-16	2.41031E-50
512 bits	.08108E-31	5.8096E-100
1024 bits	1.16873E-62	3.3752E-199

Based upon their experiment results they concluded that "This study has demonstrated that correctly wiped data cannot reasonably be retrieved even if it is of a small size or found only over small parts of the hard drive. Not even with the use of a MFM or other known methods" (Wright, Kleiman, and Sundhar 2008).

In 2007 the National Industrial Security Program Operating Manual, also known as the DOD 5220.22.M standard, was updated to provide additional recommendation for the sanitization of magnetic media. These recommended sanitization methods are documented in a Clearing and Sanitization Matrix published by the Defense Security Service (D.S.S 2007). Table 2-4 provides a summary of the 2007 sanitization recommendations from the DSS Clearing and Sanitization Matrix. The approved sanitization methods include:

A. Degauss with Type I, II, or III degausser.
B. Degauss with same Type (I, II, or III) degausser.
C. Overwrite all addressable locations with a single character.
D. Overwrite all addressable locations with a character, its complement, then a random character.
E. Each overwrite must reside in memory for a period longer than the classified data resided.
F. Overwrite all locations with a random pattern, then with binary zeroes, and finally with binary ones.
G. Remove all power to include battery power.
H. Perform a full chip erase as per manufacturer's data sheets.
I. Perform h above, then c above, a total of three times.

J. Perform an ultraviolet erase according to manufacturer's recommendation.

K. Perform j above, but increase time by a factor of three.

L. Destruction

M. Destruction required only if classified information is contained.

Table 2-4: Defense Security Service Sanitization Matrix (D.S.S 2007)

Media Type	**Approved Sanitization Methods**
Magnetic Tape (Type I, II, III)	B, L
Magnetic Disk, non-removable, rigid	A, L
Optical Disk, Write-Once, Read Many (WORM)	L
Optical Disk, Read-Many, Write Many	L, M
Dynamic Random Access Memory	C, G, L
Electronically Erasable PROM (EEPROM)	F, L
Erasable Programmable ROM (EPROM)	C, K, L
Flash EPROM (FEPROM)	C, H, L
Programmable ROM (PROM)	L
Magnetic Core Memory	A, D, L
Non-Volatile RAM (NVRAM)	C, L
Read Only Memory (ROM)	L
Synchronous DRAM (SDRAM)	C, G, L
Static Random Access Memory (SRAM)	C, G, L

A significant change implemented by DSS in 2007 was that they would no longer approve overwriting procedures for the sanitization of hard drives used for classified processing.

In 2009 Seagate and several other hard drive vendors released Self-Encrypting Drives (SED) that automatically encrypts data using the Advanced Encryption Standard (Seagate 2009). SED drives can be completely sanitized by deleting the drive's encryption key. In 2010 several Seagate SED drives received NIST FIPS 140-2 certification (Seagate 2010). Erasing a hard drive that provides internal hardware encryption is often called Full Disk Encryption (FDE) Enhanced Secure Erase or Fast Secure Erase (Hughes and Coughlin 2006). As shown in Figure 2-1, the fast secure erase process can completely sanitize a hard drive by simply changing the encryption key.

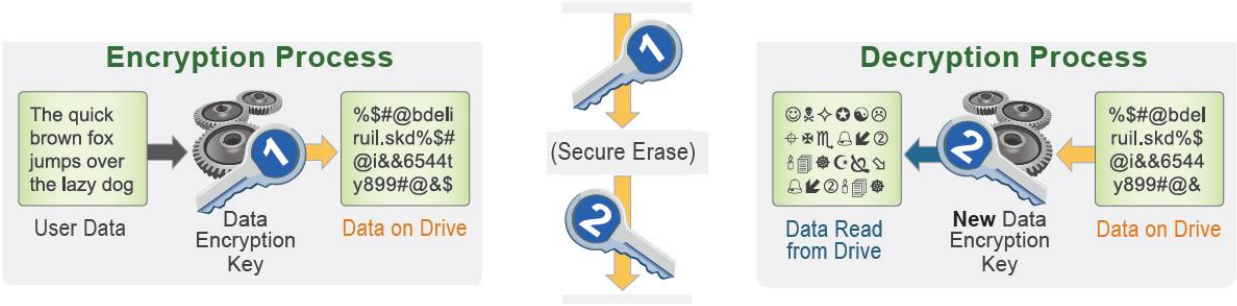


Figure 2-1: Fast Secure Erase Hard Disk Sanitization Process (Seagate 2009)

The most common hard disk drive sanitization methods in use today include the DOD 5220 Block Erase, the Secure Erase function based on the ATA-SE command, and the Fast Secure Erase method which deletes a hard drive's AES encryption key (Hughes and Coughlin 2006). A summary of key performance metrics for the most popular sanitization methods is presented in Table 2-5.

Table 2-5 - Hard Drive Sanitization Key Performance Metrics

Sanitization Method	Average Time (1 TByte Drive)	Security Posture	Comments
DOD 5220 Block Erase	Up to 2 days	Medium	Can't sanitize bad blocks
Secure Erase (ATA-SE)	Up to 5 hours	High	Requires built-in ATA-SE function
Fast Secure Erase (AES-128)	Up to 5 seconds	Very High	Requires SED drives. Up to SECRET level.
Fast Secure Erase (AES-256)	Up to 5 seconds	Very High	Requires SED drives. Up to TOP Secret level.

As shown in Table 2-5 the fastest and most secure hard disk sanitization method in use today is Fast Secure Erase using AES-256 encryption. Two issues associated with

this sanitization method are that it requires the use of SED capable drives, and a-priori government approval for use with classified data.

In the past decade some research has been published related to the sanitization of flash memory devices (Wei et al. 2011). The DOD 5220 Block Erase, Secure Erase, and Fast Secure Erase methods can also be used to sanitize flash memory. Wei et al. performed a series of sanitization tests on commercially available Solid State Drives (SSD) which were designed using flash memory. Some of these SSD drives came with ATA-SE functions, while others did not since they were built using USB technology. One of the SSD drives included built-in hardware AES encryption, and thus provided a Fast Secure Erase capability. Table 2-6 summarizes shows the results of their ATA-SE process testing on twelve different commercial SSD drives.

Table 2-6 –SSD ATA-SE Test Results (Wei et al. 2011)

SSD #	ATA-SE Test	Comments
A	Not Supported	
B	Failed	Drive reported success but all data remained on drive
C	Failed	Sanitization only successful under certain conditions
D	Failed	Sanitization only successful under certain conditions
E	Encrypted	Drive encrypted, unable to verify if keys were deleted
F	Passed	
G	Passed	
H	Passed	
I	Passed	
J	Not Supported	USB mass storage device does not support ATA security
K	Not Supported	USB mass storage device does not support ATA security
L	Not Supported	USB mass storage device does not support ATA security

Wei et al. discovered errors in the implementation of ATA-SE and Fast Secure Erase functions on some flash memory devices, and recommended that individual devices be tested to insure they were properly performing the sanitization process (Wei et al. 2011).

While some flash memory devices provide the equivalent of the ATA-SE or Self-Encrypting Drive functions, most current flash memory devices do not include these capabilities. The majority of flash memory devices on the market today must be sanitized using the DOD 5220 Block Erase method, or another equivalent overwriting technique.

2.2 Limitations of Device Sanitization Literature

The literature in the field of device sanitization discusses the sanitization of a single hard drive, non-volatile memory, or flash memory device. There is consensus within the current literature that a single overwrite pass using random data can be used to properly sanitize modern hard disk drives, or non-volatile memory devices. If a hard disk drive provides an ATA-SE function, then the device can also be properly sanitized by performing an ATA-SE Secure Erase operation. Finally if a hard disk drive provides built-in hardware encryption, then the device can be properly sanitized by deleting or replacing the drive's internal encryption key. These three sanitization techniques have been approved by NIST or other Government agencies for sanitization of hard disk drives or non-volatile memory that contains sensitive, but unclassified information.

The literature does not discuss how to sanitize a device that includes more than one type of non-volatile memory device, such as a hard drive along with firmware stored in NV-RAM. The literature does not discuss how to simultaneously sanitize many devices of the same type, how to simultaneously sanitize different types of devices, or how to automate the sanitization process so that it can be performed without human interaction. The literature does not discuss how to sanitize devices that have been used for classified processing in order to reuse those devices for processing at a lower classification level.

Finally, the literature does not discuss how device sanitization can be used to remove persistent malware.

2.3 Improving Device Sanitization

As discussed in the previous section there are a number of limitations to the current device sanitization processes, as documented in the literature. The remainder of this dissertation will present methods for improving device sanitization with a focus on addressing the following areas.

1. Simultaneous sanitization of all components within a single device type.
2. Simultaneous sanitization of multiple devices of the same type.
3. Simultaneous sanitization of different device types.
4. Automation of the sanitization process so that it does not require human intervention.
5. Sanitization of devices that have been used for classified processing so that they can be reused at a lower classification level.

Chapter 3 - Sanitization of Server, Switch and Routers Components

The first step in determining how to sanitize a server, switch, or router is to acquire a Statement of Volatility from the manufacturer. Each manufacturer uses their own format for a Statement of Volatility, but in general they contain a list of all non-volatile memory areas within the device, what those areas of memory are used for, and how the memory can be accessed (i.e. read-only, read-write, write-once, etc.). A Statement of Volatility is often published in a device's hardware reference manual or user's guide. In some cases a device manufacturer will only provide a Statement of Volatility to customers who specifically ask for one. Appendix A provides a sample Statement of Volatility from Cisco.

3.1 Sanitization of Volatile Memory

Servers, switches and routers include both volatile and non-volatile memory. Volatile memory used in these devices includes Dynamic Random Access Memory (DRAM), Static Random Access Memory (SRAM) and Double Data Rate Synchronous Dynamic Random Access Memory (DDR SDRAM). According to the National Security Agency (NSA), sanitization of volatile memory can be accomplished "by removing the power. Once power is removed, sanitization is instantaneous. Or, sanitize functioning DRAM and SRAM by overwriting all locations with a known unclassified pattern. Verify the overwrite procedure by randomly rereading the overwritten information to confirm that only the known pattern can be recovered. (NSA/CSS 2006)"

3.2 Sanitization of Non-Volatile Memory

The most common forms of non-volatile memory used in servers, switches and routers include Read-Only Memory (ROM), Programmable Read-Only Memory

(PROM), Erasable Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and flash memory. Read-Only Memory devices cannot be overwritten since they are read only. If a Read-Only Memory device contains sensitive, proprietary or classified information, or malware, the only way to sanitize it is through destruction. An approved method for sanitizing EPROM devices is "by performing an ultraviolet erase according to the manufacturer's recommendations, but increase the time requirement by a factor of three. Next, overwrite all bit locations with a known unclassified pattern" (NSA/CSS 2006). An approved method for sanitizing EEPROM devices is "overwriting all locations with a known unclassified pattern. Verify the overwrite procedure by randomly rereading the overwritten information to confirm that only the known pattern can be recovered." (NSA/CSS 2006).

Programmable ROM and other types of writable non-volatile memory devices will require sanitization since it is possible for a user, or a hacker, to change the device contents. Sanitizing non-volatile memory requires overwriting all areas of the device with known data, such as a fixed or random pattern, and then verifying that all the data has been properly overwritten. Most non-volatile memory devices do not include built-in sanitization features like ATA-SE Secure Erase or Fast Secure Erase using AES encryption. In most cases the most effective sanitization method for non-volatile memory is one based on the DOD 5220 Block Erase process.

3.3 Sanitization of Hard Disk Drives

There are three general approaches that can be used to sanitize server, switch, or router hard disk drives. These include sanitization at the physical level, at the file system level, or at the user level (Reardon, Basin and Capkun 2013). At the physical level a disk

controller is used to read/write data in fixed sized blocks. Sanitization at the physical level overwrites all of the data within a block, as well as all of the blocks on the disk.

The file deletion functions in most modern operating systems do not actually delete data from a hard drive; rather they just mark the data on the drive as unused. Several researchers have proposed extensions to operating system software to implement a secure file deletion function that properly sanitizes a file after it is deleted. For example, proposed extensions to Linux file systems can provide a secure file deletion function (Joukov and Zadok 2005). Researchers have also proposed using cryptography to implement a secure file deletion function (Peterson et al. 2005).

At the user level there are a number of commercially available tools that can be used to sanitize files, folders, metadata, or even an entire hard drive. These tools include CyberScrub, DataScrubber, DataGone, Eraser, SecureClean, Wipe, WipeDrive, WipeErasserXP and others (Garfinkel and Shelat 2003). The effectiveness of these tools depends in part upon the technical sophistication of the user and their knowledge of where sensitive data is stored on their hard drives. Some researchers question the effectiveness of user level sanitization tools due to concerns over their design, reliability, and ease-of-use (Joukov, Papaxenopoulos, and Zadok 2006).

For an automated system designed to sanitize a variety of hard drives, using the physical level sanitization approach will be better than using either the file system or user level approach. While the file system approach can be very effective, it has not been implemented for all operating systems, and thus could limit the types of devices that can be properly sanitized. The user level approach can also be effective, but requires an end user's knowledge about where sensitive data resides in order to perform the sanitization

process. Requiring end user knowledge will make automating the sanitization process difficult.

3.4 Sanitization of Flash Memory

Most servers, switches and routers contain one or more areas of flash memory. These can be in the form of an internal USB Drive, a removable SDRAM Card, or integrated flash memory circuits mounted on the motherboard or add-in cards. One of the most popular uses for flash memory is a Solid State Drive, which combines the high capacity of a traditional hard disk drive with the fast access times normally associated with random access memory. As discussed in the Related Work section, some flash memory devices can be sanitized using the ATA-SE Secure Erase, or Fast Secure Erase using AES encryption methods (Wei et al. 2011). However, only a few flash memory devices currently support the Secure Erase or Fast Secure Erase functions. Therefore for most common flash memory devices the only viable sanitization option is the DOD 5220 Block Erase method.

3.5 Slack Space

An important aspect of the sanitization process is to ensure that any slack space contained in a hard disk drive, NVRAM device, or flash memory device has been properly sanitized. Slack space in hard disk drives or flash memory includes file allocation tables, defective blocks, and unallocated blocks. In general commercially available hard disk drive sanitization software will properly sanitize this slack space.

Slack space in NVRAM devices include areas of memory that are not being used to store either firmware or configuration data. Firmware update programs from device

vendors may only overwrite those portions of NVRAM required to hold the new firmware, and may not overwrite unused portions of NVRAM. Properly sanitizing NVRAM may require working with a device or firmware vendor to make sure their update program also overwrites this slack space, and if not, having the vendor develop a custom version of the update program for use in the sanitization process.

Chapter 4 - Sanitizing a Single Server, Switch or Router

The process of sanitizing a single server, switch or router should be based on four fundamental assumptions. First we must assume that every byte of data stored on the device's hard disk drive, NVRAM, or flash memory can contain information that must be sanitized. Second we must assume that every hard disk drive, NVRAM, or flash memory can contain malware that could re-infect the device if it is not properly sanitized. Third, if the device is to be reused, we must insure that after the completion of the sanitization process the device is left in a trusted state. A trusted state means that the firmware and configuration data stored in every area of the device's non-volatile memory, and the data stored on the device's hard disk drives are identical to a predefined, and unconditionally trusted, set of firmware, configuration data, and operating system data. Finally any external devices used during the sanitization process must also be in an unconditionally trusted state.

4.1 Step 1: Establish a Root of Trust

Server, switch and router sanitization requires the use of a number of software programs to update the BIOS, perform firmware updates, and sanitize hard disk drive or flash memory devices. All of these software programs must be loaded from a source that can be trusted. Before the sanitization process can begin, a root of trust must be established, and all software used in the device sanitization process should be stored in this trusted location. Ideally the root of trust should be built upon read-only memory (e.g. a CD-ROM, DVD-ROM) and should be protected using cryptographic hash functions designed to detect unauthorized changes.

A root of trust must also be used to store device configuration data, such as the device's MAC address, IP address or host name which are typically stored in NVRAM. The process of sanitizing a device's NVRAM will erase this configuration data. In order to reuse the device the configuration data must be restored after the sanitization process has been completed.

Many servers include a Trusted Platform Module (TPM) designed to insure the integrity of the server's internal hardware. A TPM includes Platform Configuration Registers (PCR) used to store security related data, cryptographic keys used for authentication, and optionally cryptographic key management, and random number generation functions (Trusted Computing Group 2007). Figure 4-1 shows a simplified block diagram of a Trusted Platform Module.

