Global Information Assurance Certification Paper

Detachable Data Compartmentalization: Layered Defense for Laptop Data Using USB Keychain Hard Drives as Detachable Data Compartmentalization Modules

USB Keychain Hard Drive System Integration with Laptops Running OpenBSD, Linux or Windows 2000

GIAC Security Essentials Certification [GSEC]
Practical Assignment Version 1.4b – Option 1
John R. Pritchard
24 December 2003

Table of Contents

Abstract

This paper introduces the concept of Detachable Data Compartmentalization, or DDC, and how it can be used with mobile computing systems to provide a defense-in-depth layered approach to data security. DDC is a process that uses external media to compartmentalize sensitive data as well as formalize a security model for using and storing information. This places the security emphasis and value focus on digital data. I outline a prioritized, layered defense strategy that is made up of: 1) educated, security-aware computer users, 2) hardened, reproducible laptop configurations, 3) Detachable Data Compartmentalization Modules [DDCMs], 4) partition-level or disk-level DDCM encryption, and 5) token-based authentication and/or cryptography. The technical focus of this paper is to describe system integration steps for using USB keychain hard drives as DDCMs on laptops running OpenBSD, Linux or Windows2000. These small, inexpensive mass storage devices can be used as DDCMs to implement a DDC layered defense strategy for increased laptop security.

Technical Overview

The techniques described in this paper were primarily developed and tested using a single computer system and one USB keychain hard drive. While the findings of this paper should generally apply to other systems and hardware, your mileage may vary. The operating systems [OS] used for this paper where chosen because of the familiarity to the author, applicability to building secure systems, and because they covered a broad range of OS environments that could benefit from the Detachable Data Compartmentalization security model.

Keywords

best practices, compartmentalization, cryptographic file systems, CFS, data compartmentalization, data integrity, data modules, data protection, data security, data theft, DDC, DDCM, defense in depth, defense-in-depth, DELL, detachable data compartment, detachable data compartmentalization, detachable data compartmentalization module, digital assets, encryption, flash disk, flash drive, GIAC, GSEC, infrastructure architecture, infrastructure engineering, information security, InfoSec, INFOSEC, JumpDrive, keychain, keychain hard drive, laptop, laptop security, layered security, Linux, memory fob, mobile computer, mobile computing, mobile computing data, mobile devices, mobile security, notebook, notebook security, obfuscation, OpenBSD, physical security, portable hard drive, portable hard drive, private key, RedHat, removable hard drive, removable media, SANS Institute, steganography, system integration, ThumbDrive, token-based authentication, travel security, USB, USB fob, USB hard drive, USB JumpDrive, USB keychain, USB mass storage, USB pendisk, USB token, USB ThumbDrive, Win2000, Win2K, Windows 2000

Laptop Configuration Details

A DELL Latitude 800c Laptop was configured as a multi-boot system running either OpenBSD 3.3 (OpenBSD 2003), RedHat 8.0 Linux (RedHat 2003), or Windows 2000 Professional Workstation (Microsoft 2003).

The DELL Latitude 800c Laptop had a 20GB hard-drive, 512MB RAM, an ATI 128 Rage video card, and a Pentium III processor.

USB Keychain Hard Drives

For this article, I used a Memorex “ThumbDrive 256MB” USB keychain hard drive (ThumbDrives 2003). This same drive was used successfully under OpenBSD 3.3, RedHat 8.0 Linux, and Windows 2000 Professional Workstation operating systems. Additional, less extensive testing was performed with a Lexar Media 128MB “JumpDrive Secure” USB portable flash drive (JumpDrive 2003). The Lexar Media JumpDrive was used succesfully under RedHat 8.0 and Windows 2000, but was not recognized under OpenBSD 3.3. Each of these USB keychain hard drives were purchased for less than $100USD.

Typographic Conventions

I developed a number of typographic conventions for this article to aid readability. The names for these format styles where chosen to be consistent with the Simplified DocBook (DocBook 2003) Document Type Definition [DTD] technical documentation language. Examples of these conventions and their usage can be seen below.

ProgramListing

The **ProgramListing** paragraph format is used to highlight input that might be typed at a command prompt. This format includes non-typed data such as cursor and directory listings that might be also be seen. The following is an example of a **ProgramListing** under OpenBSD:

```
$ su
Password:
# cd
# umount /mnt/keychain-temp
# rmdir /mnt/keychain-temp
```

ComputerOutput

The **ComputerOutput** paragraph format is used for source code examples as

well as output from commands entered at the command prompt.

```
# $Id: usb_flash_drives.xml,v 1.2 2003/08/23 15:08:52 john Exp $
# Sample auto.master file
# Format of this file:
# mountpoint map options
# For details of the format look at autofs(8).
/mnt/keychain /etc/auto.usbfob --timeout=10
```

Example

The **Example** paragraph format is used to highlight quotes or examples that are offset from the main text.

> This is an example of the the **Example** paragraph format.

Command

The **Command** character format is used to identify commands within this article's text [rather than providing a separate ProgramListing]. The **Command** format is also used to highlight points of interest.

> The first step to using your USB Keychain, flash drive, or other mobile storage device is to **mount** it for your operating system.

Filename

The **Filename** character format is used to denote both filenames, users and groups. The following sentence uses the **Filename** and **Command** conventions.

> BSD systems do not allow you to simply modify your ***`/etc/fstab`*** entry to allow non-***`root`*** users the ability to **mount** and **umount** devices

Introduction

The concepts presented in this paper, while written to describe a novel approach to layered defense for mobile laptop users, can also be extended to other mobile computing devices as well as dedicated workstations or servers.

Laptops by their nature are mobile, portable devices that leave the potential safety and security of controlled office environments. Outside of the office, the threat and vulnerability to laptops increase significantly because of dangers such

as theft, damage, or exposure to hostile computing environments. There is significant documentation describing the threats mobile computing users face, and strategies for minimizing this risk (Ryder 2001a; Ryder 2001b; JHSPH 2003; Mueller 2001; Palmer 2001; Grant 2003).

Unfortunately, the cost of this risk is often poorly understood. In the event of laptop theft, many companies focus primarily on the cost of the computer hardware and fail to recognize or quantify the cost of the loss of data. This hardware-centric viewpoint is evident in the numerous products, services, and companies that are designed to physically secure equipment, alert movement of hardware, and/or track and recover lost or stolen devices. While such physical security tools have a role in information security, they emphasize the importance of physical things rather than the value of proprietary information or sensitive data.

The Insidious Nature of Data Loss

Data loss is insidious and difficult to quantify. On one level, there is the inconvenience of having to restore a functioning computer system. This may involve the acquisition of new hardware and software, installation of software, recovery of data from backups, configuration of the system, etc.... While a strong backup and disaster recovery policies and procedures can minimize any disruptions and loss of employee productivity (Gallagher 2001; Nemeth et. al. 2001; Parrish 2001; Cole et. al. 2003a; Cole et. al. 2003b), it is still a possibility that some information will be lost either because it was omitted in the backup-recovery process [in which case the backup-recovery process needs improvement] or the data had not been backed up immediately prior to the theft.