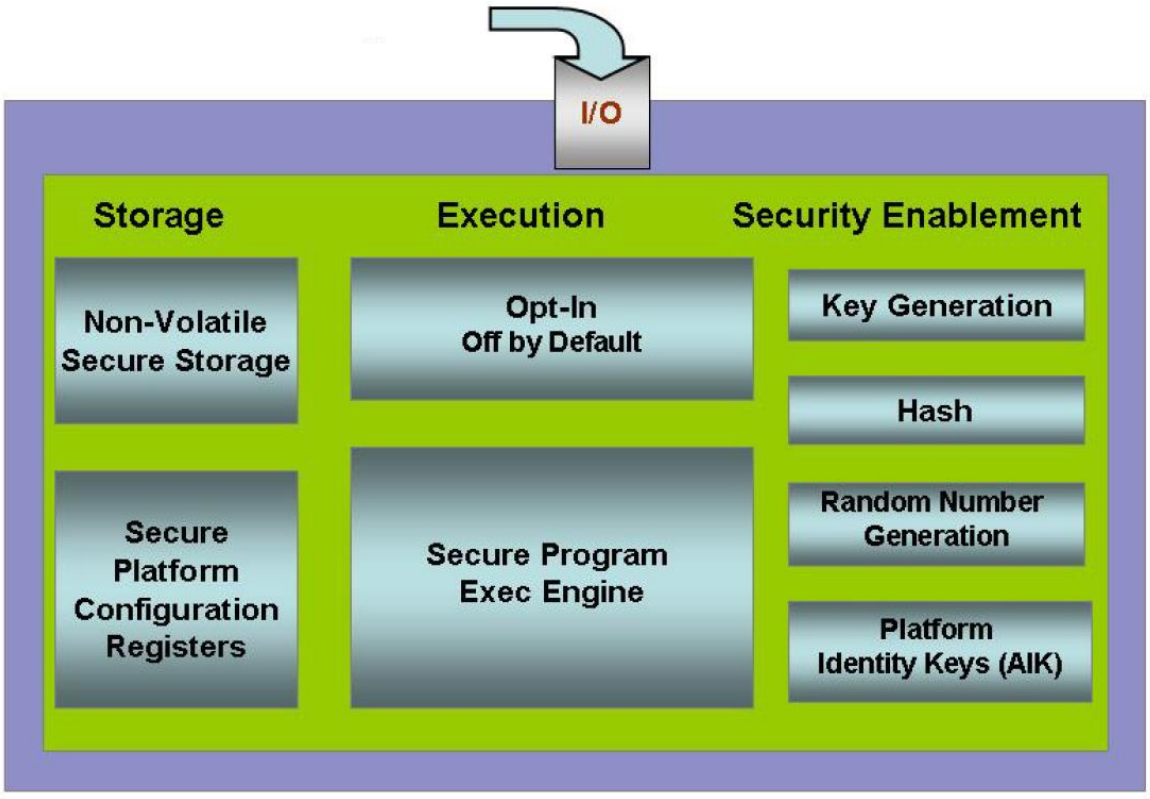


Figure 4-1: Trusted Platform Module Block Diagram (Trusted Computing Group 2007)

The server's BIOS, in combination with the Trusted Platform Module are used to form a root of trust that can be built upon as additional software, such as the server's operating system and application programs are loaded and run. The server's TPM/BIOS root of trust is based on the assumption that both the TPM and the BIOS can be completely trusted. Some researchers have questioned if a TPM can be used as a root of trust since it could be susceptible to cyber-attacks. Prior research has shown that TPM devices are potentially vulnerable to cuckoo attacks (Parno 2008), reset attacks, and RSA timing attacks (Kauer 2007).

As described above the sanitization process must rely on two distinct roots of trust. The first root of trust stores the software used for device sanitization. The second root of trust is located on the device itself and is formed through the combination of a trusted BIOS and optional TPM components. Since we will assume that the device's BIOS/TPM root of trust has been compromised, one of the first steps in the sanitization process must be to reestablish the device's BIOS/TPM root of trust by reloading the BIOS and any rewritable portions of the TPM with trusted versions of these critical firmware programs.

4.2 Step 2: Sanitize the Device Components in the Proper Order

Due to the internal complexity of servers, switches and routers, it is generally not possible to sanitize them in a single step. Many different sanitization steps must be performed in a specific order to prevent persistent malware from reinfecting the device. Understanding the boot order of the device will help to identify the various items of firmware and software that are executed by the system, and how they may depend upon each other. For example hard disk controller firmware will most likely be executed after

the system BIOS is running. In this case the hard disk controller firmware should be sanitized, and after that step has been completed the BIOS should be sanitized.

Some devices include a verbose boot mode setting in their BIOS configuration, which when selected can provide details on the various steps performed during the boot process. Additional information about a device's boot order can generally also be found in the manufacturer's hardware reference manual or user's guide. Depending upon the specific device, determining the proper sanitization order can be a relatively simple, or a very complex process. Carefully reviewing the device's Statement of Volatility, determining the boot order, and analyzing any dependencies between the device's components will help determine the proper sanitization order.

4.3 Step 3: Restore the Device to a Trusted State

After a server, switch or router has been completely sanitized it must then be restored to a trusted state. This process generally involves retrieving a trusted version of the device's BIOS, firmware and software from a root of trust, copying the software to the device's NVRAM, and then validating that the restored firmware and software are identical to the versions stored in the root of trust. Restoring the device's BIOS, firmware and software must also be done in the proper order. For example the system BIOS will need to be restored before the device's operating system software can be restored.

In addition to restoring the device's BIOS, firmware and software, the device's configuration data must also be restored. Configuration data can include the device's network MAC address, host name, or other parameters that are required for the device to operate properly. Configuration data must be retrieved from a trusted source, written into

the proper locations within a device's NVRAM, and validated to insure that it matches the trusted configuration data.

4.4 Step 4: Validate the Sanitization Results

The next step in the server, switch or router sanitization process is to validate that each of the device's hard disk drives, NVRAM devices, and flash memory devices have been completely sanitized, and updated with trusted firmware and configuration data. Ideally each of the programs used during the sanitization process will include a validation step, and will provide log files or some other form of documentation that can be used to independently validate that the sanitization process was completed properly. If the individual sanitization programs do not perform validation steps, or do not provide appropriate log files, then a separate validation and logging program must be developed to perform these functions. Appendix F of NIST 800-88 provides examples of sanitization validation information that should be documented and retained after a device has been sanitized (Kissel et al. 2006).

4.5 Step 5: Sanitize any Equipment Used During the Sanitization Process

The final step of the server, switch or router sanitization process is to sanitize any equipment used during the device sanitization process. For example if the device being sanitized is connected to a network, and the sanitization process used the network to transmit or receive data, or to execute commands during the sanitization process, then any devices that were connected to the network during the sanitization process should also be sanitized. This step will remove any persistent malware that may have migrated during the sanitization process. It is a good idea to move a device to a private network

before starting the sanitization process. This will minimize the number of additional network devices that also have to be sanitized.

Chapter 5 - An Automated Sanitization System Architecture

Figure 5-1 shows an architectural block diagram of an Automated Sanitization System. This architecture was developed by the author during Phase II of the DARPA National Cyber Range program.

The Automated Sanitization System architecture includes a Sanitization Control Computer paired with a Trusted Database. Three different types of switches are used to provide serial and network communications paths between the Sanitization Control Computer and the devices to be sanitized. These communication paths allow the Sanitization Control Computer to provide custom boot images via a network connection, and to execute low level device operating system commands via a serial console port connection.

The network switch is used to form a private TCP/IP network between the Sanitization Control Computer and the devices to be sanitized.

The Console Switch is used to provide dedicated serial connections between the Sanitization Control Computer and the serial console ports on each device to be sanitized.

The Remotely Controlled Physical Layer Switch is used to automatically make or break network and serial console connections between the Sanitization Control Computer and the devices to be sanitized. This switch is critical to automating the sanitization process by allowing devices to be connected without human involvement (e.g. connecting cables to devices).

AC power to each device to be sanitized can be remotely controlled from the Sanitization Control Computer using a standard network controlled Power Distribution Unit (PDU).

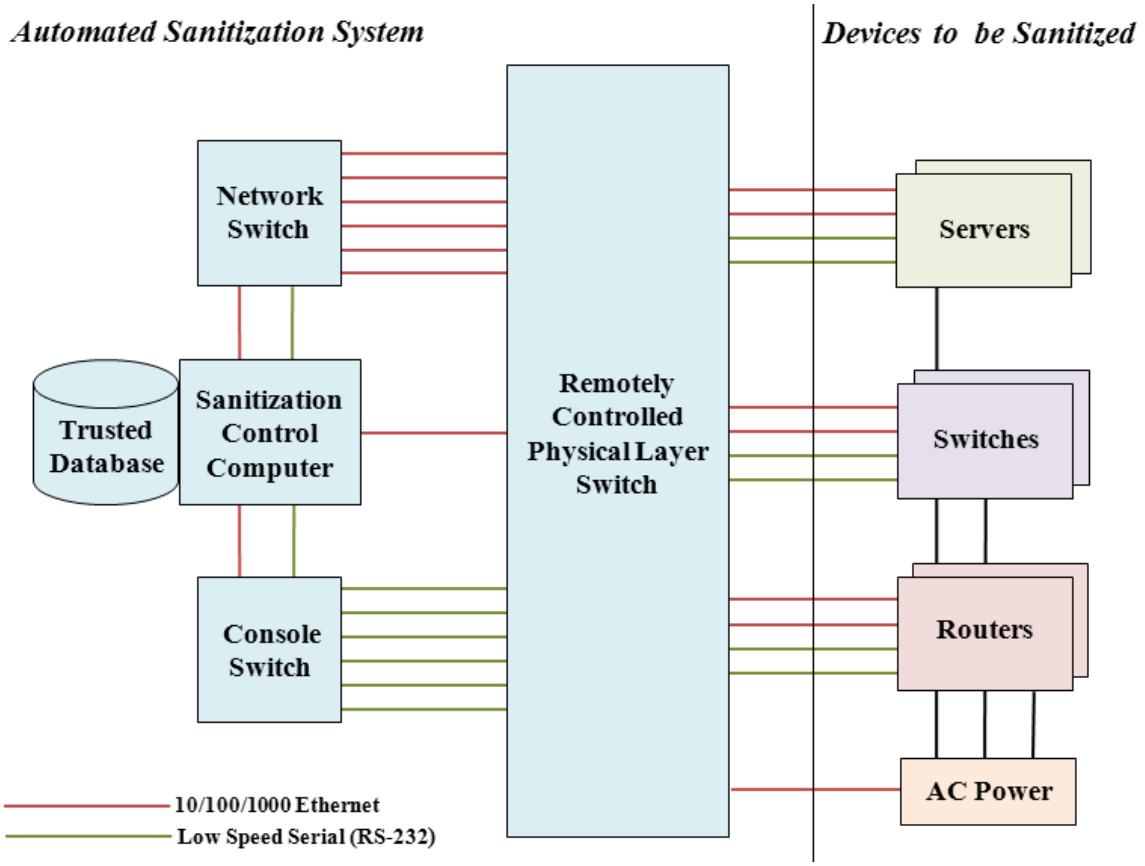


Figure 5-1: Automated Sanitization System Architecture

The Trusted Database stores all of the programs required to sanitize any device that can be connected to the Sanitization Control Computer. The Trusted Database also stores all configuration data required to restore a device to a trusted state after the sanitization process has been completed. Two important characteristics of the Trusted Database are that the programs and configuration data are stored in read-only memory, and cryptographic hash functions are used detect unauthorized changes to the programs or

configuration data. The Trusted Database is also used to store log and validation data generated during the sanitization process.

Servers to be sanitized will boot from the Sanitization Control Computer via a network connection. The sanitization boot image includes a small operating system that can be executed from the server's internal RAM. The RAM based operating system will retrieve and execute programs from the Trusted Database to sanitize individual server components in the proper order. Log and validation data collected during the sanitization process will be returned to the Sanitization Control Computer to be stored in the Trusted Database.

Switches and routers to be sanitized will be forced to boot from a local read-only bootstrap program by the Sanitization Control Computer. This forced boot is accomplished through the combination of remotely power cycling the device followed by the transmission of special control characters to the device's serial console port. Once the switch or router has been booted into its trusted bootstrap program, the Sanitization Control Computer can overwrite the internal device memory, load trusted versions of the device's operating system and firmware, and restore the device's configuration data using a combination of the serial console interface and the private TCP/IP network connection.

One important characteristic of the Sanitization Control Computer is that it must be self-sanitizing. This means that it must boot from trusted, local, read-only memory so that it can be completely sanitized by a power cycle. If possible the network switch, console switch and physical layer switch connected to the Sanitization Control Computer must also be self-sanitizing via a power cycle. If this is not possible, then the

Sanitization Control Computer must be able to remotely sanitize these switches after the device sanitization process has been completed.

Chapter 6 - An Implementation of the Automated Sanitization System Architecture

The Cyber Measurement & Analysis Center (CMAC) was a JHU/APL research and development program focusing on new and innovative technologies that can support rapid, repeatable and secure cyber experimentation. CMAC was funded by DARPA under Phases I and II of the National Cyber Range (NCR) program. During the CMAC program JHU/APL designed, developed, assembled and tested a prototype National Cyber Range system that included a large number of servers, switches, routers, packet capture devices, keyboard/video/mouse (KVM) switches, power distribution units (PDU), and serial console management (SCM) switches. These devices could be remotely configured and interconnected to form complex networks for the purpose of cyber experimentation. After the completion of an experiment each device allocated to the experiment test bed had to be sanitized to remove any sensitive/classified data or persistent malware, and to prepare it for reuse.

The DARPA levied requirements for the CMAC sanitization system included (1) that all persistent storage areas within a device are completely sanitized and the device is restored to a trusted factory default configuration; (2) that the sanitization process can be run automatically and completed in as short a period of time as possible; and (3) that a device which has been used to process classified information up to the Top Secret - Sensitive Compartmented Information (TS//SCI) level can be reused at the Unclassified level after sanitization.

6.1 CMAC Sanitization System Design

The CMAC Sanitization System (Figure 6-1) includes a Sanitization Controller, a Cisco 3750G Network Switch, and a Raritan DSX32A Serial Console Manager.

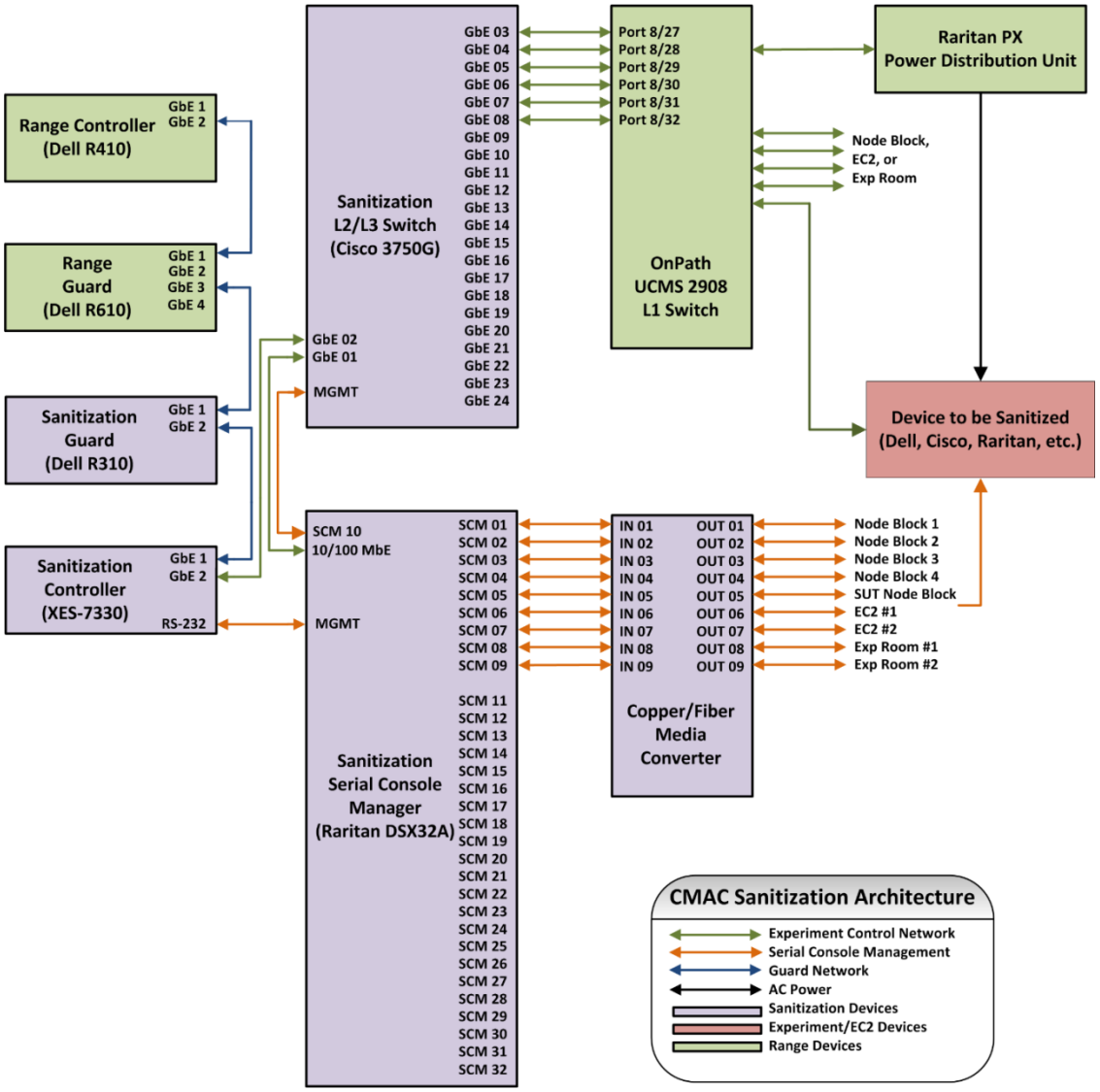


Figure 6-1: CMAC Sanitization System

These three devices are connected to the rest of the CMAC National Cyber Range system through a cross domain guard, a Layer 1 switch and a number of copper/fiber media converters. The cross domain guard, called the Sanitization Guard, facilitates command, control and status communications between the Range Controller and the

Sanitization Controller, and ensures that malware can't be transmitted from the sanitization system back to other range devices. The CMAC prototype system includes a 256 port OnPath UCMS 2908 Layer 1 Switch which interconnects devices to form unique experiment architectures, or in the case of sanitization, to form a private network of devices to be sanitized. Each equipment rack includes a Raritan PX Power Distribution Unit (PDU) that provides remote control of the power to any device in the CMAC system.

When one or more CMAC devices require sanitization, the Range Controller connects them to the Sanitization System using dedicated layer 1 switch ports, and then issues a command to the Sanitization Controller via the Sanitization Guard to begin the sanitization process. After completion of the sanitization process the Sanitization Controller sends status and logging information to the Range Controller via the Sanitization Guard. If the sanitization process was completed without errors, the Range Controller will disconnect the devices from the dedicated layer 1 switch ports, and reallocate those devices to new cyber experiments as required.

6.2 CMAC Sanitization Controller

The Sanitization Controller is the heart of the CMAC Sanitization System. It serves as the Root of Trust for the sanitization process, stores all of the sanitization programs for the devices to be sanitized, and includes automation software designed to complete the sanitization process without human guidance or interaction. The Sanitization Controller is built using an Extreme Engineering Solutions XES-7330 Single Board Computer running a standard VxWorks operating system. A key feature of the XES-7330 is that all of its internal firmware, flash memory, and even the boot drive can be write protected using a

hardware jumper. The XES-7330 includes an Intel® Core™i7 processor, and 4 GBytes of SDRAM (volatile memory). Two key benefits of the hardware write protection feature are that the XES-7330 meets the requirements for a Root of Trust (i.e. read-only) and it can be sanitized through a simple power-cycling process. All of the Sanitization Controller's software, including the operating system, the sanitization software, and the device firmware update programs are stored in write-protected memory areas of the XES-7330 Single Board Computer. A block diagram of the XES-7330 Single Board Computer is shown in Figure 6-2.

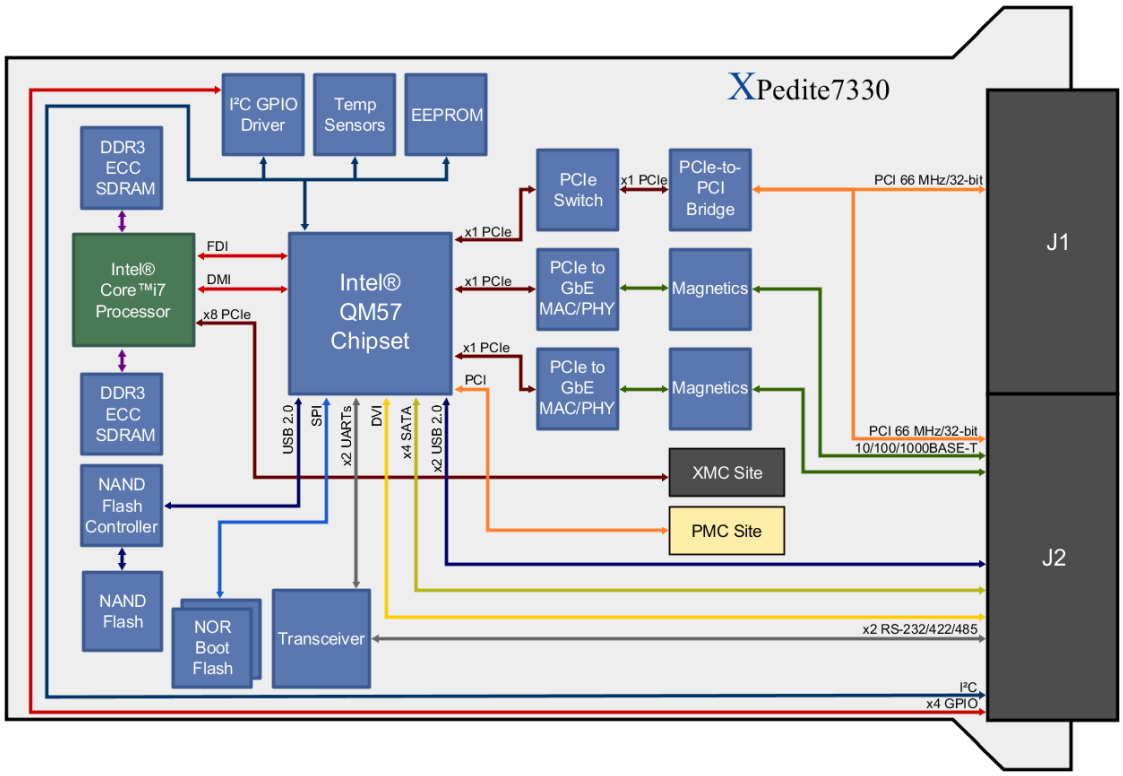


Figure 6-2: XES 7330 Single Board Computer Block Diagram (Extreme Engineering Solutions 2013)

In addition to running the automation software that controls the entire sanitization process, the Sanitization Controller is also configured as a Dynamic Host Configuration

Protocol (DHCP) server, a Preboot Execution Environment (PXE) server, a Network File System (NFS) server, and a Trivial File Transfer Protocol (TFTP) server.

Intel's Preboot Execution Environment (PXE) (Intel 1999) defines a common set of pre-boot services, and specifies how standard Internet protocols such as TCP/IP, DHCP and TFTP can be used to remotely boot a computer from a network server. PXE booting allows each server on the sanitization network to be remotely booted with a trusted sanitization Network Boot Program (NBP). The Sanitization Controller's PXE Boot server automatically provides each server to be sanitized with a trusted boot image via their network connection. Figure 6-3 provides a diagram of the PXE Boot process.

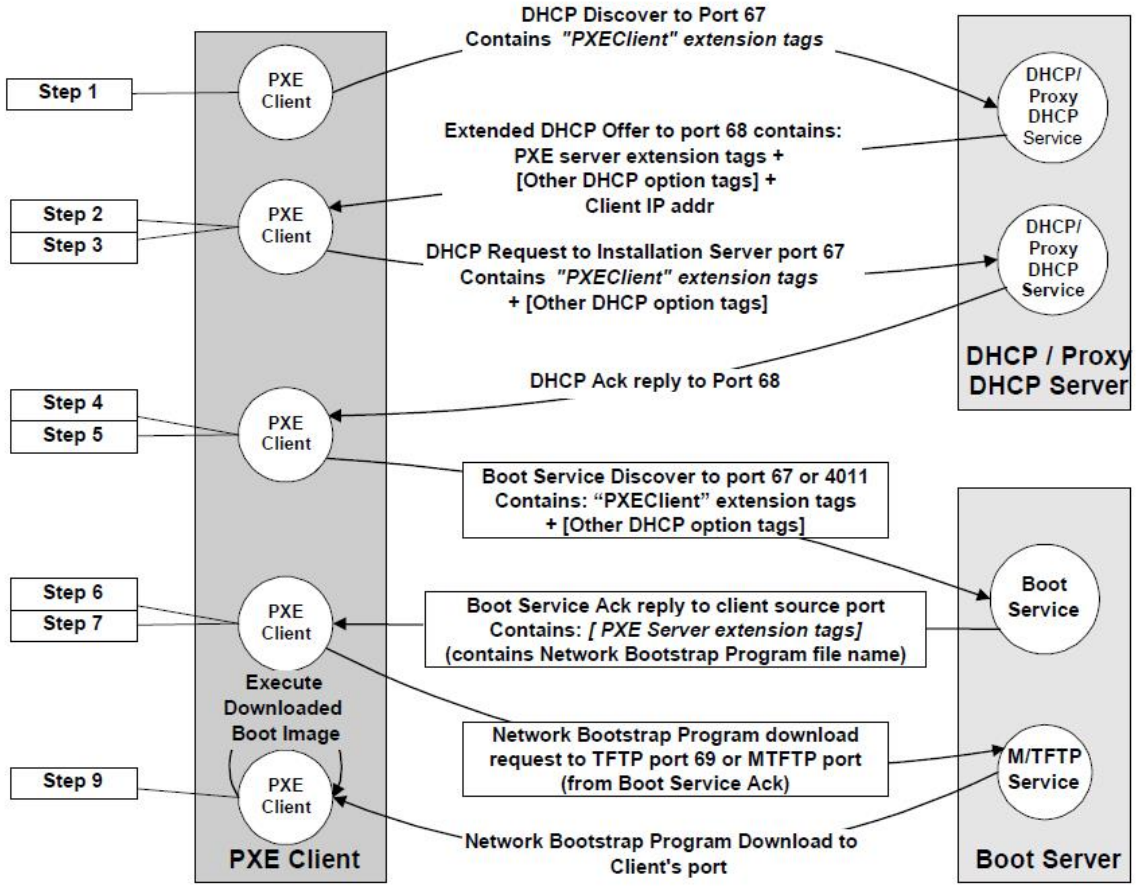


Figure 6-3: PXE Boot Process (Intel 1999)

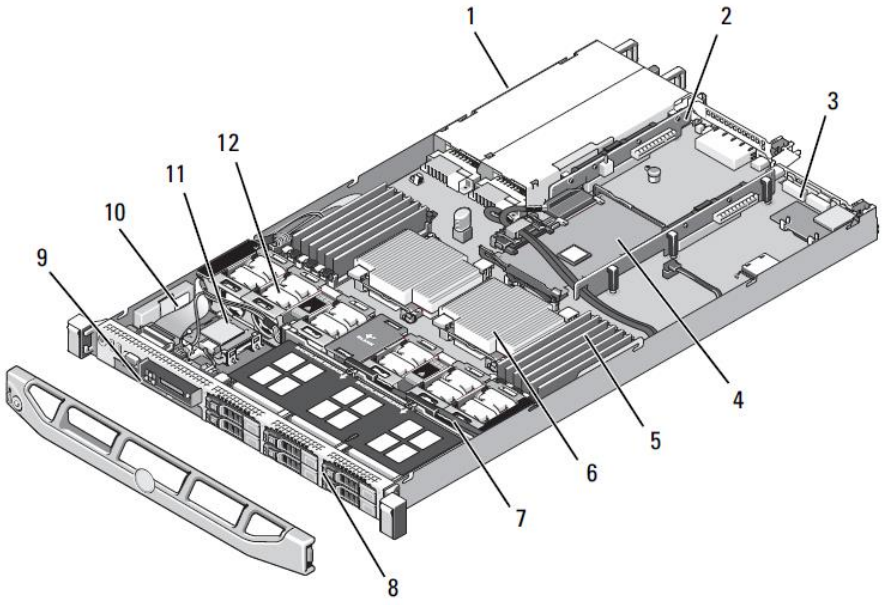
The Dynamic Host Configuration Protocol (DHCP) (Alexander and Droms 1997) is an industry standard method used to provide configuration and control information to devices on an IP network. DHCP uses a client-server model where clients request network configuration information, such as an IP address, from a DHCP server on the network. The Sanitization Controller's DHCP server automatically assigns unique IP addresses to each device on the sanitization network.

The Network File System (NFS) was developed to allow devices to access files over a network connection (Sandberg et al. 1984). NFS was subsequently documented in a series of Internet RFCs and has become an industry standard method of sharing files over a network. The Sanitization Controller's NFS server is used to store trusted copies of the sanitization software for each device type and trusted copies of the unique configuration data required for each device. The NFS server is also used to temporarily store log files created during the sanitization process.

The Trivial File Transfer Protocol (TFTP) is used to transfer trusted versions of a device's BIOS, firmware, software and configuration data from the Sanitization Controller's root of trust. This data transfer method is primarily used during the sanitization of network switches and routers whose ROM based boot loader program provides built-in TFTP client software. TFTP is built on top of the User Datagram Protocol (UDP) and was designed to facilitate the transfer of files between network hosts (Sollins 1992).

The CMAC prototype system includes over 70 Dell PowerEdge R310, R410 and R610 servers. The Dell R310 server is a 1U rack mounted computer with a single Intel

Xeon quad-core processor, 4 GBytes of RAM, a local 1 TByte SATA SED hard drive, and two Gigabit Ethernet ports. The Dell R410 server is a 1U rack mounted computer with a single Intel Xeon quad-core processor, 8 GBytes of RAM, a local 1 TByte SATA SED hard drive, and two Gigabit Ethernet ports. The Dell R610 server is a 1U rack mounted computer with dual Intel Xeon hex-core processors, 32 GBytes of RAM, a local 1 TByte SATA SED hard drive, and four Gigabit Ethernet ports. Figure 6-4 shows an internal view of a Dell R610 server.



1	power supply bays (2)	2	expansion-card riser (2)
3	iDRAC6 Enterprise card	4	integrated storage controller card
5	memory modules (12)	6	heat sink/processor (2)
7	SAS backplane	8	hard drives (6)
9	optical drive	10	RAID battery (PERC only)
11	Internal SD Module	12	fans (5 or 6)

Figure 6-4: - Dell R610 Server Internal View (Dell 2013)

JHU/APL worked closely with Dell to develop a complete list of all non-volatile memory areas in each server. Table 6-1 provides a summary of all the non-volatile

memory areas in a Dell R610 server. Table 6-1 also shows which non-volatile memory areas have configuration data that must be restored after sanitization, and the names of the programs used to restore each area of firmware.

Table 6-1: Dell R610 Non-Volatile Memory Areas

#	*NVRAM Description*	*Configuration Data File(s)*	*Firmware Update Program*
1	System BIOS SPI Flash		PER610_BIOS_LX_2.3.12.BIN
2	LAN on Motherboard (LOM)	ETHDUMP ETHMAC	NETW_FRMW_LX_R294342.BIN
3	System Specific Information	sysinfo	
4	iDRAC Firmware		IDRAC6_FRMW_LX_R279926.BIN
5	iDRAC LC Data	MAS001.img MAS002.img	IDRAC6_FRMW_LX_R279926.BIN
6	iDRAC Logs		IDRAC6_FRMW_LX_R279926.BIN
7	System FRU (CPU)	FRUID0 FRUID2 FRUID3 FRUID4 FRUID5 FRUIDindex.txt	
8	CMOS	CMOSCONFIG	
9	PERC Controller Configuration		RAID_FRMW_LX_R287842.BIN
10	PERC Firmware		RAID_FRMW_LX_R287842.BIN
11	PERC Disk Configuration Data		RAID_FRMW_LX_R287842.BIN
12	Serial Attached SCSI Backplane		SASBP_FRMW_LX_R203413.BIN
13	Seagate SED HDD Firmware	MegaSAS.log	FRMW_LX_R283319.BIN
14	Power Supply Unit Firmware		PER610_PSU_FRMW_LX_R286745.BIN

Table 6-2 provides a summary of the shell scripts and programs used to sanitize a Dell R610 server. The shell scripts and two custom PXE boot images were developed specifically for the CMAC Sanitization System.