In cases of data loss or data theft, there are also potential costs associated with repeating the work, lost credibility, or consequences of losing irreplaceable data. For example, the cost in credibility and liability exposure to both an employee and their company for “losing” sensitive information could be significant.

Kevin Mitnick (Mitnick and Simon 2003) provides numerous examples of targeted data theft and the use of stolen data by competing companies in his recent book on social engineering. A review of the past few months of email newsletters from the System Administration, Audit, Networking, and Security [SANS] Institute also reveal numerous examples of data theft (SANS NewsBites 5/40; SANS NewsBites 5/43; SANS NewsBites 5/45; SANS NewsBites 5/47; SANS NewsBites 5/48a; SANS NewsBites 5/48b; SANS NewsBites 5/51). Descriptions of laptop theft and methods to protect against laptop theft are also quite common (Hiebert 1997; JHSPH 2003; Mueller 2001; Palmer 2001; Kay 2001; Ryder 2001a; Ryder 2001b; BUPD 2002; Artner 2002; Grant 2003).

Quantifying the cost of data theft can be challenging, and it may be most

productive to develop a range that describes the potential impact. On one end of the spectrum, the cost would be associated with a “hope for the best” scenario - where there is no perceived value for our stolen data and the thieves effectively destroy or permanently “lose” our information. On the other end of the spectrum, the cost would be associated with a “plan for the worst” scenario – where all of our confidential information is in the hands of our competitors or posted publicly, our credibility with our clients and vendors is ruined, we lose numerous future contracts, and face extensive litigation.

Clearly, when the “worst case scenario” is envisioned, the total cost to a company could “make or break” an organization, and the costs could exceed several hundred thousand or several million dollars. Such multi-million dollar estimates are not unreasonable when one looks at examples where companies are penalized for using proprietary information from their competitors. Boeing Aerospace is currently embroiled amidst allegations of inappropriately using proprietary information [belonging to one of its competitors]. “In response to Boeing's action, the Air Force stripped the company of $1 billion in government business this summer and suspended its space unit from competing for new contracts” (Merle 2003). While this example does not appear to have involved “data theft”, but rather, newer Boeing employees “sharing” their previous employer's proprietary information, it does illustrate that significant “cost” can be associated with both losing your own or using your competitor's confidential information. It is important to keep such multi-million dollar costs in mind when we evaluate the overall risk for laptop and data theft, and the lengths to which thieves and competing companies might go to obtain such information.

It is not safe to assume that protection such as laptop BIOS passwords, strong authentication methods, laptop tracking and monitoring devices, or even disk and file encryption technology will safeguard your data if it is stolen. We should adopt a mindset that if information is stolen, it is merely a matter of time and effort before it becomes transparent to the data thieves.

Detachable Data Compartmentalization or DDC

Given the significant threats and vulnerabilities faced by our mobile computing devices and users, we need to develop innovative, cost-effective approaches to minimize our overall risk level and safeguard our mobile computing data.

I have developed a concept called **Detachable Data Compartmentalization**, or **DDC**, to address this issue and provide a layered defensive strategy for protecting mobile computer users and their data. In simple terms, Detachable Data Compartmentalization is a twist on using removable media to protect sensitive information, and users who already regularly backup critical data to USB Keychains, ZIP disks, CD-ROMs, or other external media will find the concept fairly straightforward. But, as you will see, DDC goes beyond just simply

using external media and presents a security model for using and storing information that places the security emphasis and value focus on the digital data.

The “compartmentalization” concept was developed in part by my own need to find a way to work securely in potentially hostile computing environments. Reading and thinking about the U.S. Intelligence Community's “Protecting Sensitive Compartmented Information Within Information Systems: Policy” (DCID 6/3) also influenced the growth of this idea.

However, the bulk of my inspiration for DDC model of layered security came from the ideas of Steve Traugott and his peers (Traugott et. al. 2003a; Traugott and Huddleston 1998) on infrastructure architecture, infrastructure engineering, configuration management, and automated systems administration.

I would argue that a philosophical paradigm shift will be necessary for our society to effectively transition into a group of educated, security-aware, computer users. Computers and most computing devices need to be viewed simply as hardware that is used to provide the muscle and applications for manipulating and presenting our data - whether that data be word processing documents, spreadsheets, images, video, music, presentations, databases, source code, special applications, or anything else. In this respect, while computers are continually patched and hardened to provide a secure computing platform, if all sensitive laptop data can be compartmentalized and stored using the DDC model (or, alternatively - a centralized, network-based data distribution system), the computer hardware itself becomes easily replaceable and essentially a disposable commodity. Steve Traugott and his team describe this mindset in how they assess their success in meeting disaster recovery goals in a networked computing environment (Traugott et. al. 2003b):

> The test we use when designing infrastructures is "Can I grab a random machine that's never been backed up and throw it out the tenth-floor window without losing sysadmin work?" If the answer to this was "yes", then we knew we were doing things right.

Our Detachable Data Compartmentalization model can be thought of in an analogous manner. However, in the case of DDC, our acid test would be:

> “Can a DDC user have their laptop stolen or destroyed without losing or exposing any critical or sensitive information? Do we have a reliable, reproducible method to quickly deploy a newly configured laptop system for this user to use with their DDCM?” If the answers are “yes”, then we have effectively implemented a DDC model.

Effective Detachable Data Compartmentalization requires: 1) the identification of sensitive information, 2) the identification of any data-dependency requirements, and 3) isolation of this information in a removable or detachable “compartment”.

Detachable Data Compartmentalization Modules or DDCMs

This external storage device, where mobile computer users “compartmentalize” their sensitive data, will be called a **Detachable Data Compartmentalization Module** or **DDCM**.

Small USB Keychain hard drives (also known as Memory Fobs, FlashDrives, JumpDrives, PenDisks, ThumbDrives and numerous other terms) make an ideal compartment for implementing DDC. Such USB Keychain hard drives can easily be purchased for under $100USD. While use of DDC requires additional steps on computer systems running OpenBSD or Linux, these configuration steps are relatively straightforward. Use of most USB Keychain hard drives on Windows2000 and XP systems can be achieved without any additional drivers or configuration steps.

In this paper, I present system integration methods for USB keychain mobile mass storage devices to be used as DDCMs on laptops running either OpenBSD, Linux, or Windows2000. Recent advances in these small, cheap, 32mb–1.5GB ($30 - $200USD) USB Keychain hard drives have created an opportunity to acquire DDCMs and implement a simple DDC security model as a low-cost, simple, and effective method to add robust, “defense-in-depth” layers of protection to mobile computing data.

Once a user is accustomed to the DDC model, a user's focus will be on “their data” and knowing “that system”. The emphasis will be on the DDCM (core data compartmentalized in a small, portable, and easily hidden and protected storage device) and how it interacts with their hardened laptop computing system in the context of a DDC model.