Table 6-2: Dell R610 Sanitization Scripts & Programs

Shell Script or Program Name	***Shell Script / Sanitization Program Description***	***Custom or Off-the-shelf?***
Backupconfig.txt	Text file which controls the server backup process	Custom
BackupLC.sh	Script to backup iDRAC Lifecycle Controller Data	Custom
config	Text file which controls the sanitization mode	Custom
CustomBIOS_Upd.sh	Script to install custom BIOS for sanitization	Custom
default	PXE Network Boot Program Configuration Data	Custom
DUPRestore.sh	Script to perform restore operations	Custom
FRUBackupTest.sh	Script to back up the FRU (CPU) Configuration Data	Custom
FRURestoreTest.sh	Script to restore the FRU (CPU) Configuration Data	Custom
FRUVerify.sh	Script to verify the FRU (CPU) Sanitization	Custom
iDRACUpd.sh	Script to install custom iDRAC firmware for sanitization	Custom
ipmitool	Program used to perform low-level server IPMI functions	Off-the-shelf
LCWipe.sh	Script to sanitize the iDRAC Lifecycle Controller	Custom
Logging_script.sh	Script to update sanitization log file	Custom
LOMBackup.sh	Script to back up the LOM Configuration Data	Custom
LOMVerify.sh	Script to verify the LOM Sanitization Process	Custom
MegaCLI	Program to initialize virtual disk on RAID	Off-the-shelf
PSUUpdate.sh	Script to sanitize the Power Supply Unit	Custom
RecreateLC.sh	Script to restore iDRAC Lifecycle Controller Data	Custom
SA.1	PXE Network Boot Program	Custom
SA.2	PXE Network Boot Program	Custom
sasdupie	Program to sanitize the SAS Backplane	Off-the-shelf
Script1.sh	Main sanitization script – calls all other scripts	Custom
Scrubconfig.txt	Text file which controls the server sanitization process	Custom
Storagescript.sh	Script to sanitize the Seagate SED drives	Custom
Verify.sh	Script to perform sanitization verification	Custom

Several off-the-shelf programs are called by the shell scripts to perform low-level functions such as partitioning a hard drive (MegaCLI), or setting server configuration data (ipmitool). A number of vulnerabilities have recently been disclosed related to the use of the Intelligent Platform Management Interface (IPMI) (Bonkoski et al. 2013). The IPMI functions on Dell R310, R410, and R610 servers are provided by the iDRAC Lifecycle Controller, which is completely sanitized by the CMAC Sanitization System. The CMAC Server Sanitization software was designed so off-the-shelf firmware update programs can be replaced with new versions when they become available. The sanitization software was also designed to provide detailed log information so that each step of the sanitization process is properly documented. Appendix A provides a state diagram and flow charts for the Dell Server Sanitization software.

6.2.1 Dell Server BIOS Sanitization

Sanitization of a Dell server's BIOS involves updating the BIOS with a new version from a DOS environment. The DOS BIOS tool sanitizes the entire BIOS Flash (NVRAM) chip and updates the BIOS firmware. The system is rebooted to DOS via custom iDRAC firmware. The DOS image is programmed on the iDRAC internal FLASH and will be used to boot from it for updating the BIOS.

6.2.2 Dell Server iDRAC Sanitization

The iDRAC sanitization process updates the Dell server's iDRAC with custom firmware designed specifically for the CMAC Sanitization System. The custom iDRAC firmware is based on the standard Dell iDRAC firmware with a few additional features.

The custom firmware plays a major role in automating the entire server sanitization process. The iDRAC sanitization process includes the following steps.

1. Update the server iDRAC with the custom sanitization firmware
2. Authenticate the custom iDRAC firmware
3. Initiate the server sanitization process
    a. The iDRAC firmware sanitizes the iDRAC Lifecycle Controller FLASH
    b. Sanitize the LC logs, system and iDRAC FRUs.
    c. Backup the system and iDRAC FRU data.
    d. Track the sanitization progress, detect errors, and report progress.
4. After the iDRAC sanitization process has been completed the custom iDRAC firmware will be used to restore the system and iDRAC FRU data.

6.2.3 Dell Server iDRAC Authentication

Since the iDRAC firmware can't be trusted prior to the start of the sanitization process, there needs to be a mechanism to authenticate the custom iDRAC firmware after it has been loaded into memory. The custom firmware uses an authentication method that makes use of public and private keys. The public key is embedded in the PXE boot image that is stored on a read-only root of trust, and the private key is embedded in the custom iDRAC firmware image.

6.2.4 Dell Server LOM Sanitization

The Dell server sanitization process uses embedded LOM tools to sanitize the LOM and restore it to a factory default state. The sanitization operation will need access to the LOM data that was backed up as part of the server sanitization backup process. When the LOM data is restored, the entire NVRAM area of the LOM is re written with the backup data, including all the LOM configurations, and MAC addresses.

6.2.5 Dell Server RAID Sanitization

The Dell server sanitization process uses embedded RAID tools to sanitize the RAID configuration, erase the NVRAMs on RAID, and then restore factory default setting to the RAID controller. The RAID sanitization process includes the following steps.

1. Erase the RAID configuration data from the PERC RAID Controller
2. Update the SAS backplane firmware
3. Update the RAID PERC Controller firmware
4. Update the hard disk drive firmware
5. Create a new virtual disk on the hard drive

6.2.6 Dell Server TPM Sanitization

The Dell server TPM NVRAM data will be sanitized using a tool embedded in the custom sanitization firmware. This process will clear the TPM of any configuration data stored previously if the server was setup for trusted platform. The sanitization tool is designed to run on the server without requiring any root of trust to be established. Prior to sanitization the platform is not trusted and once the sanitization process has been completed, the TPM can be used as a root of trust to build a trusted platform.

6.3 Dell Server Sanitization Modes of Operation

The CMAC Dell server sanitization software includes three unique modes of operation. A Backup Mode retrieves critical server configuration information including the Dell Asset Tag Number, Dell Service Tag Number, Field Replaceable Unit (FRU) data, and LAN on Motherboard data (e.g. MAC addresses). The Backup Mode is only run the first time a server is connected to the Sanitization Controller. The backup configuration data is stored in the Sanitization Controller's non-volatile flash memory so

it can be used to restore each server's unique configuration data during the Sanitization Mode. The Backup Mode is critical to defining a trusted state for each server to be sanitized by storing server specific configuration information that must be restored after sanitization. It is important to note that the Backup Mode only retrieves configuration data from a server; it does not make backup copies of the actual server firmware. Appendix B provides a sample Backup Log from a Dell R610 server sanitization backup operation.

A Sanitization Mode is used to erase all non-volatile memory devices within a server, write a trusted version of each firmware program into the appropriate area of NVRAM, restore the configuration data, and then verify that the server is in a trusted state. The trusted copies of server firmware used during the Sanitization Mode were generated by Dell, include cryptographic hashes to verify their integrity, and are stored and retrieved from the Sanitization Controller's Trusted Database in write-protected memory.

A Verification Mode compares the contents of a server's non-volatile memory areas with their trusted state values to ensure that they have been properly sanitized. Detailed log information is kept for each mode of operation, and for each server that goes through any of the sanitization modes. The Dell Service Tag number is used to uniquely identify servers in each of the sanitization log entries.

The design of the CMAC sanitization system and the server sanitization software supports the sanitization of more than one server at a time. During a typical sanitization operation fourteen different Dell R310, R410 and R610 servers are sanitized simultaneously.

6.4 Dell Server Hard Drive Sanitization

The hard disk drives used in the CMAC Dell R310, R410 and R610 servers are second generation Seagate Self-Encrypting Drives (SED). These SED drives use AES 256-bit encryption, which is considered strong enough to protect classified data up to the TS//SCI level (Hathaway 2003). JHU/APL worked closely with the National Security Agency's Information Assurance Directorate to ensure that the use of second generation Seagate SED drives, in combination with other CMAC security policies and procedures, would support using an SED drive for TS//SCI operations, and then reusing the same drive at the Unclassified level after sanitization.

Seagate SED drives provide two standard modes of operation, secure-erase and auto-locking. In the secure-erase mode the encryption key is only stored on the local hard drive. Sanitization of an SED drive operating in the secure-erase mode of operation requires the execution of a secure-erase command, which can be performed manually through the hard disk controller's user interface, or remotely using the hard disk controller's application programming interface (API). In secure-erase mode the user does not need an authentication key or password to decrypt and read data from the hard drive. Thus in the secure-erase mode the encryption and decryption processes are completely transparent to the end user.

A Seagate SED drive operating in the auto-lock mode requires an authentication key from an external source in order to read/write data from the hard drive. Typically an external key management service is used to generate and manage the SED drive encryption keys. Unlike the secure-erase mode of operation, there is no requirement to

execute a secure-erase command to sanitize an SED drive in auto-lock mode since the encryption key is not stored on the local hard drive.

The CMAC prototype system's SED hard drives are all operated in the secure-erase mode. Thus all CMAC hard drives are sanitized using the drive's built in secure-erase command. A secure-erase command requires less than five seconds for complete sanitization of an SED drive.

6.5 Dell Server Sanitization Mode of Operation Details

The Dell Server Sanitization Mode of operation includes three distinct phases. During Phase 1 each area of non-volatile memory within a Dell server is erased, and reinitialized to a trusted state. During Phase 2 the BIOS and iDRAC areas of non-volatile memory are again erased and reinitialized to a trusted state. During Phase 3, the CMOS area of non-volatile memory is erased and reinitialized, and a virtual drive is built and initialized on the internal SED drive. The specific steps within each phase and the order of those steps are designed to insure that any persistent malware stored in one area of non-volatile memory can't reinfect other areas of non-volatile memory. At the end of each phase the server is rebooted to insure that the next phase starts from a trusted state. The three Dell server sanitization phases and the steps within each phase include:

1) Phase 1
    a) SAS Backplane Sanitization
    b) PERC Configuration Sanitization
    c) SED HDD Security Key Deletion
    d) PERC Firmware Sanitization
    e) PERC Configuration
    f) PERC RAID Array Creation

g) LAN on Motherboard Sanitization
h) LAN on Motherboard Configuration
i) FRU Sanitization
j) FRU Configuration
k) iDRAC Sanitization
l) iDRAC Configuration
m) BIOS Sanitization
n) PSU Sanitization

2) Phase 2
   a) BIOS Sanitization
   b) iDRAC Sanitization
   c) iDRAC Configuration

3) Phase 3
   a) CMOS Sanitization & Configuration
   b) PERC RAID Initialization
   c) SED HDD Security Key Generation

The BIOS and iDRAC components are sanitized during both Phase 1 and Phase 2, due to their critical importance to the sanitization of all the other server components, and because each server maintains an internal backup copy of these images. The BIOS and iDRAC software must be fully functional to sanitize the SAS, PERC, LAN on Motherboard, PSU, and SED components. Once these components have been sanitized, the BIOS and iDRAC components are immediately resanitized to ensure that no persistent malware can reinfect the BIOS or iDRAC software.

The SED hard disk drives in each server are sanitized during Phase 1. This includes securely erasing all user data areas, sanitizing the SED drive firmware, and creating a new virtual drive using the PERC H-700 RAID controller. During Phase 3 the virtual

drive created in Phase 1 is initialized and a new security key is generated for the SED drive completing the hard disk drive sanitization process.

Appendix D provides a sample log from a Dell R610 sanitization operation.

6.6 Cisco Switch & Router Sanitization

During CMAC cyber-experiments Cisco 3750G layer 2/3 switches, Cisco 2851 routers and Cisco 3845 routers are used to form experiment and control networks. After an experiment has been completed these network switches and routers must be completely sanitized. A Cisco 3750G layer 2/3 switch is also used during the sanitization process to form the sanitization network. Thus the Sanitization Controller must be able to sanitize Cisco 3750G switches that have been allocated to cyber-experiments as well as the Cisco 3750G switch that is part of the CMAC sanitization system.

During the CMAC program JHU/APL worked closely with Cisco to develop sanitization procedures that would restore the 3750G switch, 2851 router and 3845 router to their original factory default configuration. This sanitization procedure clears and reformats the device's internal flash memory (including all slack space) and then installs a trusted version of the device's operating system software. It should be noted that the Cisco 3750G, 2851 and 3845 bootstrap loading programs are stored in read-only memory, and thus do not need to be sanitized.

The Cisco 3750G, 2851 and 3845 sanitization procedures include the following steps.

1. Connect the Sanitization Controller to the Cisco device to be sanitized.
2. Power cycle the device, and force it to boot from its ROM based bootstrap program.
3. Login to the device using administrative credentials.

4. Overwrite all of the device's internal flash memory.
5. Validate that all internal flash memory has been overwritten.
6. Configure the device's host name, IP address and the IP address of the Sanitization Controller's TFTP server.
7. Load a trusted copy of the device's operating system (IOS) from the Sanitization Controller's root of trust.
8. Copy the operating system software into the boot area of the internal flash memory.
9. Validate that the device has been properly sanitized.
10. Reboot the device using the trusted, local copy of IOS
11. Perform any device configuration steps required for proper operation, such as turning off spanning tree services.

6.7 Raritan Serial Console Manager Switch Sanitization

During CMAC cyber-experiments Raritan DSX serial console managers can be used to manage a wide variety of network devices including servers, routers, and switches. A Raritan DSX is also used during the sanitization process for Cisco 3750G layer 2/3 switches, Cisco 2851 and 3845 routers. Thus the CMAC Sanitization Controller must be able to sanitize Raritan DSX devices that have been allocated to experiments as well as the Raritan DSX device that is part of the CMAC sanitization system.

During the CMAC program JHU/APL worked closely with Raritan to develop a sanitization procedure that would restore the DSX to its original factory default configuration. This sanitization process clears and reformats the device's internal flash memory file system (including all slack space), installs a trusted copy of the device's operating system, and installs a trusted version of the device's bootstrap program.

The Raritan DSX sanitization procedure includes the following steps.

1. Connect the Sanitization Controller to the Raritan DSX device.
2. Power cycle the device.
3. Boot the device into Diagnostic Mode.
4. Configure the IP address of the device and the IP address of the TFTP server on the Sanitization Controller.
5. Overwrite the device's internal flash memory.
6. Validate that the internal flash memory has been sanitized.
7. Retrieve a trusted version of the device's root file system from the Sanitization Controller via TFTP.
8. Write the trusted version of the device's root file system to flash memory.
9. Retrieve a trusted version of the device's operating system software from the Sanitization Controller via TFTP.
10. Write the trusted version of the device's operating system software to flash memory
11. Retrieve a trusted version of the device's boot loader software from the Sanitization Controller via TFTP.
12. Write the trusted version of the device's boot loader software to flash memory
13. Validate that the device has been properly sanitized.

6.8 Sanitize the Sanitization System Devices

Upon successful completion of the device sanitization process the CMAC Sanitization Controller sends a network command to a Raritan Power Distribution Unit to remove power from each server, switch and router. The Sanitization Controller then sanitizes the Cisco 3750G switch and the Raritan DSX Serial Console Manager used during the sanitization process, and then removes power from these devices.

Finally, the Sanitization Controller sends a completion message along with the sanitization log files to the Range Controller. The Range Controller disconnects all devices from the sanitization network, archives the log files, and then removes power from the Sanitization Controller, which performs the final step of the sanitization process by sanitizing the Sanitization Controller. The entire CMAC sanitization process can be completed without any human interaction.

Chapter 7 - CMAC Sanitization System Performance

The Sanitization System performed extremely well during the CMAC system integration and testing efforts. As discussed previously, typically fourteen Dell servers, two Cisco switches, two Cisco routers and one Raritan switch are sanitized simultaneously. Over the course of a two month testing period the average amount of time required to automatically sanitize each device type was:

- (1) Raritan DSX Switch ~ 10 minutes
- (1) Cisco 3750G Switch ~ 20 minutes
- (1) Cisco 2851 Router ~ 20 minutes
- (1) Cisco 3845 Router ~ 20 minutes
- (1) Dell R310 Server ~ 55 minutes
- (1) Dell R410 Server ~ 55 minutes
- (1) Dell R610 Server ~ 60 minutes
- (14) Dell R310, R410 & R610 Servers ~ 70 minutes
- An entire rack of CMAC Equipment (14 Dell R310, R410 & R610 servers, 2 Cisco 3750G switches, 2 Cisco routers and 1 Raritan DSX switch) ~ 3 hours

For the Dell servers these are remarkably short periods of time to sanitize one TByte hard disk drives; 14 different areas of non-volatile memory; and restore the server to a trusted factory default configuration. The small differences in the amount of time required to sanitize each server type are related to several factors including the number of processors and cores in each server, and the size of the server's BIOS and iDRAC firmware images.

During the CMAC program a completely manual sanitization process was tested to measure the results of automating the sanitization process. Manually sanitizing 14 Dell servers, 2 Cisco switches, 2 Cisco routers and 1 Raritan switch took one engineer over 40 hours to complete. The same sanitization process was completed automatically by the

CMAC Sanitization System in just 3 hours, an order of magnitude decrease over the manual sanitization process.

Chapter 8 - Summary & Conclusions

This dissertation discussed the automated sanitization of servers, switches and routers to remove sensitive, proprietary or classified data as well as persistent malware. A number of Government laws and regulations require sanitization before a server or network device can be sold, recycled or repurposed. Proper device sanitization can be accomplished by overwriting all data in every area of non-volatile memory in the device, including hard drives, the BIOS, and rewritable firmware areas. If hard drive or flash memory data has been encrypted using AES, then these components can be sanitized by deleting the AES encryption key.