A key benefit of using a DDC model for protecting information is that because user-specific, critical information is compartmentalized, backup procedures for this data are radically simplified and transparent to the end user. To backup the data, one simply copies the contents of their DDC Module. Since most USB keychain hard drives are under the 700mb CD-R limit, a single CD could hold several full DDC backups. The small size of USB Keychain hard drives, relative to the ubiquitous 20-120GB hard drives, also promotes a mindset and awareness among computer users that “less is more”, and encourages good security practices such as logical organization of data, version control, and routine archiving of extraneous or legacy data. I would recommend new DDC users start with smaller DDCMs (on the order of 32mb to 128mb), and increase

the size of the DDCM (and probably also the complexity of their DDC model) as they gain experience and understanding of this unique method to protect mobile data.

The small, contained nature of DDC also makes it easier for users to focus on the “know your system” (Cole et. al. 2003a; Cole et. al. 2003b) concept of security awareness, and the responsibility for this data security becomes an easily graspable and attainable goal for mobile computer users. Responsibility for the computing platform, while shared with the user, lies primarily with IT staff - especially with respect to setup, configuration, hardening and regular patching of the system. Moreover, use of the DDC model also allows IT staff to decouple the process of backing up data from the process of backing up systems. In this respect, we create an environment where automated configuration management solutions could be used to deploy and manage generic computing systems that are well tested, patched, hardened and understood.

In the following paper, I focus on the DDC model and using DDCMs as an innovative approach to add layers of defense for laptop data. The technical focus will be on how USB Keychains can be used as DDCMs on laptops running either OpenBSD, Linux or Windows. Thus, rather than relying on technology-intensive encryption algorithms, hardware-focused tracking devices or motion-sensor alarms, a user protects her [or his] data by compartmentalizing it and practicing strong physical security of the DDCM. While the laptop computer remains an important component in the overall DDC security model, it is limited to the roll of simply providing a generic computing system with the processing power, memory, and applications to access, manipulate and present the DDC stored data. In this respect, if a laptop is or destroyed, it should merely be a sanitized piece of hardware where the total cost is limited to cost of the laptop hardware and time required to set up a new generic, hardened, computing system. Thus, data availability may be temporarily disrupted, but data confidentiality and data integrity remain intact.

The DDC Model of Defense-In-Depth for Laptops

As with any information security initiative, security is a process that must balance the trade-offs between usability and security. While I would argue that DDC simplifies the process of information security, and thereby makes it easier to increase both data and system security at the same time as improving usability, DDC does require a philosophical shift in how most users, information technology staff, and hardware and software vendors approach the concept of data storage and digital asset protection. It is my hope that the logic of DDC will be embraced by the information security community and its practice will become widespread.

As with any defense-in-depth security approach, we layer defenses to protect the most critical asset: our data. Security is an ongoing process that requires constant vigilance, and I assume that basic practices such as user education, security patches, firewalls, anti-virus updates and vulnerability assessments will be performed on a regular basis. DDC is simply another tool that is available to information security professionals and organizations to expand their arsenal of available defense-in-depth security options. While each organization will need to judge the merits of DDC with respect to its own needs, policies and procedures, I have outlined my thoughts and priorities for where DDC falls in developing a robust defense-in-depth strategy to protect mobile computer users and their data.

Prioritized, DDC Defensive Layers Strategy

(1) Educated, Security-Aware Users
(2) Hardened, Reproducible Laptop Configurations
(3) Detachable Data Compartmentalization Modules [DDCMs]
(4) Add Partition-Level or Disk-Level DDCM Encryption
(5) Add Token-Based Authentication and/or Cryptography

Layer 1: Educated, Security-Aware Users

Computer users need to be empowered with a basic understanding and appreciation of the information security threats and vulnerabilities that exist, and put them, their computer systems, and their digital data at risk. Businesses need to develop clear and logical security policies and procedures that their employees both understand and support. Security is a process and the single best defense in Information Security is an informed, educated, and security-aware workforce (Mitnick and Simon 2003; Gulati 2003).

These concepts are especially true for users of mobile computing devices where the risk for system exploitation, data loss, or data leakage can increase substantially. Staff need to understand, practice, and advocate security concepts such as firewalls, strong passwords, good physical security, risk awareness, anti-virus scanning, minimizing the number of services running, and other proactive security measures.

Our goal for education is to foster a culture of security-literate users and advocates that perpetuate themselves. While formal training, security conferences, policies and procedures, and other traditional methods have their place, I would recommend including a security education component that is run by the users themselves.

For instance, a weekly, bi-weekly or monthly “Security Lunch” where individual

staff suggest, present, and lead a discussion on a security-related topic. Topics could include reviewing a technical paper, a news story, a book or book chapter, a magazine article, or even a “story” about one of their own experiences as a user. By creating an environment where staff openly discuss and debate security issues, we build a workforce of cooperative, security-concious, informed staff.

I suggest two “must reads” for mobile computer users. The first is an excellent paper entitled “Defence in Depth on the Home Front” by Thomas Harbour (Harbour 2003). This paper provides a great summary of home network security, the defense-in-depth layered approach to security, securing systems, and potential risks faced by telecommuters.

The second “must read” is a recent book entitled “The Art of Deception: Controlling the Human Element of Security” (Mitnick and Simon 2003). This “pulp fiction”-styled book provides an entertaining read of how the “Human Element” of security is almost always an organization's weakest link. This idea was recently reinforced to me when I saw a T-shirt with the following text: “Social Engineering Expert: Because there is NO PATCH for Human Stupidity”.

Within the context of a Detachable Data Compartmentalization layered mobile computing defensive strategy, user education and awareness are paramount. Providing users a means to carry large quantities of sensitive or classified data places significant responsiblity on the user. The user must acknowledge and accept this responsibility in order for a DDC-layered defense strategy to be effective. An important point to emphasize is that **just because a user CAN do something, does not mean that they SHOULD**.

DDC must only be used in conjunction with a secured computing platform such as the user's hardened laptop system. As with public computing terminals, a vendor's computer, a customer's computer, or even a home, friend, or co-worker's sytem, the mobile computing user must acknowledge that keystroke logging and other monitoring methods could be active, and present a risk to their DDCM. A user must not insert their DDCM into any computing device unless it has been appropriately hardened, secured, and authorized in accordance with their company's security policies and procedures.

Otherwise, the DDCM could be subject to risks such as unauthorized copying, data modification, data destruction, contamination (e.g. via viruses, trojans or other attack vectors), or monitoring. Care must also be taken to insure that backup of DDCMs and transfer of files are done in a secure and approved manner (e.g. SSH, sFTP, scp, or copying to “sacrificial” removable media). Other more general mobile computing security issues, such as using Virtual Private Network (VPN) technology if connecting remotely to the company's internal network, and other encrypted communication protocols should be practiced in accordance with the organization's security policies and procedures.

Clearly, intelligent, educated, security-aware users are the most important layer of defense when addressing mobile computing and DDCM security.

Layer 2: Hardened, Reproducible Laptop Configurations

In theory, we should be able to configure hardened, reasonably secure computer systems. In practice, this task is extremely challenging. The unending flood of new security patches, new exploit discoveries, anti-virus updates, and other ongoing maintenance tasks make it difficult to keep a “secure” system secure. Mobile computing systems are specificially difficult and notorious for falling behind in the cycle of updates and maintenance because they are often outside the corporate network, or have access to high bandwidth connections for extended periods of time.

In order for us to effectively manage and track security updates, configuration changes, and software applications, we need to automate the process of performing these tasks. Advances in automated system administration, configuration management and patch management may offer solutions to this long-standing problem (Traugott et. al. 2003a; Traugott et. al. 2003b; Traugott and Huddleston 1998; PatchLink 2003; Marimba 2003; SecureInfo 2003). Instead of building and maintaining computing systems as “one-of-a-kind” entities, we need to develop automated, scalable solutions that will enable us to rigorously test and evaluate a small number of representative test systems, and then, based on our findings, be able to automatically apply changes to dozens, hundreds, and/or thousands of computer systems. We should also have the ability to “roll back” to any previous version if a problem is encountered.

This concept, common in version control of source code, is simply extended to manage large heterogenous computing networks (Traugott and Huddleston 1998). Generally, a central server maintains a list of configuration details for each computer system and client systems query the server and “pull down” their latest set of configuration updates. On occasion, a “push model” is also used. While this technology is often focused on building “thin clients” in a networked office environment, it has great potential for managing configurations of mobile computing systems using the Detachable Data Compartmentalization security model.

Using version-control-based automated configuration management, we should be able to sequentially build, test, harden and store pre-configured sets of mobile computing system configurations. If a company's mobile computer fleet is small or limited to a few types of hardware configurations, then implementing configuration and patch automation for the DDC model may be an ideal “test bed” for exploring the merits and cost-effectiveness for applying such automation technology across the organization's entire network.

Since the DDC model separates management of the computing platform from

the user's data, development of pre-built mobile system configurations is greatly simplified. If standard applications are deployed for established user groups within the company, then management of software configurations can also be scaled more effectively.

Imagine the following scenario: a Sales Engineer is about to leave the corporate office for a two week promotional tour. Before the engineer departs, they “check out” their data [presentations, sales contracts, non-disclosure agreements, technical documentation, product advertising, and other potentially sensitive information] onto a DDCM. They are then issued a laptop that has been configured with the software applications the Sales Engineer will need, hardened to protect against the latest security threats, and setup to work with their authorization credentials and DDCM. While away from the office, the laptop might be configured to “pull down” additional security updates and patches. Upon return to the office, the laptop is returned to the information technology staff, analyzed for security and policy breaches, sanitized, and then returned to the organization's “pool” of mobile laptops for use when the next user needs a hardened mobile computing system. Meanwhile, the Sales Engineer's data is retrieved from the DDCM, checked for integrity, planned updates, and potential security breaches, and then re-integrated back into the organization's networked information management infrastructure.

Licensing Issues and OpenSource Alternatives

As you can imagine, software and operating system licensing issues can significantly increase the complexity of attempting to automate deployment of operating systems, security patches, anti-virus updates, and suites of software applications. Simplification of licensing requirements, ability to review source code, and reduced costs are some of the motivations why I include OpenSource alternatives for operating systems [OpenBSD (OpenBSD 2003) and Linux (RedHat 2003; Linux Online 2003)] and key software applications (see below) in this paper.

Application	Function	URL
Mozilla Firebird	Web Browser	www.mozilla.org/products/firebird
Mozilla Thunderbird	Email Client	www.mozilla.org/products/thunderbird
OpenOffice	Office Suite	www.openoffice.org

Your organization may wish to evaluate three opensource, multi-platform software utilities: 1) Mozilla Firebird (Firebird 2003), 2) Mozilla Thunderbird (Thunderbird 2003), and 3) OpenOffice (OpenOffice 2003). Adoption of these free alternative tools offer the potential to create a familiar, consistent set of software applications that can be used by any user across a mixed PC/Mac/Linux/Unix heterogenous computing environment. Deployment of these

applications on your DDC mobile computing systems could be a way to evaluate these products on a small group of more technically adept users.

An Alternative to Automated Configuration Management

In the event that your organization does not have the technical or fiscal resources to implement a full-scale automated configuration management solution, there is another low-cost alternative to consider if you only need to manage one or a few number of mobile computing systems.

Purchase a large external hard drive (e.g. 120GB), and a copy of Norton Ghost by Symantec (Ghost 2003). You can then build and harden each computing system, and periodically take “images” of the hard drive as you work towards what will be a final “disk image” for a “reproducible, hardened laptop configuration” to deploy as your DDC solution. Thus, you will have the ability to restore an image to a new set of laptop hardware in the event a laptop is lost, stolen, or damaged.

DDC Mobile Computing System Hardening

Regardless of the method you use to create reproducible laptop configurations, you will need to insure that your configurations are hardened against potential security vulnerabilities. While the details of creating hardened OpenBSD, Linux and Win2K systems are beyond the scope of this paper, there are some basic concepts to keep in mind:

(1) Establish Strong Authentication
(2) Establish Strong Perimeter Defense
(3) Minimize Services Running
(4) Perform Vulnerability Assessments
(5) Patch System on Regular Basis

There are a number of resoucres available to help you understand the process of auditing and hardening computer systems depending on the operating system, services running, and purpose of the computing system. Resources on understanding hacking and security principles (Skoudis 2001; Friedl 2003; Cole 2001: McClure et. al. 2003; Cole et. al. 2003a; Cole et. al. 2003b; Harbour 2003), systems administration (Nemeth et. al. 2001; Lucas 2002; Urban & Tiemann 2003; Best 2003; Zakrzewski and Haddad 2002), secure coding principles (Graff and Van Wyk 2003), encryption (Singh 2001; Renfro 2002; Petullo 2003), and configuration and patch management (Traugott and Huddleston 1998; PatchLink 2003; Marimba 2003; SecureInfo 2003) can provide you with a good foundation for building secure computing platforms.

Hardening is a challenging, intensive process and having an automated method

to quickly deploy well-tested, hardened mobile computing systems is a keystone concept in building a successful DDC mobile computing program.

Layer 3: Detachable Data Compartmentalization Modules [DDCMs]

While our “Educated, Security-Aware Users” and “Reproducible, Hardened Laptop Configurations” are important steps in buiding a Detachable Data Compartmentalization security model for mobile computing users, the crux of the model is actually compartmentalizing your sensitive data onto external storage media, or a Detachable Data Compartmentalization Module [DDCM]. In many respects, the user's awareness and ability to safely access and protect their DDCM far out weigh any other defensive layers we discuss.

In order to compartmentalize data on a DDCM, it is necessary to:

(1) Identify the Sensitive Information
(2) Identify & Address any Data-Dependency Requirements
(3) Compartmentalize this Data in our DDCM

DDC security model requires an extremely good understanding of the data you are trying to protect. While the principle itself is extremely simple, effective implementation of DDC can be quite challenging. It can literally be like opening “Pandora's Box”. My advise is to start simple and progressively add additional complexity as you gain an appreciation for the concept and how it works. The primary challenges come from wrestling with potential data-dependency issues between software programs that may reside on the DDC hardened mobile computing system, and the data on the DDCM. Remember, the goal is that the mobile computing hardware is disposable and free of any sensitive information.