In order to remove persistent malware from a device, the sanitization process must be performed in the proper order, and in some cases portions of the sanitization process must be performed more than once. Determining the proper sanitization order requires a careful review of the device's hardware specifications, including any related Statements of Volatility, and an analysis of the boot order for all firmware and software used in the device. Once the proper sanitization order has been determined, a detailed sanitization process can be developed, and subsequently automated to reduce the amount of time required to execute the sanitization process. In general the device sanitization process includes the following steps.

1. Establish a root of trust
2. Sanitize each device component in the proper order
3. Validate that each device component has been properly sanitized
4. Restore each device component to a trusted state
5. Restore the device's configuration data
6. Validate that the device has been restored to a trusted state and has been properly configured

7. Sanitize all equipment used during the device sanitization process
8. Save the sanitization log data for future reference

During the CMAC program JHU/APL designed, developed, implemented and tested an automated, rapid and secure system for sanitizing a large number of Dell PowerEdge servers, Cisco switches and routers, and Raritan Serial Console Management switches. The use of second generation Seagate Self-Encrypting Drives was a key enabling technology that allowed CMAC to quickly and securely sanitize large hard disk drives containing classified data. The sanitization process also rapidly sanitized and restored all of the NVRAM used in these devices to a trusted factory default configuration.

8.1 The CMAC Sanitization System Met Its Requirements

Each of the DARPA National Cyber Range sanitization requirements were met by the CMAC Sanitization System. All non-volatile memory areas within each CMAC device are overwritten and restored to a trusted state, the device sanitization process is automated and does not require human intervention, and devices that were used to process classified data up to the TS//SCI level can be reused at an unclassified level.

The CMAC automated sanitization process is very fast, requiring just 3 hours to properly sanitize 14 Dell servers, 2 Cisco Switches, 2 Cisco Routers and 1 Raritan Switch. Manually sanitizing these devices would take an engineer over 40 hours to complete. Thus the CMAC Sanitization System reduced the time required to sanitize these devices by more than an order of magnitude.

8.2 Issues Related to Implementing the Automated Sanitization System Architecture

The most difficult issues encountered during CMAC's implementation of the Automated Sanitization System Architecture was determining the precise set of steps, and

their correct order, to properly sanitize each server, switch and router used in the CMAC system. In some cases this was a simple process since a vendor had already developed and published procedures for sanitizing their device. For example Cisco had published a detailed step by step procedure to sanitize a Catalyst 4500 switch (Cisco 2005). This procedure was used as a starting point to develop the detailed step by step procedures required to sanitize the Cisco 3750G switch.

In other cases this was a complex process that required working with a vendor for a long period of time. For example the process and custom software programs required to automatically sanitize Dell PowerEdge servers were defined and executed under a contract between JHU/APL and Dell. A Statement of Work was developed that outlined the requirements for sanitizing three Dell server models, a contract was written and put in place, and a six month design, development and test effort was undertaken by Dell to deliver the PXE Network Boot Programs, sanitization shell scripts, sanitization programs, and firmware update programs required to completely sanitize these servers. Dell delivered alpha, beta and final versions of these programs, and JHU/APL tested each version with the CMAC Sanitization Controller to insure they were working properly. As with any custom software development effort some problems were discovered during system testing which required the development team at Dell to make changes to the software. All of the problems were eventually resolved and the final versions of the Dell PXE Network Boot Programs, sanitization shell scripts, sanitization programs, and firmware update programs worked well with the CMAC Sanitization System.

8.3 Dissertation Hypotheses Conclusions

This dissertation has shown, through the use of a case study of the CMAC Sanitization System, that it is possible to remove all sensitive, proprietary or classified information, as well as persistent malware from servers, switches and routers through the use of a well-defined sanitization process. The dissertation has also shown that the amount of time required to sanitize servers, switches and routers can be significantly reduced by automating and parallelizing a well-defined sanitization process within a sanitization system. The dissertation working hypotheses H_1 and H_2 have been shown to be true, and the null hypothesis H_0 has been shown to be false.

Chapter 9 - Areas for Future Research

The Automated Sanitization System Architecture presented in section 5 is designed to sanitize a wide variety of networked devices. If a device can be sanitized via a network connection and/or a serial console port, then the architecture allows adding this device type to the roster of supported devices. Specific implementations of the Automated Sanitization System architecture, such as the CMAC Sanitization System presented in section 6, will be limited to a subset of device types and models for which up-front engineering analysis and development work has been completed, resulting in a defined sanitization process and a set of sanitization programs for those devices.

For example the CMAC Sanitization System can be used to completely sanitize any Dell R310, R410 or R610 server, but it cannot currently be used to sanitize any other type of Dell server, or servers from another vendor. In order to use the CMAC Sanitization System to sanitize a Dell R710 server (or another vendor's server) a number of engineering efforts must be completed. These include analyzing the new server's statement of volatility to determine what components must be sanitized; analyzing the boot order to determine the proper order for sanitizing the components; development of a custom PXE Network Boot Program; and development of custom sanitization shell scripts and programs. Once these engineering efforts have been completed, the CMAC Sanitization System can be upgraded to sanitize this new type of server. A similar set of engineering efforts would need to be completed to allow the CMAC Sanitization System to support a different Cisco switch or router, or another vendor's switch or router.

One suggested area for future research is to develop a standard for the remote sanitization of information technology devices. Simply put the sanitization process

would be better, faster, and more reliable if a device was capable of sanitizing itself in response to a remote command from a network or serial console interface. Implementation of self-sanitization functions could increase a device's power, space, weight and cooling requirements, and most certainly could increase the device's development and production cost. The potential benefits of simplifying the device sanitization process would have to be weighed against the increased costs to provide these features in a variety of information technology devices. There is some hope that information technology vendors will understand the need for self-sanitizing devices and undertake to add this feature as a product discriminator. In the past several years many vendors have implemented a brick function that can remotely sanitize and disable a cell phone if it has been lost or stolen (Fledel et al. 2012). The authors recommend that additional research be undertaken to develop information technology remote sanitization standards or recommended practices, and to encourage IT vendors to offers these functions in their future server, switch and router product lines.

Another suggested area for future research is to extend the Automated Sanitization System design to work with servers from other major vendors such as IBM and HP, and to work with routers and switches from other vendors such as Juniper and Brocade. Most modern server architectures support PXE booting via a network connection, and most modern routers and switches provide access to a low-level, read-only operating system via a serial console connection. Thus there is every reason to believe that the Automated Sanitization System architecture described in this dissertation, and implemented on the JHU/APL CMAC program can be easily extended to work with many different vendor's servers, routers and switches.

References

Alexander, Steve, and Ralph Droms. "DHCP options and BOOTP Vendor Extensions." (1997).

Arora, Mohit. “How secure is AES against brute force attacks?” accessed January 310, 2014, http://www.eetimes.com/document.asp?doc_id=1279619, (2014)

Bogdanov, Andrey, Dmitry Khovratovich, and Christian Rechberger. "Biclique cryptanalysis of the full AES." In Advances in Cryptology–ASIACRYPT 2011, pp. 344-371. Springer Berlin Heidelberg, 2011.

Bonkoski, Anthony J., Russ Bielawski, and J. Alex Halderman. "Illuminating the security issues surrounding lights-out server management." In Proceedings of the 7th USENIX conference on Offensive Technologies, pp. 10-10. USENIX Association, 2013.

Cisco. “Resetting Catalyst Switches to Factory Defaults”, accessed May 26, 2013, http://www.cisco.com/image/gif/paws/24328/156.pdf (2005)

Cisco. “Cisco Security Advisories, Responses, and Notices”, accessed May 26, 2013, http://tools.cisco.com/security/center/publicationListing.x#~CiscoSecurityNotice

Cui, Ang, Jatin Kataria, and Salvatore J. Stolfo. "Killing the Myth of Cisco IOS Diversity." USENIX Workshop on Offensive Technologies, California, USA (2011).

DARPA Broad Agency Announcement (BAA), National Cyber Range, Strategic Technology Office (STO), DARPA-BAA-08-43. (2008)

Dell. “Dell™ PowerEdge™, R610 Systems, Hardware Owner’s Manual (2013)

Deutch, J. M. "DoD 5220.22-M National Industrial Security Program Operating Manual." (2004).

D. S. S. "Clearing and Sanitization Matrix as of: June 28, 2007, Department of Defence/Defense Security Service, 2007."

Extreme Engineering Solutions, “XPedite 7330 Block Digram”, accessed February 115, 2014, http://www.xes-inc.com/products/view/xpedite7330/

FIPS, PUB. "197." Advanced Encryption Standard (AES) 26 (2001).

Fledel, Yuval, Asaf Shabtai, Dennis Potashnik, and Yuval Elovici. "Google Android: An Updated Security Review." In Mobile Computing, Applications, and Services, pp. 401-414. Springer Berlin Heidelberg, 2012.

Garfinkel, Simson L., and Abhi Shelat. "Remembrance of data passed: A study of disk sanitization practices." Security & Privacy, IEEE 1, no. 1 (2003): 17-27.

Ge, Livian (2011) “BIOS Threat is Showing up Again!”, Symantec Official Blog, accessed May 26, 2013, http://www.symantec.com

Gutmann, Peter. "Secure deletion of data from magnetic and solid-state memory." In Proceedings of the Sixth USENIX Security Symposium, San Jose, CA, vol. 14. 1996.

Hathaway, Lynn. "National policy on the use of the advanced encryption standard (AES) to protect national security systems and national security information." National Security Agency (2003).

Heasman, John. "Implementing and detecting an acpi bios rootkit." Black Hat Federal 368 (2006).

Heasman, John , 2006, “Implementing and Detecting a PCI Rootkit”. Next Generation Security Software

Hughes, Gordon, and Tom Coughlin. "Tutorial on disk drive data sanitization." (2006) Accessed May 26, 2013, http://cmrr.ucsd.edu/people/Hughes/documents/DataSanitizationTutorial.pdf

Hughes, Gordon F., Tom Coughlin, and Daniel M. Commins. "Disposal of disk and tape data by secure sanitization." Security & Privacy, IEEE 7, no. 4 (2009): 29-34.

Intel, P. X. E. "Preboot execution environment (PXE) specification." Intel Corporation (1999).

Joukov, Nikolai, and Erez Zadok. "Adding secure deletion to your favorite file system." In Security in Storage Workshop, 2005. SISW'05. Third IEEE International, pp. 8-pp. IEEE, 2005.

Joukov, Nikolai, Harry Papaxenopoulos, and Erez Zadok. "Secure deletion myths, issues, and solutions." In Proceedings of the second ACM workshop on Storage security and survivability, pp. 61-66. ACM, 2006.

Kauer, Bernhard. OSLO: Improving the security of Trusted Computing. Technical report, Technische Universitt Dresden, Department of Computer Science, 2007.

Kissel, Richard, Matthew Scholl, Steven Skolochenko, and Xing Li. "Guidelines for media sanitization." NIST Special Publication 800 (2006): 88.

Lynn, Michael. “The Holy Grail: Cisco IOS Shellcode And Exploitation Techniques" 2005, accessed May 26, 2013, http://boingboing.net/2005/07/29/michael-lynns-contro.html

McAfee Labs, 2013, McAfee Threats Report: Fourth Quarter 2012, pp. 6. Accessed May 26, 2013, http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2012.pdf

Muniz, Sebastian. "Killing the myth of Cisco IOS rootkits: DIK, 2008.", accessed May 26, 2013, http://eusecwest.com/esw08archive.html

Narine, Ryan, 2010, “Dell ships motherboard with malicious code”, Accessed May 16, 2013, http://www.ZDNet.com

NIST FIPS. "140-2: Security requirements for cryptographic modules." Information Technology Laboratory, National Institute of Standards and Technology (2001).

NSA/CSS, 2006, STORAGE DEVICE DECLASSIFICATION MANUAL (2006-5-13), Section 5.a.1, pp 4.

Parno, Bryan. "Bootstrapping trust in a “trusted” platform." In Proceedings of the 3rd conference on Hot topics in security, pp. 1-6. USENIX Association, 2008.

Peterson, Zachary NJ, Randal Burns, Joe Herring, Adam Stubblefield, and Aviel Rubin. "Secure deletion for a versioning file system." In Proceedings of the USENIX Conference on File And Storage Technologies (FAST), pp. 143-154. 2005.

Privacy Rights Clearinghouse, Chronology of Data Breaches. Acccesed on May 26, 2103, http://www.privacyrights.org/data-breach

Reardon, Joel, David Basin, and Srdjan Capkun. "SoK: Secure Data Deletion." 2013 IEEE Symposium on Security and Privacy

Sachs, Marcus. “Supply Chain Attacks”. Available at http://crissp.poly.edu/media/sachs_slides.pdf

Sandberg, Russel, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. "Design and implementation of the Sun network filesystem." In Proceedings of the Summer USENIX conference, pp. 119-130. 1985.

Seagate Technology LLC. “Self-Encrypting Drives for Servers, NAS and SAN Arrays”, 2009

Seagate Technology LLC. “FIPS 140-2 Standard and Self-Encrypting Drive Technology”, 2010

Sollins, K. "The TFTP protocol (revision 2)." (1992).

Trusted Computing Group. TPM Specification version 1.2. Parts 1-3 (2007).

US DHHS. Accessed May 26, 2013, http://www.hhs.gov/ocr/privacy/hipaa/administrative/breachnotificationrule/breachtool.html

Wei, Michael, Grupp, Laura, Spada, Frederick and Swanson, Steven, 2011, “Reliably Erasing Data From Flash-Based Solid State Drives”, Proceedings of the 9th USENIX conference on File and storage technologies, Berkeley, CA, USA, 2011, pages 1-13

Wright, Craig, Dave Kleiman, and Shyaam Sundhar RS. "Overwriting hard drive data: The great wiping controversy." In Information Systems Security, pp. 243-257. Springer Berlin Heidelberg, 2008.

Appendix A – Sample Statement of Volatility

VOLATILITY STATEMENT

Component Classification

Classification	Name	Description
1	Simple	No Non-Volatile Memory, No Battery Back RAM, No Transmitters (ex .Cables, Chassis, Power Supplies)
2	Flash	External/Internal Flash Components (ex. Flash SIMMS, Flash Cards)
3	Processing	Built in NVRAM, Built in Flash and Possible External Flash (ex. Route Processors, Supervisors)
N/A	Not Applicable	Components like software and documentation where a non-volatility statement is not required
N/C	Not Covered	Components not covered by this non-volatility statement

The associated volatility spreadsheet list items by part number and provides a description and non-volatility classification. An empty classification indicates a non-volatility classification has not yet been determined for the component.

Questions and Answers

1. Are all memory components within the hardware device volatile, meaning that any data stored on these components is lost when power to the unit is removed?

Type 1: Yes.
Type 2/3: No.

2. If non-volatile components exist, are any of them designed to be modified by the user or by the devices during normal operations? Please briefly describe the type of data stored in these components.

Type 1: No non-volatile components exist.
Type 2/3: The non-volatile components are not modified during normal operations. Only during reconfiguration, system software upgrades and, in some cases, system failures are the flash components modified. Reconfiguration and software upgrades, by default, require privileged access to the device (This default behavior can be modified). The information stored in the non-volatile memory includes device configuration information, operating system software and crash files. The non-volatile components are not in the data path. At the appropriate privilege level mechanisms exist in the device to copy non-volatile memory files to and from other network enabled devices. These are privileged

activities and are multi step processes and therefore would not likely occur accidentally. These mechanisms are properly used to save and update operating system software; to save and restore system configurations; and to copy crash files for offline review. Crash files are only written flash when the device is specifically configured to do so. These files could potentially contain information from the data path.

3. Are any RAM components battery-backed? If so, please briefly describe the nature of the data being retained and location of the memory.

Type 1: No.
Type 2: No.
Type 3: Yes. The data stored in these components during normal operation include device configuration information. The battery-backed RAM components are not in the data path. Mechanisms exist in the device to copy the battery-backed configurations files to and from other network enabled devices. These are privileged level activities and are multi step processes and therefore would not likely occur accidentally. These mechanisms are properly used to save and restore system configurations.

4. Where is the BIOS located? Can it be locked out with a password? If yes, please provide the command sequence to do so.

Type 1/2/3: Not applicable.

5. Does this equipment contain any devices, such as RF transmitters and dial out capabilities via either telephone landline or cellular transmission?

Type 1/2/3: No RF transmitters are contained in any of these components. Some of these components possess interfaces that could be connected to a telephone line for dial out capability; in most cases this would require the addition of an external modem device.

Clearing Procedures
The following procedures are used by the community for clearing non-volatile and battery-backed memory on the aforementioned Cisco components, prior to shipment back to the vendor.