Proof of Concept: My First “Real World” Application

While the idea of DDC had floated around in my mind for quite some time, my first “real world” application of this technology came when I prepared to attend a series of security conferences in July and August 2003. While I have the utmost respect for my peers, I saw travelling to a series of security conferences as a high risk endeavor from the perspective of safeguarding any proprietary data. I imagined instances of wireless monitoring, having my laptop stolen, being socially engineered, or just falling prey to a rogue “blackhat” hiding in our crowd of information security professionals. In short, I was both paranoid and scared.

What made matters worse, was, for the training program I was to attend, I had to bring a laptop configured in such a manner that I knew it was “insecure”. This made sense because part of my training was to allow my classmates access to my computer as they attempted to perform various probes, exploits, and other

“hacking” attempts. But, it did little to ease my fears.

It was obvious that putting any proprietary or sensitive information on the laptop was out of the question. And, the SANS Institute clearly stated this point.

However, I knew I needed a way to continue to run my company during this conference and at the ones to follow. Luckily, my needs were fairly small. I needed to be able to update some source code, some technical documentation, and keep up with my email.

The laptop I brought with me was configured as a dual-boot system running RedHat Linux and Windows 2000. I installed the WinCVS 1.3.13beta version control system on the Windows2000 system (WinCVS 2003). I then setup a USB keychain to be used as my DDCM. To the DDCM I copied a subset of my personal CVS repository for the CVS modules I would need to access during my time outside of my corporation's network. I also copied a small, secure shell [SSH] application called PuTTY to the DDCM (PuTTY 2003).

With these tools and data, I was able to effectively implement a DDC mobile computing security program. The laptop system itself was sanitized and contained no personal information about myself or my company. However, I could insert the DDCM into the system, and from it, access source code and technical documentation I had stored in my CVS repository. I was careful to keep the CVS repository and any “checked out” modules or files on the DDCM. The computer system was also always un-networked whenever the DDCM was connected to the computer. Furthermore, after working on any files, I would delete the working copies so that only the CVS Repository was present on the DDCM. While I would have liked to have added encryption or similar technology to safeguard the data on the DDCM, I only had time to implement a basic DDC security model. However, I felt the cryptic nature of CVS repository files might offer some level of data protection in the event my DDCM was lost or stolen.

Accessing email was done by using PuTTY's **ssh** to port-forward my laptop's web browser from http://127.0.0.1 to my remote web-based mail software web page. Thus, I was able to encrypt the connection between my web browser and the remote email server. This set was necessary because the web-based mail program did not provide Secure Socket Layers [SSL] encryption. After responding to my email online, the cache of my web browser was cleared. PuTTY was also configured not to store any login information so that in the event the DDCM was lost or stolen, no SSH access or authorization information was left on the DDCM.

All in all, I was impressed with how my rudimentary DDC mobile computing security experiment worked. Using a USB keychain as a DDCM, I felt slightly comfortable leaving my laptop unattended (but hidden) in my hotel room as I went off to dinner, breakfast, lunch or other meetings. There was a nice secure

feeling knowing that my DDCM was always with me, and I frequently found myself giving it a reassuring “pat”.

Steganography: Concealment and Obsfucation Principles

Steganography, or stego, is simply the process of hiding information. Early methods of steganography were very simple, and if someone knew “where to look” and/or “what to look for”, the message was easily interpreted. Gradually, the principle of hiding messages (perhaps in a secret compartment, under a wax tablet, or written in invisible ink), was combined with cryptography giving a greater degree of secrecy even in the event the hidden message was discovered. Modern methods of steganography include hiding data (often encrypted) inside of digital photos, text files, or other media (Singh 2001; Judge 2001; Silman 2001; Cole 2001; Cole et. al. 2003b).

In this discussion of steganography, I am limiting myself to merely the idea of hiding the DDCM, obscuring the data on the DDCM, or leveraging the general lack of awareness about small mass storage devices. However, it has not escaped my attention that more sophisticated methods of steganography could be employed to add additional levels of defense to data stored on a DDCM.

Stego: Concealment

The small nature of USB keychains make them very easy to conceal, and via this property, protect them (and their data) from unauthorized access. While I have focused attention on USB keychains as DDCM devices, I would be remiss not to mention similar devices that could offer the same utility as DDCMs but increase the “concealment” factor. For instance, this past August, one of my peers was quite excited to show me his new “watch”. While the device looked like an ordinary digital watch, it was infact a USB mass storage device. The “cable” to connect his watch to his laptop computer was concealled within a special “belt” that he wore around his waist. I discovered another example this past week when flipping through some newspaper advertisements. There was an ad for an MP3 player that also had the ability to serve as a USB flash drive.

Clearly, the number and sophistication of these small mass storage devices will only continue to increase. While this bodes well for the availability of inexpensive, interesting, “gadgets” that can be deployed as DDCMs, it also means that the ability to copy and carry large amounts of potentially sensitive data is only getting easier.

The availability of such small mass storage devices could also represent a significant threat to an organization's infrastructure. Such devices could, if used inappropriately, be used to launch malicious software or steal proprietary information. Again, intelligent security policies and procedures, and an

education, security-aware workforce are probably the best strategy to take advantage of the benefits of this new storage technology, while at the same time, protect your organization from network breaches or corporate espionage.

Stego: Obfuscation

Obfuscation, or security through obscurity, has a potential application in how the DDCM is configured, or even, just on the type of data stored on the DDCM. However, the principle of security through obscurity is well recognized as weak, and should not be viewed as a “secure” solution for protecting data.

My use of CVS repository files on a DDCM is an excellent example of obsfucation, but really, offered absolutely no protection from a determined attack. I relied on the fact that CVS repository files “look weird”. Instead of normal file extensions, CVS repository files are appended with “,v”. However, all of the information is quite readable, and with a little research, an attacker who had never seen CVS files before could install CVS on a system and extract my files from its repository format into the more “expected” file formats. But, **if** someone just happened upon my lost DDCM, **maybe** they might look on the drive, find my contact information (contained in a REWARD_IF_RETURNED.txt file), note that the bulk of the information looked very strange, and then take steps to return the lost DDCM. **Maybe**.

This point of obfuscation providing a poor level of security is important to keep in mind when evaluating hardware for use as DDCMs, if the goal is to take advantage of password-protection programs or other “security features”. For example, during my assessment of USB keychain devices, I included a quick review of Lexar Media's JumpDrive Secure (JumpDrive 2003). This product included a small executable software program called “Safe Guard” that would reside on the laptop. “Safe Guard” (JumpDrive 2003) allowed me to create a password-protected “Private” zone on the USB keychain in addition to a “Public” zone. The Safe Guard program controlled which USB keychain zone [the “Private” or “Public” portion] was visible to Windows2000. It is interesting to note, that while I was not able to access “Private” sections of the USB keychain by attempting to access the device from Linux [where only the “Public” portion was USB mass storage device was viewable] or OpenBSD [where I was unable to access the device at all], Lexar Media provides no information concerning how the information is protected, and even if it is encrypted at all. In this respect, this piece of hardware [and its software-based access control] seem to be leveraging the principle of obfuscation [where the vendor is not revealing detailed information about their security process]. Until such time I can verify such devices use established, publically-vetted encryption algorithums such as RSA, DES, triple DES, AES, or Blowfish, I will treat such password-protection “security features” as weak.

Use of alternative file formats are another way that obfuscation methods could

be added to the DDCM. For example, a small DOS partition could be created on the DDCM to provide sanitized contact information (e.g. PO Box, an external voicemail drop, etc) and maybe some bogus data. Such information might aid an “honest person” that was seeking to return it to the owner. If a DDCM with multiple partitions is inserted into a Windows computer system, it is possible that only the Windows readable file systems (i.e. FAT, FAT32, or NTFS) will be noticed and investigated. Other alternate partition formats (e.g. OpenBSD's UFS, Linux's ext2fs, Win2000's NTFS, etc) might afford some level of increased security through obsfucation. Milo's (Milo 2001) listing of various file formats may be useful in designing a DDCM using a relatively obscure file format. The methods described in my USB keychain system integration section will allow you to use your USB keychain as a DDCM in any of the described operating systems. This is because in each example, I have left the USB Keychain flash drive with the default FAT or MSDOS drive format. Consequently, I am not taking advantage of any obfuscation or increased performance benefits that may exist from using an alternative format.

If you wanted to dedicate your DDCM device for use on OpenBSD or Linux, you could reformat it to use one of the Unix or Linux alternative drive formats (Lavigne 2001a; Lavigne 2001b; Lucas 2002; Nemeth et. al. 2001). If the DDCM was only going to be used on Win2K, you could use the NTFS file format to offer some level of increased protection. The process for formatting a USB keychain under OpenBSD, Linux and Windows 2000 into a few different file formats are listed below.

OS	File Format	Formatting Steps
OpenBSD	[OpenBSD Native]	**# newfs /dev/sd0i** or **# fdisk /dev/sd0**
OpenBSD	FAT [msdos FAT16]	**# fdisk /dev/sd0** (Use menu)
OpenBSD	FAT32 [msdos FAT32]	**# fdisk /dev/sd0** (Use menu)
Linux	Ext2 [Linux Native]	**# mke2fs /dev/sda1**
Linux	FAT [msdos FAT16]	**# mkdosfs -F 16 /dev/sda1**
Linux	FAT32 [msdos FAT32]	**# mkdosfs -F 32 /dev/sda1**
Linux	FAT [generic]	**# mkfs -t vfat /dev/sda1**
Win 2K	FAT [msdos FAT16]	Use Windows Explorer “**Format...**” option
Win 2K	FAT32 [msdos FAT	Use Windows Explorer “**Format...**” option
Win 2K	NTFS	Use Windows Explorer “**Format...**” option

Future DDCM Software Application Support

The user's web browser and email client are two primary applications that need to be supported on the DDCM. I am currently investigating how two Mozilla Open Source applications (Mozilla Firebird and Mozilla Thunderbird) can be deployed

on removable media such as a DDCM.

The advantage of having the web browser deployed in such a fashion is that potentially sensitive (but useful) information such as a list of bookmarks or browsing history follow the user when they utilize a DDC mobile computing system.

The advantage of having email client data stored on the DDCM is that the user would have access to such information as contact lists and historical correspondence.

Indeed, I am not alone in such endeavors, and there has been recent discussion on the Mozilla forum for attempting to get Mozilla Thunderbird to run from portable hard drives so that users can access their email consistently as they move from one workstation to another (Mozilla Forum 2003a; Mozilla Forum 2003b; Mozilla Forum 2003c; Mozilla Forum 2003d).

Layer 4: Add Partition-Level or Disk-Level DDCM Encryption

When I employ obfuscation methods, I still treat the DDCM as though it would be transparent to any potential thief. Even if more sophisticated methods of security such as encryption were involved, an intelligent “human element” controlling the safety and security of the DDCM is what gives the DDC mobile computing security model its strength.

Nonetheless, people are fallible, and in the same way we lose car keys, wallets, purses, laptops, and even SecureIDs, it is reasonable to expect that someone will lose their DDCM. Strong encryption of the DDCM's data would make it significantly less likely that someone finding or stealing a DDCM could obtain access to any confidential information in a reasonable period of time.

While file-level encryption, such as Pretty Good Privacy (PGP 2003), is an option for protecting DDCM data, it may be more effecient to develop a scheme to automatically encrypt and decrypt information “on the fly” as it is written and read from a dedicated DDCM partition on the mobile hard drive, or even the entire DDCM disk. An encrypted DDCM partition might be used if it was desired to have a second unprotected partition on the DDCM that contained public contact information so that the DDCM could be returned in the event that it was lost and then found. The entire DDCM might be encrypted if it were to be treated as completely disposable in the event it was lost or stolen.

A discussion of specifics on how strong encryption technology can be applied to the DDCM is beyond the scope of this paper. However, our logic for providing system integration information for configuring USB keychains as DDCMs under OpenBSD, Linux and Win2K is that we give the reader a range of potential

options to tackle this issue. For technically adept readers, they may choose to investigate how to setup an open source solution for strong encryption using OpenBSD or Linux. For readers that are more comfortable with vendor-based solutions, a commercial software option may be more appropriate.

Readers wishing to investigate OpenBSD or Linux solutions may find various discussions of cryptographic frameworks and cryptographic file systems [CFS] (de Raadt et. al. 1999; Keromytis et. al. 2003; Blaze 1993; Provos 2000) or the Cryptographic API for Linux (Butcher 2002; Bryson 2002) useful resources.

Layer 5: Add Token-Based Authentication and/or Cryptography

Use of strong encryption technology frequently requires the use of digital keys or certificates in addition to passwords or passphrases. If such technology is required to complete the encryption steps on the DDCM, it will be necessary to protect such sensitive information in the same manner as we have the data in the DDCM. Thus, a second USB keychain or other storage device (PCMCIA card, CDROM, or other external media) may be needed to manage the encryption key assets. The important point is that if such technology is utilized, the encryption key or certificate must get the same level of vigilence as the DDCM itself and can NOT be stored on the DDC mobile computing system.

For readers that are trying to develop their own token-based authentication or cryptography methods, Matt Blaze and his peers have written papers that may be useful (Blaze et. al. 1998; Blaze 1996).

For readers interested in commercial solutions to token-based authentication or cryptography methods, there are a significant number of commercial USB token devices that provide strong encryption technologies using algorithums such as RSA, DES, triple DES, AES , SHA or Blowfish (RSA 2003; Perera 2002; Aladdin 2003; DesLock 2003; Authenex 2003; Raak 2003)

Finding Balance: Developing a “Secure-Enough” Solution

Graff and Van Wyk discuss the concept that nothing can be completely secure, and that when deciding to produce software we need to adopt a mentality of “secure-enough” (Graff and Van Wyk 2003). The “secure-enough” principle takes into account the resources available to a project, the level of security required, and the probable risk profile. Basically, you seek to find an effective balance in the cost-benefit ratio between how much effort goes in to securing the application, versus usability, training, and other resource demands.