From privileged mode, “write erase” (IOS) or “clear config all” and “delete vlan.dat” (CatOS) to clear battery-backed memory on Type 3 components. And “format [device]:” to clear non-volatile memory on Type 2/3 components. Additionally, in many cases, non-volatile memory (Flash) is on removable flash cards or SIMMs. Procedures for removal are provided in the specific device hardware installation and maintenance guide.

Appendix B – Dell Server Sanitization Software Design Diagrams

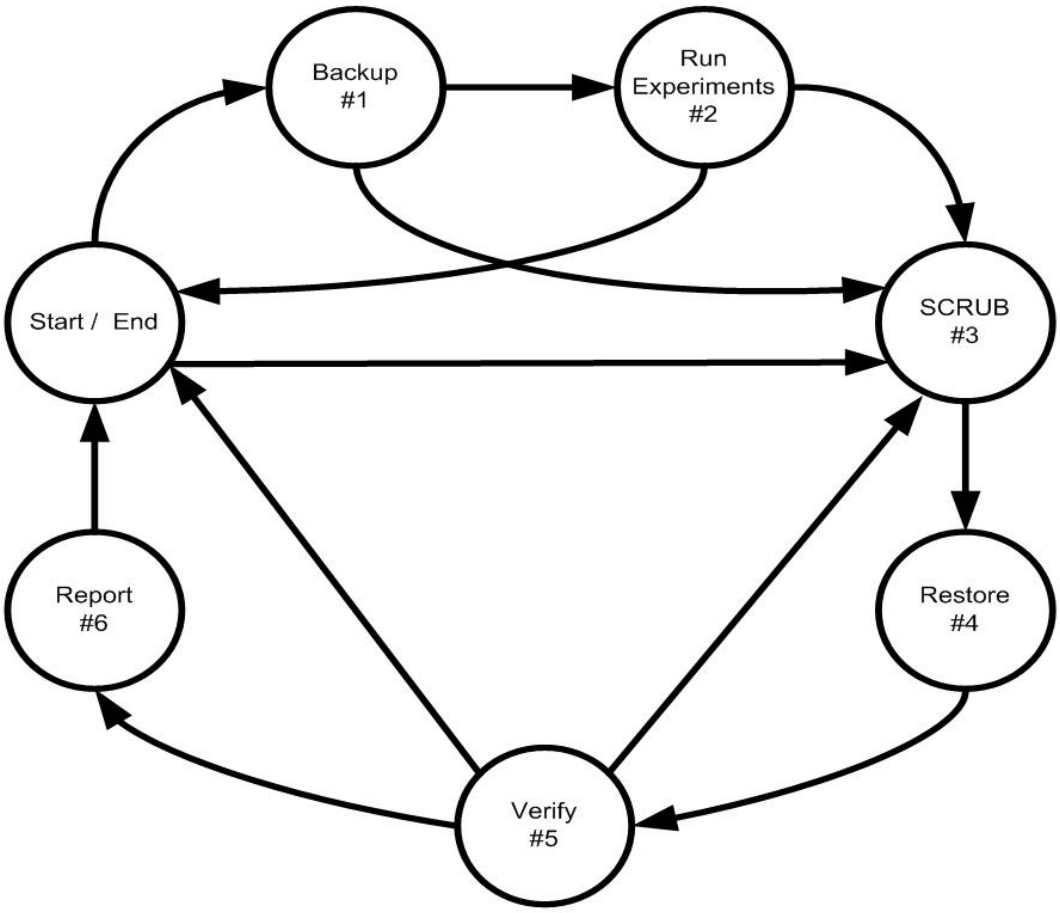


Figure B-1: Dell Server Sanitization Software State Diagram

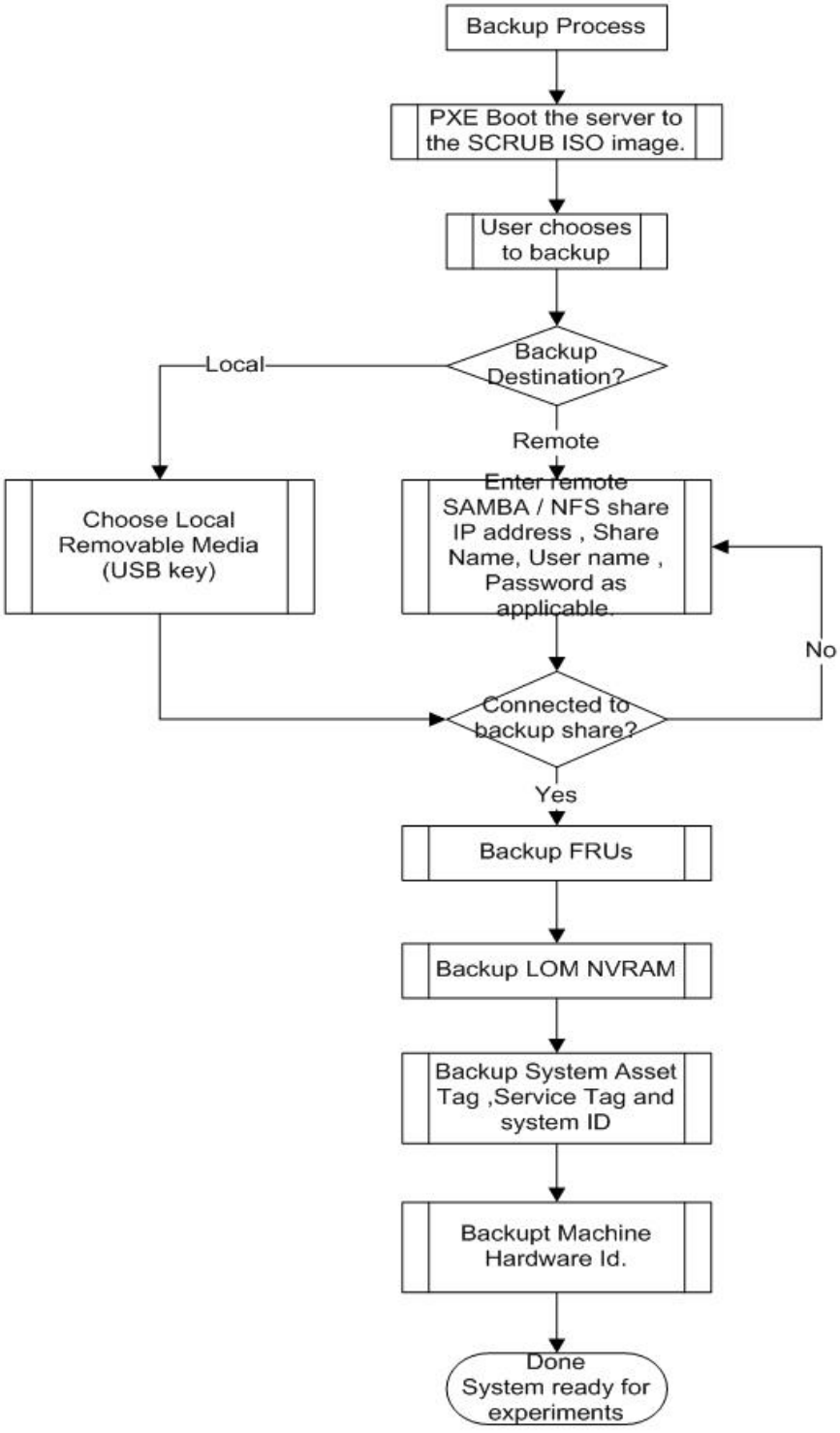


Figure B-2: Dell Server Sanitization Backup Mode Flow Chart

Figure B-3: Dell Server Sanitization Software Flow Chart

Appendix C – Sample Dell Server Sanitization Script

```
#Scrub/Backup Utility.
#2011 Dell Inc. All Rights Reserved.


#!/bin/sh

#check to see if the NFSShare folder exist at the /mnt point

if [ -d /mnt/NFSShare ]
then
        nfs_mount=/mnt/NFSShare
        if [ -f $nfs_mount/config.txt ]
        then
                action=`cat $nfs_mount/config.txt | grep action | cut -d '=' -f 2`
        else
                sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4` "NFS MOUNT"
"config.txt not found. Please make sure it exist." $nfs_mount/tracelog
                echo "config.txt not found. Please make sure it exist."
                echo "Exiting"
                exit -1
        fi
else

#       sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4` "NFS Mount" "Not able to find the
mount share. Exiting" $nfs_mount/tracelog

        echo "Not able to find the mount share. Exiting"
        exit -1
fi
        match_result=0
        BackupLOMFlag=0
        BackupFRUFlag=0
        BackupLCFlag=0

#       ScrubBMCFlag=0
        ScrubPSUFlag=0
        ScrubSASBPFlag=0
        ScrubRAIDFlag=0
        ScrubHDDFlag=0
#       ScrubDVDFlag=0

        ScrubLOMFlag=0
        ScrubFRUFlag=0
```

```
        ScrubBIOSiDRACFlag=0
#       ScrubiDRACFlag=0

        ScrubState=0


#Backup Data function. This function gathers the vital system information and places it in the info file on the server shared backup location. It also creates the backup directories to save the backup data.

backup_data ()
{
# check to see if backupconfig.txt exist. If it does then backup and copy back again to preserve the settings

sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP PROCESS" "Backup process started successfully" $destination/backuplog

        if [ -f $destination/backupconfig.txt ]
        then
                cp  $destination/backupconfig.txt /tmp/

                cp /tmp/backupconfig.txt $destination/
                rm -f /tmp/backupconfig.txt
        fi
# Create a new sysconfig, each time the backup executes.

        if [ -f $destination/sysinfo.txt ]
        then
                rm $destination/sysinfo.txt
        fi

        echo "Backing up to $destination"
        echo "Platform="`getSystemId | grep PowerEdge | cut -d ' ' -f 4` > $destination/sysinfo.txt
        echo "Service_Tag="`getSystemId | grep Tag | cut -d ' ' -f 4` >> $destination/sysinfo.txt
        echo "MB_SN="`dmidecode | grep ..CN | cut -d ' ' -f 3`>> $destination/sysinfo.txt
        echo "Asset_Tag="`assetTag | cut -d ' ' -f 4`>> $destination/sysinfo.txt
        echo "Creating Backup Directories"

#check to see if the conditions for Backing up LOM and FRU are true or false. If true, then backup that section, else skip

        if [ $BackupLOMFlag == 0 ]
        then
                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP LOM DATA" "LOM Backup started successfully" $destination/backuplog

                echo "LOM Backup in Progress"
                rm -f -r $destination/LOM/
```

```
                mkdir $destination/LOM
                sh ./LOMBackup.sh $destination

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP LOM DATA"
"LOM Backup completed successfully" $destination/backuplog

        else
                echo "LOM not being backed up"
        fi

        if [ $BackupFRUFlag == 0 ]
        then

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP FRU DATA"
"FRU Backup started successfully" $destination/backuplog

                echo "FRU Backup in Progress"
                rm -f -r $destination/FRU/
                mkdir $destination/FRU
                sh ./FRUBackupTest.sh $destination

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP FRU DATA"
"FRU Backup completed successfully" $destination/backuplog

        else
                echo "FRU not being backed up"
        fi

        if [ $BackupLCFlag == 0 ]
        then

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP LC DATA"
"LC Backup started successfully" $destination/backuplog

                echo "LC Backup in Progress"
                rm -f -r $destination/LC
                mkdir $destination/LC
                chmod -R 777 $destination
                # backup the LC partitions #1 and 2
                sh ./BackupLC.sh 0 $destination

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP LC DATA"
"LC Backup completed successfully" $destination/backuplog

        else
                echo "LC not being backed up"
        fi

#       echo $destination
```

```
	chmod -R 777 $destination

	sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP PROCESS" "Backup process completed successfully" $destination/backuplog

	echo "Backup Process Completed."
}



#Function specific to R310 servers, this would determine the data to be stored, and which directory needs to be looked for exact backup data. Execute the restoration scripts

restore_data_R310 ()
{
	echo "This is the Restoration process for R310"
	echo "Platform R310 found!!"
	SERVICE_TAG=`getSystemId | grep Tag: | cut -d ' ' -f 4`
	ASSET_TAG=`assetTag | grep Tag: | cut -d ' ' -f 4`
	echo "Service Tag is " $SERVICE_TAG
	echo "Asset Tag is " $ASSET_TAG
	sh ./DUPRestore.sh $SERVICE_TAG $nfs_mount $target_arch $BackupSource $ScrubFlag $ScrubState $arch

}



#Function specific to R410 servers, this would determine the data to be stored, and which directory needs to be looked for exact backup data. Execute the restoration scripts

restore_data_R410 ()
{
	echo "This is the Restoration process for R410"
	echo "Platform R410 found!!"
	SERVICE_TAG=`getSystemId | grep Tag: | cut -d ' ' -f 4`
	ASSET_TAG=`assetTag | grep Tag: | cut -d ' ' -f 4`
	echo "Service Tag is" $SERVICE_TAG
	echo "Asset Tag is" $ASSET_TAG
	sh ./DUPRestore.sh $SERVICE_TAG $nfs_mount $target_arch $BackupSource $ScrubFlag $ScrubState $arch

}



#Function specific to R610 servers, this would determine the data to be stored, and which directory needs to be looked for exact backup data. Execute the restoration scripts
```

```
restore_data_R610 ()
{
        echo "This is the Restoration process for R610"
        echo "Platform R610 found!!"
        SERVICE_TAG=`getSystemId | grep Tag: | cut -d ' ' -f 4`
        echo "Service Tag is "$SERVICE_TAG
        echo "Asset Tag is" $ASSET_TAG
        sh ./DUPRestore.sh $SERVICE_TAG $nfs_mount $target_arch $BackupSource $ScrubFlag
$ScrubState $arch

}

#This section would do match of Critical System information.

match_id ()
{

        ls /mnt/NFSShare/PEbackup/ | grep SVCTAG > /tmp/systems

        count=0

        for readline in `cat /tmp/systems`
        do
#               echo hello $readline
                backup_path=/mnt/NFSShare/PEbackup/$readline/
                TMPCN_NUM=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt | grep MB_SN | cut
-d '=' -f 2`
                target_platform=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt | grep Platform | cut
-d '=' -f 2`
                target_svctag=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt | grep Service_Tag |
cut -d '=' -f 2`
                target_assettag=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt | grep Asset_Tag |
cut -d '=' -f 2`
                count=`expr $count + 1`
#               echo "count = $count "

                SERVICE_TAG=`getSystemId | grep Tag: | cut -d ' ' -f 4`
                ASSET_TAG=`assetTag | grep Tag: | cut -d ' ' -f 4`
                if [ $TMPCN_NUM == $target_ID ]
                then
                        echo "System IDENTIFIER Matched!!!"
                        match_result=1

                        if [ $SERVICE_TAG == $target_svctag ]
                        then
                                echo "Service tag Matched !! No need to change Service Tag"
                                echo "Platform = "$target_platform
```

```
                        echo "service tag = "$target_svctag
                        echo "Stored Backup Path = "$backup_path
                else
                        echo "Service tag mismatch!! Reprogramming the Service Tag"

                        serviceTag -s $target_svctag

        #reassign the original value of service tag read from the backup to the system
                        SERVICE_TAG=$target_svctag
                fi

                if [ -z $target_assettag ]
                then
                        assetTag -s ""
                return

                else

                if [ -z $ASSET_TAG ]
                then
                        echo "Asset Tag is Null" $ASSET_TAG
                        assetTag -s $target_assettag
                return

                if [ $ASSET_TAG == $target_assettag ]
                then
                        echo "Asset tag Match found!!"
                else
                        echo "Asset Tag found on this system is" $ASSET_TAG
                        echo "Asset tag information incorrect!! Fixing it!"
                        echo "Correct asset tag is" $target_assettag
                        assetTag -s $target_assettag
                        echo "Asset Tag Updated"
                fi
                fi
                fi
                return
        else
                echo "System backup not found!!"
                echo "Please make sure the backup folder SVCTAG_$SERVICE_TAG is
intact"
        fi

    done
}


case "$action" in
```

```
        backup)
                echo "----------------------"
                echo "This is Backup Process"
                echo "----------------------"

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP ACTION"
"Backup Process Started" $nfs_mount/tracelog


                SERVICE_TAG=`getSystemId | grep Tag: | cut -d ' ' -f 4`

#Following Condition to check the existence of a service tag, while backing up. If service tag field is
empty, then backup won't work

                if [ -z $SERVICE_TAG ]
                then
                        echo "ERROR! Service Tag value cannot be null while backing up the system."
                        echo "Make sure that proper Service Tag exist on this system. Please try again
once confirmed."

                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4` "SERVICE TAG
CHECK" "ERROR! Service Tag value cannot be null while backing up the system." $nfs_mount/tracelog

                        exit -1
                else
                        echo "Service Tag found is "$SERVICE_TAG

                        sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "SERVICE TAG
CHECK" "Service Tag value positive." $nfs_mount/tracelog

#Backup check whether the Backup Folder containing the service tag of the system exists on the server

                        if [ -d $nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG ]
                        then
                                echo "Located backup of the System"

                                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"EXISTING BACKUP FOLDER" "Backup Located on NFS server" $nfs_mount/tracelog

                                destination=$nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG

#if system found, then it checks for the backupconfig.txt file, and the options present in it, with regards to
backup for LOM and FRU.

                                if [ -f $destination/backupconfig.txt ]
                                then
                                        echo "Found the backupconfig file!!"
```

```
                    sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"EXISTING BACKUP CONFIG FILE" "Found the backupconfig file" $destination/backuplog

                    BACKUP_LOM=`cat $destination/backupconfig.txt | grep
BACKUP_LOM | cut -d '=' -f 2`
                    echo "LOM BACKUP = $BACKUP_LOM"

                  BACKUP_FRU=`cat $destination/backupconfig.txt | grep
BACKUP_FRU | cut -d '=' -f 2`
                    echo "FRU BACKUP = $BACKUP_FRU"

                    BACKUP_LC=`cat $destination/backupconfig.txt | grep
BACKUP_LC | cut -d '=' -f 2`
                    echo "LC BACKUP = $BACKUP_LC"

#setting flag for LOM and FRU, this information will be passed to the backup function when backing up
the data.

                    if [ $BACKUP_LOM == true ]
                    then
                            BackupLOMFlag=0
                            echo "LOM will be backed up"

                    elif [ $BACKUP_LOM == false ]
                    then
                            BackupLOMFlag=1
                            echo "LOM will not be backed up"
                    else
                            echo "Invalid value for BACKUP_LOM. Cannot
continue ahead!!"

                            sh ./logging_script.sh 1 `getSystemId | grep Tag | cut
-d ' ' -f 4` "BACKUP CONFIG VALUES" "Invalid argument passed for BACKUP_LOM. Cannot continue
ahead!!" $destination/backuplog

                            exit -1
                    fi
                            echo "Backup LOM value =" $BACKUP_LOM

                    if [ $BACKUP_FRU == true ]
                    then
                            echo "FRU will be backed up"
                            BackupFRUFlag=0
                    elif [ $BACKUP_FRU == false ]
                    then
                            echo "FRU will not be backed up"
                            BackupFRUFlag=1
                    else
```

```
                                        echo "Invalid value for BACKUP_FRU. Cannot
continue ahead!!"

                                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut
-d ' ' -f 4` "BACKUP CONFIG VALUES" "Invalid value for BACKUP_FRU. Cannot continue ahead!!"
$destination/backuplog

                                        exit -1
                                fi
                                        echo "Backup FRU value =" $BACKUP_FRU

                                if [ $BACKUP_LC == true ]
                                then
                                        echo "LC will be backed up"
                                        BackupLCFlag=0

                                elif [ $BACKUP_LC == false ]
                                then
                                        echo "LC will not be backed up"
                                        BackupLCFlag=1
                                else
                                        echo "Invalid value for BACKUP_LC. Cannot
continue ahead!!"

                                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut
-d ' ' -f 4` "BACKUP CONFIG VALUES" "Invalid value for BACKUP_LC. Cannot continue ahead!!"
$destination/backuplog

                                        exit -1
                                fi
                                        echo "Backup LC value =" $BACKUP_LC

                        else

        # create config file for backup, if it doesnt exist
                                echo "Cannot locate backup config file"
                                echo "configuring: $destination/backupconfig.txt"
                                touch $destination/backupconfig.txt
                                echo "BACKUP_LOM=true" > $destination/backupconfig.txt
                                echo "BACKUP_FRU=true" >> $destination/backupconfig.txt
                                echo "BACKUP_LC=true" >> $destination/backupconfig.txt
                                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"BACKUP CONFIG SCRIPT" "No existing backup config file found. Default backup config file created
successfully" $destination/backuplog

                        fi

                        if [ -f $destination/scrubconfig.txt ]
                        then
```

```
                                echo "Scrub Config Found"
                        else
                                echo "Cannot locate scrub config file. Assuming default
Credentials. Making one."
                #               echo "SCRUB_BMC=true" > $destination/scrubconfig.txt
                                echo "SCRUB_PSU=true" >> $destination/scrubconfig.txt
                                echo "SCRUB_SASBP=true" >> $destination/scrubconfig.txt
                                echo "SCRUB_RAID=true" >> $destination/scrubconfig.txt
                                echo "SCRUB_HDD=true" >> $destination/scrubconfig.txt
                #               echo "SCRUB_DVD=true" >> $destination/scrubconfig.txt
                                echo "SCRUB_LOM=true" >> $destination/scrubconfig.txt

                                echo "SCRUB_FRU=true" >> $destination/scrubconfig.txt

                                echo "SCRUB_BIOS_iDRAC=true" >>
$destination/scrubconfig.txt


                        fi

                else

# If no backup exist for the given service tag, create the folder and structure from scratch

                        echo "System not found ... Creating folder"
                        mkdir $nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG
                        destination=$nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG
                        echo "Cannot locate config file"
                        echo "configuring: $destination/backupconfig.txt"
                        touch $destination/backupconfig.txt
                        echo "BACKUP_LOM=true" > $destination/backupconfig.txt
                        echo "BACKUP_FRU=true" >> $destination/backupconfig.txt
                        echo "BACKUP_LC=true" >> $destination/backupconfig.txt

                #       echo "SCRUB_BMC=true" > $destination/scrubconfig.txt
                        echo "SCRUB_PSU=true" >> $destination/scrubconfig.txt
                        echo "SCRUB_SASBP=true" >> $destination/scrubconfig.txt
                        echo "SCRUB_RAID=true" >> $destination/scrubconfig.txt
                        echo "SCRUB_HDD=true" >> $destination/scrubconfig.txt