Determining the “Secure-Enough” balance for a DDC mobile security model will

differ from organization to organization, and will likely change through time. Even within a given organization, different DDC mobile security models may be employed given changing factors such as the sensitivity of the data, the expertise of the end user, the potential risk for the system, etc.... Additional layers of defense may be added to different DDC mobile computing systems in accordance with the relative need to safeguard the confidentiality and integrity of the data stored on its DDCM.

Not all five layers of my outlined “Prioritized, DDC Defensive Layers Strategy” need to be implemented in order to start leveraging the security benefits of DDC. In fact, Layer One [educated, security-aware users] and Layer Three [DDCMs] are “enough” to get a program started. While Layer Two [hardened, reproducible laptop configurations] is important in developing a scalable solution, a rudimentary DDC mobile computing security model could be run with a single hardened system. I see Layer Four [addition of partition-level or disk-level encryption to DDCM] and Layer Five [addition of token-based authentication and/or encryption] as useful where data confidentiality and intergrity needs were high, but their implementation should be carefully assessed and balanced with tradeoffs such as increasing the DDC security model complexity, placing additional demands on information technology staff and mobile computer users, and possibly creating a false sense of added security.

USB Keychain Hard Drive System Integration for Use as a DDCM

The technical focus for this article is how to configure USB keychains for use as DDCMs under OpenBSD, Linux or Win2K. USB mass storage device selection should be made with care, and tested thoroughly on the intended operating system architecture. [Note: While most USB keychains do not advertise support for OpenBSD or Linux, many may function in these operating systems without difficulty. However, care should be taken when exploring use of these devices under various operating systems and the risk of data loss or hardware destruction must be recognized. You have been warned.]

The first step to using your USB Keychain, flash drive, or other mobile storage device as a DDCM is to **mount** it for your operating system. This step enables you to access the device as a distinct hard drive or directory, just as you would with a floppy drive or cdrom. In the following sections, I describe this process for a Memorex “256mb ThumbDrive” (ThumbDrives 2003) under our three target operating systems.

OpenBSD

Initially, we determine how to get the USB Keychain or USB Flash Drive working using the ***root*** user account. Then, we configure the system so that any user can **mount** and **umount** such a USB mobile mass storage device.

Using root to mount your USB Keychain

Make sure that your USB Keychain is plugged in during the initial system startup. You can take a look at your **dmesg** output to see if your USB Keychain was recognized. The following illustrates this process:

```
$ dmesg | tail
    umass0 at uhub0 port 1 configuration 1 interface 0
    umass0: USB Solid state disk, rev 1.10/1.00, addr 2
    umass0: using SCSI over BBB-P
    scsibus1 at umass0: 2 targets
    sd0 at scsibus1 targ 1 lun 0: <Memorex, ThumbDrive, 1.11> SCSI2 0/direct removable
    sd0: mode sense (4) returned nonsense; using fictitious geometry
    sd0: 252MB, 252 cyl, 64 head, 32 sec, 512 bytes/sec, 516096 sec total
    sd0: mode sense (4) returned nonsense; using fictitious geometry
    sd0: mode sense (4) returned nonsense; using fictitious geometry
    sd0: mode sense (4) returned nonsense; using fictitious geometry
```

From this output, we can see that the USB Keychain was recognized and that it is somewhere on ***/dev/sd0***. However, we need another step to determine the exact partition of ***sd0*** that we will need to **mount**. This can be accomplished using the **disklabel** command as ***root*** (Starling 2003).

```
$ su
Password:
# disklabel sd0
    # /dev/rsd0c:
    type: SCSI
    disk: SCSI disk
    label: ThumbDrive
    flags:
    bytes/sector: 512
    sectors/track: 32
    tracks/cylinder: 64
    sectors/cylinder: 2048
    cylinders: 252
    total sectors: 516096
    rpm: 3600
    interleave: 1
    trackskew: 0
    cylinderskew: 0
    headswitch: 0           # microseconds
    track-to-track seek: 0  # microseconds
    drivedata: 0
```

```
16 partitions:
#       size   offset    fstype   [fsize bsize   cpg]
  c:   516096        0    unused        0     0          # (Cyl.    0 - 251)
  i:   515552       32     MSDOS                         # (Cyl.    0*- 251*)
```

This excerpt of the **disklabel sd0** output shows us that the MSDOS partition is on the ***/dev/sd0i*** partition. Therefore, we should be able to **mount** and access the USB keychain to the ***/mnt/keychain-temp*** directory with the following commands:

```
# mkdir -p /mnt/keychain-temp
# mount /dev/sd0i /mnt/keychain-temp
# cd /mnt/keychain-temp
# ls
```

You should now be able to use the USB keychain hard drive as ***root***. However, our goal is to configure OpenBSD so that any user can **mount** and **umount** a USB keychain. Although there may be situations where you could use the ***root*** permission requirements to limit users' abilities to use such USB-based mass storage devices.

You can **umount** the USB keychain and remove our temporary mounting directory with the following commands. We will then move on to configure the system to support general USB keychain usability.

```
# cd
# umount /mnt/keychain-temp
# rmdir /mnt/keychain-temp
```

Configuring your USB Keychain for Manual Mounting

BSD systems do not allow you to simply modify your ***/etc/fstab*** entry to allow non-***root*** users the ability to **mount** and **umount** devices such as CDROMs, floppies and USB mass storage devices (Bouyer 1999).

However, your BSD system can be customized to permit non-***root*** users to be able to access these removable storage devices. The following steps must be completed:

- Create a special ***usbfob*** group
- Add your USB keychain users to the ***usbfob*** group
- Modify the group ownership and permissions of the USB keychain ***/dev/sd0i*** device
- Modify ***/etc/sysctl.conf*** to permit non-***root*** mounting

First, we created the new group ***usbfob***, and added ***root*** and user ***john*** [I am using myself as an example user] to the group.

```
$ su
Password:
# cat /etc/group | tail -2
nogroup:*:32766:
nobody:*:32767:

# echo "usbfob:*:32768:root,john" >> /etc/group
# cat /etc/group | tail -3
nogroup:*:32766:
nobody:*:32767:
usbfob:*:32768:root,john
```

In this example, our last two ***/etc/group*** entries had GIDs of 32766 and 32767, so I specified the GID of 32768 for our ***usbfob*** group. You will want to pick users and a GID to fit your own system's configuration.

Next, we checked the default permissions on the ***/dev/sd0i*** device, changed the group ownership to our new ***usbfob*** group, and lastly added read and write permission for the ***usbfob*** group.

```
# ls -lt /dev/sd0i
brw-r-----  1 root  operator    4,   8 Aug 15 20:45 /dev/sd0i

# chgrp usbfob /dev/sd0i
# chmod 660 /dev/sd0i
# ls -lt /dev/sd0i
brw-rw----  1 root  usbfob    4,   8 Aug 15 20:45 /dev/sd0i
```

Finally, we modify the ***/etc/sysctl.conf*** configuration file to permit non-***root*** mounting of devices. Of course, non-***root*** mounting requires that appropriate permissions exist on the device such that a non-***root*** user could **mount** the device. Our customization of ***/dev/sd0i*** with the ***usbfob*** group is one such modification.

```
# echo "kern.usermount=1   # 1=Enable non-root device mounting"  >> /
etc/sysctl.conf
```

Note: On other BSD-style operating systems (such as FreeBSD or NetBSD) you may need to replace the **kern.usermount=1** with **vfs.usermount=1**. Check your distribution's documentation or run a search on ***usermount*** if you are running a BSD operating system other than OpenBSD.