                #       echo "SCRUB_DVD=true" >> $destination/scrubconfig.txt
                        echo "SCRUB_LOM=true" >> $destination/scrubconfig.txt

                        echo "SCRUB_FRU=true" >> $destination/scrubconfig.txt

                        echo "SCRUB_BIOS_iDRAC=true" >> $destination/scrubconfig.txt


                fi
```

```
                fi
                backup_data

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "BACKUP ACTION"
"Backup Process Completed successfully" $nfs_mount/tracelog

        ;;

        scrub)
                echo "Scrub Process Started"

                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "SCRUB ACTION"
"Scrub Process Started successfully" $nfs_mount/tracelog

                target_ID=`dmidecode | grep ..CN | cut -d ' ' -f 3`
                match_id
                rm -f /tmp/systems

                if [ $match_result == 1 ]
                then

                        sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "SYSTEM
BACKUP" "System Backup found." $nfs_mount/tracelog

                        BackupSource=$nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG
                        chmod -R 777 $BackupSource
                        match_result=0

                        if [ -f $BackupSource/scrubstateflag ]
                        then

                                echo "Found the scrub state file!!"
                                ScrubState=`cat $BackupSource/scrubstateflag | grep ScrubState | cut -
d '=' -f 2`
                                echo "Scrub State = $ScrubState"
                        else
                                echo "ScrubState=0" > $destination/scrubstateflag
                                ScrubState=0
                        fi

                        if [ -f $BackupSource/scrubconfig.txt ]
                        then
                                echo "Found the scrubconfig file!!"
#                               SCRUB_BMC=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_BMC | cut -d '=' -f 2`
#                               echo "SCRUB BMC = $SCRUB_BMC"
                                SCRUB_PSU=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_PSU | cut -d '=' -f 2`
#                               echo "SCRUB PSU = $SCRUB_PSU"
```

```
                                SCRUB_SASBP=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_SASBP | cut -d '=' -f 2`
#                               echo "SCRUB SASBP = $SCRUB_SASBP"
                                SCRUB_RAID=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_RAID | cut -d '=' -f 2`
#                               echo "SCRUB RAID = $SCRUB_RAID"

                                SCRUB_HDD=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_HDD | cut -d '=' -f 2`
#                               echo "SCRUB HDD = $SCRUB_HDD"
#                               SCRUB_DVD=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_DVD | cut -d '=' -f 2`
#                               echo "SCRUB DVD = $SCRUB_DVD"
                                SCRUB_LOM=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_LOM | cut -d '=' -f 2`
#                               echo "SCRUB LOM = $SCRUB_LOM"
                                SCRUB_FRU=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_FRU | cut -d '=' -f 2`
#                               echo "SCRUB FRU = $SCRUB_FRU"
                                SCRUB_BIOS_iDRAC=`cat $BackupSource/scrubconfig.txt | grep
SCRUB_BIOS_iDRAC | cut -d '=' -f 2`
#                               echo "SCRUB BIOS = $SCRUB_BIOS_iDRAC"



                                if [ $SCRUB_PSU == true ]
                                then
                                        ScrubPSUFlag=0
                                        echo "PSU will be scrubed"

                                elif [ $SCRUB_PSU == false ]
                                then
                                        ScrubPSUFlag=1
                                        echo "PSU will not be scrubbed"
                                else
                                        echo "Invalid argument passed for SCRUB_PSU. Cannot
continue ahead!!"
                                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB CONFIG VALUES" "Invalid argument passed for SCRUB_PSU. Cannot continue ahead!!"
$BackupSource/scrublog
                                        exit -1
                                fi
                                        echo "Scrub PSU value =" $SCRUB_PSU

                                if [ $SCRUB_SASBP == true ]
                                then
                                        ScrubSASBPFlag=0
                                        echo "SASBP will be scrubed"
```

```
elif [ $SCRUB_SASBP == false ]
then
        ScrubSASBPFlag=1
        echo "SASBP will not be scrubbed"
else
        echo "Invalid argument passed for SCRUB_SASBP. Cannot
continue ahead!!"
        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB CONFIG VALUES" "Invalid argument passed for SCRUB_SASBP. Cannot continue ahead!!"
$BackupSource/scrublog
        exit -1
fi
        echo "Scrub SASBP value =" $SCRUB_SASBP

if [ $SCRUB_RAID == true ]
then
        ScrubRAIDFlag=0
        echo "RAID will be scrubed"

elif [ $SCRUB_RAID == false ]
then
        ScrubRAIDFlag=1
        echo "RAID will not be scrubbed"
else
        echo "Invalid argument passed for SCRUB_RAID. Cannot
continue ahead!!"
        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB CONFIG VALUES" "Invalid argument passed for SCRUB_RAID. Cannot continue ahead!!"
$BackupSource/scrublog
        exit -1
fi
        echo "Scrub RAID value =" $SCRUB_RAID

if [ $SCRUB_HDD == true ]
then
        ScrubHDDFlag=0
        echo "HDD will be scrubed"

elif [ $SCRUB_HDD == false ]
then
        ScrubHDDFlag=1
        echo "HDD will not be scrubbed"
else
        echo "Invalid argument passed for SCRUB_HDD. Cannot
continue ahead!!"
        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB CONFIG VALUES" "Invalid argument passed for SCRUB_HDD. Cannot continue ahead!!"
$BackupSource/scrublog
```

```
                exit -1
        fi
                echo "Scrub HDD value =" $SCRUB_HDD



        if [ $SCRUB_LOM == true ]
        then
                ScrubLOMFlag=0
                echo "LOM will be scrubed"

        elif [ $SCRUB_LOM == false ]
        then
                ScrubLOMFlag=1
                echo "LOM will not be scrubbed"
        else
                echo "Invalid argument passed for SCRUB_LOM. Cannot
continue ahead!!"
                sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB CONFIG VALUES" "Invalid argument passed for SCRUB_LOM. Cannot continue ahead!!"
$BackupSource/scrublog
                exit -1
        fi
                echo "Scrub LOM value =" $SCRUB_LOM

        if [ $SCRUB_FRU == true ]
        then
                ScrubFRUFlag=0
                echo "FRU will be scrubed"

        elif [ $SCRUB_FRU == false ]
        then
                ScrubFRUFlag=1
                echo "FRU will not be scrubbed"
        else
                echo "Invalid argument passed for SCRUB_FRU. Cannot
continue ahead!!"
                sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB CONFIG VALUES" "Invalid argument passed for SCRUB_FRU. Cannot continue ahead!!"
$BackupSource/scrublog
                exit -1
        fi
                echo "Scrub FRU value =" $SCRUB_FRU

        if [ $SCRUB_BIOS_iDRAC == true ]
        then
                ScrubBIOSiDRACFlag=0
                echo "BIOS_iDRAC will be scrubed"
```

```
                                elif [ $SCRUB_BIOS_iDRAC == false ]
                                then
                                        ScrubBIOSiDRACFlag=1
                                        echo "BIOS will not be scrubbed"
                                else
                                        echo "Invalid argument passed for SCRUB_BIOS_iDRAC.
Cannot continue ahead!!"
                                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB CONFIG VALUES" "Invalid argument passed for SCRUB_BIOS_iDRAC. Cannot continue
ahead!!" $BackupSource/scrublog
                                        exit -1
                                fi
                                        echo "Scrub BIOS value =" $SCRUB_BIOS_iDRAC



                        else
                                        echo "Scrub file not found!! Creating One with defaults at
$BackupSource"
#                                       echo "SCRUB_BMC=true" > $BackupSource/scrubconfig.txt
                                        echo "SCRUB_PSU=true" >> $BackupSource/scrubconfig.txt
                                        echo "SCRUB_SASBP=true" >> $BackupSource/scrubconfig.txt
                                        echo "SCRUB_RAID=true" >> $BackupSource/scrubconfig.txt
                                        echo "SCRUB_HDD=true" >> $BackupSource/scrubconfig.txt
#                                       echo "SCRUB_DVD=true" >> $BackupSource/scrubconfig.txt
                                        echo "SCRUB_LOM=true" >> $BackupSource/scrubconfig.txt

                                        echo "SCRUB_FRU=true" >> $BackupSource/scrubconfig.txt

                                        echo "SCRUB_BIOS_iDRAC=true" >>
$BackupSource/scrubconfig.txt



                        fi
                else
                        echo "System match failed"
                        echo "System Backup not found"

                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4` "SYSTEM
BACKUP" "System Backup could not be located. Cannot Continue." $nfs_mount/tracelog
                        exit -1
                fi

                arch=`getSystemId | grep PowerEdge | cut -d ' ' -f 4`
                echo "Underlying platform is "$arch
                touch /tmp/Scrubflag
                ScrubFlag=/tmp/Scrubflag
#               echo "ScrubBMCFlag="$ScrubBMCFlag > $ScrubFlag
                echo "ScrubPSUFlag="$ScrubPSUFlag >> $ScrubFlag
```

```
                echo "ScrubSASBPFlag="$ScrubSASBPFlag >> $ScrubFlag
                echo "ScrubRAIDFlag="$ScrubRAIDFlag >> $ScrubFlag
                echo "ScrubHDDFlag="$ScrubHDDFlag >> $ScrubFlag
#               echo "ScrubDVDFlag="$ScrubDVDFlag >> $ScrubFlag
                echo "ScrubLOMFlag="$ScrubLOMFlag >> $ScrubFlag
                echo "ScrubFRUFlag="$ScrubFRUFlag >> $ScrubFlag
                echo "ScrubBIOSiDRACFlag="$ScrubBIOSiDRACFlag >> $ScrubFlag


                case "$arch" in
                        R310)

                                echo "Restoration for R310 would be used"
                                target_arch=$nfs_mount/DUPrepo/R310

                                BackupSource=$nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG

                                restore_data_R310
                                                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut
-d ' ' -f 4` "SCRUB ACTION" "Scrub Process completed successfully" $nfs_mount/tracelog
                        ;;

                        R410)
#                               echo "Identified Architecture as R410"

                                echo "Restoration for R410 would be used"
                                target_arch=$nfs_mount/DUPrepo/R410

                                BackupSource=$nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG

                                restore_data_R410
                                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB ACTION" "Scrub Process completed successfully" $nfs_mount/tracelog
                        ;;

                        R610)
#                               echo "Identified Architecture as R610"

                                echo "Restoration for R610 would be used"
                                target_arch=$nfs_mount/DUPrepo/R610

                                BackupSource=$nfs_mount/PEbackup/SVCTAG_$SERVICE_TAG
#                               sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"PLATFORM IDENTIFICATION" "Identified Platform as R610." $BackupSource/scrublog
                                restore_data_R610
                                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SCRUB ACTION" "Scrub Process completed successfully" $nfs_mount/tracelog
                        ;;
```

```
                *)
                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"PLATFORM IDENTIFICATION" "Identified Platform not supported." $nfs_mount/tracelog
                        echo "Error! Current platform is not supported by the utility !!"
                        exit -1
                ;;
        esac
;;


verify)
        echo "Verification Process Started"
        sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4` "VERIFICATION
PROCESS STARTED" "Start of verification process" $nfs_mount/tracelog
        ls /mnt/NFSShare/PEbackup/ | grep SVCTAG > /tmp/systems

        count=0
        match_result=0
        for readline in `cat /tmp/systems`
        do
#               echo hello $readline
                backup_path=/mnt/NFSShare/PEbackup/$readline/
                if [ -f /mnt/NFSShare/PEbackup/$readline/sysinfo.txt ]
                then
                        TMPCN_NUM=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt |
grep MB_SN | cut -d '=' -f 2`
                        target_platform=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt |
grep Platform | cut -d '=' -f 2`
                        target_svctag=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt |
grep Service_Tag | cut -d '=' -f 2`
                        target_assettag=`cat /mnt/NFSShare/PEbackup/$readline/sysinfo.txt |
grep Asset_Tag | cut -d '=' -f 2`
                        count=`expr $count + 1`
                #       echo "count = $count "

                        SERVICE_TAG=`getSystemId | grep Tag: | cut -d ' ' -f 4`
                        ASSET_TAG=`assetTag | grep Tag: | cut -d ' ' -f 4`
                        target_ID=`dmidecode | grep ..CN | cut -d ' ' -f 3`
                else
                        echo "Cannot locate sysinfo.txt for `getSystemId | grep Tag: | cut -d ' ' -
f 4`"
                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"VERIFICATION PROCESS FAILED" "Cannot locate sysinfo.txt file" $nfs_mount/tracelog
                        exit -1
                fi

                if [ $TMPCN_NUM == $target_ID ]
                then
                        echo "System IDENTIFIER Verified"
```

```
                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SYSTEM BACKUP IDENTIFICATION" "System found in the backup" $nfs_mount/tracelog

                match_result=1
                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SYSTEM VERIFICATION STARTED" "System verification started" $backup_path/verifylog

                if [ $SERVICE_TAG == $target_svctag ]
                then
                        echo "Service tag Matched !!"
                        echo "Platform = "$target_platform
                        echo "service tag = "$target_svctag
                        echo "Stored Backup Path = "$backup_path
                        sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SERVICE TAG IDENTIFICATION" "Service Tag information matches" $backup_path/verifylog
                else
                        echo "Service tag Mismatch !!"
                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SERVICE TAG IDENTIFICATION" "Service Tag information does not matches"
$backup_path/verifylog
                fi

                if [ $ASSET_TAG == $target_assettag ]
                then
                        echo "Asset tag Match found!!"
                        sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"ASSET TAG IDENTIFICATION" "Asset Tag information matches" $backup_path/verifylog
                else
                        echo "Asset Tag Mismatch"
                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"ASSET TAG IDENTIFICATION" "Asset Tag information does not matches" $backup_path/verifylog
                fi


                sh ./LOMVerify.sh $backup_path

                sh ./FRUVerify.sh $backup_path

                sh ./verify.sh $nfs_mount $backup_path $target_platform BIOS

                sh ./verify.sh $nfs_mount $backup_path $target_platform SAS

                sh ./verify.sh $nfs_mount $backup_path $target_platform LOM


                sh ./verify.sh $nfs_mount $backup_path $target_platform RAID

                sh ./verify.sh $nfs_mount $backup_path $target_platform HDD
```

```
                                sh ./verify.sh $nfs_mount $backup_path $target_platform iDRAC

                                sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"SYSTEM VERIFICATION COMPLETE" "System verification completed successfully"
$backup_path/verifylog
                        else

                                if [ $match_result == 1 ]
                                then
                                        match_result=1

                                else if [ $match_result == 0 ]
                                then
                                        echo "iterating"
                                    fi
                                fi
                        fi
                done

                                if [ $match_result == 0 ]
                                then
                                        echo "System backup not found!!"
                                        echo "Please make sure the backup folder
SVCTAG_$SERVICE_TAG is intact"
                                        sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4`
"VERIFICATION PROCESS FAILED" "Platform data could not be located in backup "
$nfs_mount/tracelog
                                        exit -1
                                else
                                        echo "System found. Verification successful"
                                        sh ./logging_script.sh 0 `getSystemId | grep Tag | cut -d ' ' -f 4`
"VERIFICATION PROCESS SUCCESSFUL" "Platform verification process successful. "
$nfs_mount/tracelog
                                fi


        ;;

        *)
                echo "Error!! Wrong Action specified in config.txt"
                cat $nfs_mount/config.txt
                sh ./logging_script.sh 1 `getSystemId | grep Tag | cut -d ' ' -f 4` "ACTION CHECK"
"Wrong Action specified in config.txt" $nfs_mount/tracelog
                exit 1
        ;;
        esac
```

Appendix D – Sample Dell R610 Back Up Log

<?xml version="1.0" encoding="UTF-8"?>

<Element lang="**en**" schemaVersion="**1.0**">

<Element timestamp="**2011-06-23:144630**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP PROCESS**">

<Message> Backup process started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144631**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP LOM DATA**">

<Message> LOM Backup started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144637**" status="**success**" service_tag="**G2JZLN1**" name="**MAC ADDRESS BACKUP**">

<Message> MAC address backup successful </Message>

</Element>

<Element timestamp="**2011-06-23:144653**" status="**success**" service_tag="**G2JZLN1**" name="**ETH DATA BACKUP**">

<Message> ETH Data backup successful </Message>

</Element>

<Element timestamp="**2011-06-23:144654**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP LOM DATA**">

<Message> LOM Backup completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144654**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP FRU DATA**">

<Message> FRU Backup started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144655**" status="**success**" service_tag="**G2JZLN1**" name="**FRU BACKUP START**">

<Message> FRU backup start successful </Message>

</Element>

<Element timestamp="**2011-06-23:144659**" status="**success**" service_tag="**G2JZLN1**" name="**FRU 0 BACKUP**">

<Message> FRU 0 backup successful </Message>

</Element>

<Element timestamp="**2011-06-23:144700**" status="**success**" service_tag="**G2JZLN1**" name="**FRU 2 BACKUP**">

<Message> FRU 2 backup successful </Message>

</Element>

<Element timestamp="**2011-06-23:144701**" status="**success**" service_tag="**G2JZLN1**" name="**FRU 3 BACKUP**">

<Message> FRU 3 backup successful </Message>

</Element>

<Element timestamp="**2011-06-23:144702**" status="**success**" service_tag="**G2JZLN1**" name="**FRU 5 BACKUP**">

<Message> FRU 5 backup successful </Message>

</Element><Element timestamp="**2011-06-23:144703**" status="**success**" service_tag="**G2JZLN1**" name="**FRU 4 BACKUP**">

<Message> FRU 4 backup successful </Message>

</Element>

<Element timestamp="**2011-06-23:144704**" status="**success**" service_tag="**G2JZLN1**" name="**FRU BACKUP COMPLETE**">

<Message> FRU backup completed </Message>

</Element>

<Element timestamp="**2011-06-23:144705**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP FRU DATA**">

<Message> FRU Backup completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144705**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP LC DATA**">

<Message> LC Backup started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144732**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP LC IMAGE**"><

Message> LC Backup of MAS001.img completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144737**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP LC IMAGE**">

<Message> LC Backup of MAS002.img completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144739**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP LC DATA**">

<Message> LC Backup completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144740**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP CMOS SETTINGS**">

<Message> CMOS Backup started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144746**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP CMOS SETTINGS**">

<Message> CMOS Backup completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:144747**" status="**success**" service_tag="**G2JZLN1**" name="**BACKUP PROCESS**">

<Message> Backup process completed successfully </Message>

</Element>

</Element>

Appendix E – Sample Dell R610 Sanitization Log

<?xml version="1.0" encoding="UTF-8"?>
<Element lang="**en**" schemaVersion="**1.0**">

<Element timestamp="**2011-06-23:145520**" status="**success**" service_tag="**G2JZLN1**"
name="**SCRUB PHASE**">

<Message> First Phase Started </Message>

</Element><Element timestamp="**2011-06-23:145756**" status="**success**"
service_tag="**G2JZLN1**" name="**SAS BACKPLANE SCRUB SUCCESS**">

<Message> SAS Backplane Scrub was successful </Message>

</Element>

<Element timestamp="**2011-06-23:145756**" status="**success**" service_tag="**G2JZLN1**"
name="**STORAGE SCRUB**">

<Message> Storage Scrub started successfully </Message>

</Element><Element timestamp="**2011-06-23:145757**" status="**success**"
service_tag="**G2JZLN1**" name="**RAID CONFIG DELETION**">

<Message> Existing RAID Config Deleted Successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145800**" status="**success**" service_tag="**G2JZLN1**"
name="**SECURITY KEY DELETION**">

<Message> Security Key Deleted Successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145807**" status="**success**" service_tag="**G2JZLN1**"
name="**RAID FIRMWARE**">

<Message> RAID Firmware sanitized Successfully </Message>

</Element><Element timestamp="**2011-06-23:145811**" status="**success**"
service_tag="**G2JZLN1**" name="**RAID ADAPTER DEFAULTS**">

<Message> RAID Controller successfully set to factory defaults </Message>

</Element>

<Element timestamp="**2011-06-23:145816**" status="**success**" service_tag="**G2JZLN1**" name="**RAID-0 CONFIGURATION**">

<Message> RAID Controller successful in creating RAID array </Message>

</Element>

<Element timestamp="**2011-06-23:145820**" status="**success**" service_tag="**G2JZLN1**" name="**STORAGE SCRUB**">

<Message> Storage Scrub completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145825**" status="**success**" service_tag="**G2JZLN1**" name="**HDD SCRUB SUCCESS**">

<Message> HDD Scrub was successful </Message>

</Element><Element timestamp="**2011-06-23:145826**" status="**success**" service_tag="**G2JZLN1**" name="**LOM SCRUB**">

<Message> LOM Scrub started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145826**" status="**success**" service_tag="**G2JZLN1**" name="**LOM DATA RESTORE**">

<Message> LOM data restoration started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145901**" status="**success**" service_tag="**G2JZLN1**" name="**LOM DATA RESTORE**">

<Message> LOM data restoration completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145918**" status="**success**" service_tag="**G2JZLN1**" name="**LOM FIRMWARE**">

<Message> LOM firmware sanitized successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145924**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth0 updated successfully </Message>

</Element><Element timestamp="**2011-06-23:145930**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth1 updated successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145936**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth2 updated successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145942**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth3 updated successfully </Message>

</Element>

<Element timestamp="**2011-06-23:145948**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth4 updated successfully </Message>

/Element>

<Element timestamp="**2011-06-23:145954**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth5 updated successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150000**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth6 updated successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150006**" status="**success**" service_tag="**G2JZLN1**" name="**ETHERNET MAC**">

<Message> MAC address for eth7 updated successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150007**" status="**success**" service_tag="**G2JZLN1**" name="**LOM SCRUB**">

<Message> LOM Scrub completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150007**" status="**success**" service_tag="**G2JZLN1**" name="**FRU SCRUB**">

<Message> FRU Scrub started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150008**" status="**success**" service_tag="**G2JZLN1**" name="**FRU DATA RESTORE**">

<Message> FRU data restoration started successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150008**" status="**success**" service_tag="**G2JZLN1**" name="**FRU INDEX LOCATED**">

<Message> FRU index file located </Message>

</Element>

<Element timestamp="**2011-06-23:150017**" status="**success**" service_tag="**G2JZLN1**" name="**FRU ID 0 RESTORED**">

<Message> FRU ID 0 file located and restored successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150017**" status="**success**" service_tag="**G2JZLN1**" name="**FRU ID 2 RESTORED**">

<Message> FRU ID 2 file located and restored successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150018**" status="**success**" service_tag="**G2JZLN1**" name="**FRU ID 3 RESTORED**">

<Message> FRU ID 3 file located and restored successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150019**" status="**success**" service_tag="**G2JZLN1**" name="**FRU ID 5 RESTORED**">

<Message> FRU ID 5 file located and restored successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150021**" status="**success**" service_tag="**G2JZLN1**" name="**FRU ID 4 RESTORED**">

<Message> FRU ID 4 file located and restored successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150021**" status="**success**" service_tag="**G2JZLN1**" name="**FRU DATA RESTORE COMPLETE**">

<Message> FRU data restoration successful </Message>

</Element>

<Element timestamp="**2011-06-23:150022**" status="**success**" service_tag="**G2JZLN1**" name="**FRU SCRUB**">

<Message> FRU Scrub completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:150022**" status="**success**" service_tag="**G2JZLN1**" name="**CUSTOM iDRAC**">

<Message> Custom iDRAC update started </Message>

</Element>

<Element timestamp="**2011-06-23:150846**" status="**success**" service_tag="**G2JZLN1**" name="**CUSTOM iDRAC**">

<Message> Custom iDRAC update successful </Message>

</Element>

<Element timestamp="**2011-06-23:150846**" status="**success**" service_tag="**G2JZLN1**" name="**CUSTOM iDRAC**">

<Message> SCRUB iDRAC NVRAM started </Message>

</Element>

<Element timestamp="**2011-06-23:151832**" status="**success**" service_tag="**G2JZLN1**" name="**RESTORE LC IMAGE**">

<Message> LC RESTORE of MAS001.img completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:151858**" status="**success**" service_tag="**G2JZLN1**" name="**RESTORE LC IMAGE**">

<Message> LC RESTORE of MAS002.img completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:151947**" status="**success**" service_tag="**G2JZLN1**" name="**CUSTOM BIOS**">

<Message> Custom BIOS update successful </Message>

</Element>

<Element timestamp="**2011-06-23:152900**" status="**success**" service_tag="**G2JZLN1**" name="**PSU UPDATE**">

<Message> Successfully Updated Power Supply Unit #1 </Message>

</Element>

<Element timestamp="**2011-06-23:153558**" status="**success**" service_tag="**G2JZLN1**" name="**SCRUB PHASE**">

<Message> First Phase Completed Successfully </Message>

</Element>

<Element timestamp="**2011-06-23:153918**" status="**success**" service_tag="**G2JZLN1**" name="**SCRUB PHASE**">

<Message> Second Phase Started </Message>

</Element>

<Element timestamp="**2011-06-23:153919**" status="**success**" service_tag="**G2JZLN1**"
name="**CUSTOM BIOS UPDATE SUCCESS**">

<Message> Custom BIOS Update was successful </Message>

</Element>

<Element timestamp="**2011-06-23:153919**" status="**success**" service_tag="**G2JZLN1**"
name="**STOCK BIOS**">

<Message> Stock BIOS update started </Message>

</Element>

<Element timestamp="**2011-06-23:153927**" status="**success**" service_tag="**G2JZLN1**"
name="**STOCK BIOS**">

<Message> Stock BIOS update successful </Message>

</Element>

<Element timestamp="**2011-06-23:153928**" status="**success**" service_tag="**G2JZLN1**"
name="**STOCK iDRAC**">

<Message> Stock iDRAC update started </Message>

</Element>

<Element timestamp="**2011-06-23:154807**" status="**success**" service_tag="**G2JZLN1**"
name="**CUSTOM iDRAC**">

<Message> Custom iDRAC enhanced scrubbing successful </Message>

</Element>

<Element timestamp="**2011-06-23:155218**" status="**success**" service_tag="**G2JZLN1**"
name="**STOCK iDRAC**">

<Message> Stock iDRAC update successful </Message>

</Element>

<Element timestamp="**2011-06-23:155219**" status="**success**" service_tag="**G2JZLN1**"
name="**SCRUB PHASE**">

<Message> Second Phase completed successfully </Message>

</Element>

<Element timestamp="**2011-06-23:155631**" status="**success**" service_tag="**G2JZLN1**" name="**SCRUB PHASE**">

<Message> Third Phase Started </Message>

</Element>

<Element timestamp="**2011-06-23:155649**" status="**success**" service_tag="**G2JZLN1**" name="**CMOS CONFIG RESTORE**">

<Message> CMOS Setting restore successful </Message>

</Element>

<Element timestamp="**2011-06-23:155649**" status="**success**" service_tag="**G2JZLN1**" name="**RAID INIT**"><

Message> RAID initialization successful </Message>

</Element>

<Element timestamp="**2011-06-23:155650**" status="**success**" service_tag="**G2JZLN1**" name="**RAID SECURITY KEY**">

<Message> RAID Array Security key Jhuapl*123 successful </Message>

</Element>

<Element timestamp="**2011-06-23:155652**" status="**success**" service_tag="**G2JZLN1**" name="**RAID SECURE**">

<Message> RAID Array Secure successful </Message>

</Element>

<Element timestamp="**2011-06-23:155652**" status="**success**" service_tag="**G2JZLN1**" name="**SCRUB PHASE**">

<Message> Third Phase completed successfully </Message>

</Element>