Our last step is for our user ***john*** to create their own directory for mounting the

USB keychain, and then **mount** the device. It may be necessary to reboot your system before completing these steps.

```
# exit
$ cd
$ mkdir -p usbfob
$ mount -t msdos /dev/sd0i /home/john/usbfob
$ ls -lt /home/john/usbfob
```

If you're successful, you will see that the files located on the USB keychain hard drive belong to the owner and group ***john***. This is preferable to using **su** to **mount** the USB Keychain because you can operate at the ***user*** level rather than operate as ***root***. This strategy of operating at a stage of "Least Privilege" improves your overall Defense-in-Depth method for protecting your data and your system (Cole et. al. 2003a; Cole et. al. 2003b).

RedHat Linux

Initially, we will get your USB keychain hard drive working using the ***root*** account. Once we have confirmed that we can access and use the mobile mass storage device as ***root***, we will reconfigure your system so that any user can access and use the USB keychain without needing to **su** or otherwise increase their user privilege.

Using root to mount your USB Keychain

For our initial USB keychain hard drive attempts, you will need ***root*** access on the computer. Take a look at your **dmesg** output to determine if your USB keychain was recognized. I found that it was necessary to have the keychain installed during system startup, or I was not able to get it to **mount**. Here is a sample of my **dmesg** output:

```
SCSI subsystem driver Revision: 1.00
Initializing USB Mass Storage driver...
usb.c: registered new driver usb-storage
scsi0 : SCSI emulation for USB Mass Storage devices
  Vendor: Memorex   Model: ThumbDrive       Rev: 1.11
  Type:   Direct-Access                     ANSI SCSI revision: 02
WARNING: USB Mass Storage data integrity not assured
USB Mass Storage device found at 2
USB Mass Storage support registered.
```

It took me a bit of trial and error to determine the appropriate ***device*** for my USB keychain. It turned out to be loaded into the system as ***/dev/sda1***. The following set of commands were used to **mount** the USB flash drive:

```
[user]$ su
```

```
Password:
[root]# mkdir -p /mnt/keychain-temp
[root]# mount /dev/sda1 /mnt/keychain-temp
```

Once the device is mounted, you will be to access it on the ***/mnt/keychain-temp*** directory. However, by default, you may find that the files and everything else are all owned by ***root***, which can make working on files somewhat challenging if you do not have **su privileges.** Moreover, you'll need to have the USB keychain plugged into the USB port during the boot sequence in order for you to be able to **mount** the mobile flash drive.

These usability issues for USB keychain hard drives under Linux can be easily overcome with a few minor configuration steps [which we present in the next section]. Alternatively, these very requirements for ***root*** or **su** access could be used to limit general use of USB mass storage devices within a Linux-based network environment. As with all security, there is a balance between security and usability, which will vary from organization to organization as defined by its security policies, procedures and culture.

USB Keychain Hard Drive Mounting Options

I present two options for general use of a USB Keychain. My preference is for Option Two. You should only use one of the options presented.

However, before I get started we will **umount** the USB keychain drive and remove the temporary mount directory, ***/mnt/keychain-temp***, we created during our ***root*** USB activation above. These operations are still performed as ***root***.

```
[root]# cd /
[root]# umount /mnt/keychain-temp
[root]# rmdir /mnt/keychain-temp
```

Option One: Configuring your USB Keychain for Automounting

In most instances, you probably don't want to have to reboot your computer each time you want install your USB keychain. It is also likely that you won't want to have to **su** to ***root*** every time you want to **mount** the mobile flash drive, or write any files.

An elegant solution to this ***root*** permission issue is to use the **autofs** command (Hermann 2003). This is done with modification of two configuration files: ***/etc/auto.master*** and ***/etc/auto.keychain***.

First, we will edit the ***/etc/auto.master*** configuration file and prepare our

setup for automatic mounting of USB keychains. If your system is configured by default with **autofs** you will already have a ***/etc/auto.master*** file, and it is simply a matter of editing ***/etc/auto.master***. RedHat 8.0 Linux includes **autofs** by default.

The following is the contents of our modified ***/etc/auto.master*** file:

```
# $Id: usb_flash_drives.xml,v 1.2 2003/08/23 15:08:52 john Exp $
# Sample auto.master file
# Format of this file:
# mountpoint map options
# For details of the format look at autofs(8).
# /misc /etc/auto.misc   --timeout=60
/mnt/keychain /etc/auto.usbfob --timeout=10
```

Next, we will create a ***/etc/auto.usbfob*** file that will enable any user of the computer to mount a USB keychain hard drive, or fob. Here are the steps we took to create our ***/etc/auto.usbfob*** file.

```
[user]$ su
Password:
[root]# echo "usbfob -fstype=vfat,sync,umask=000 :/dev/sda1" > /
etc/auto.usbfob
```

Now, any user of the computer will be able to access the USB keychain fob by simply using **cd /mnt/keychain/usbfob**. In addition, because **autofs** is used to automatically **mount** the USB mass storage device, it does not need to be plugged in during the boot process. Another advantage of this **autofs** automounting solution is that the ***/mnt/keychain/usbfob*** is automatically unmounted after 10 seconds of inactivity (assuming you are not in the ***/mnt/keychain/usbfob*** directory).

While this option works quite well, my preference is for a manual mounting option. This solution offers the same benefits of the automounting option, but I find the steps of actually typing **mount** and **umount** reassuring.

Option Two: Configuring your USB Keychain for Manual Mounting

Another solution is to configure your ***/etc/fstab*** so that any user can **mount** and **umount** the USB keychain. This solution is similar to the automounting option above, but users will have to manually type commands. The following commands can be used to update your USB keychain so that it can be mounted under ***/mnt/usbfob***:

```
[user]$ su
```

```
Password:
[root]# echo "/dev/sda1 /mnt/usbfob vfat noauto,user 0 0" >> /etc/fstab
[root]# mkdir /mnt/usbfob
[root]# exit
```

Now, the USB keychain can be mounted by any user of the computer under the ***/mnt/usbfob*** directory. Moreover, once the USB keychain is mounted, the user who typed the **mount** command will be seen as the owner of the drive and associated files. Thus, there will not be any permission issues for writing to the drive.

The USB keychain flash drive is easily mounted with the following command.

```
[user]$ mount /mnt/usbfob
```

Once you are finished using the drive, you can leave the ***/mnt/usbfob*** mounted directory and **umount** the USB flash drive.

```
[user]$ cd
[user]$ umount /mnt/usbfob
```

Windows 2000

Windows 2000 does a good job of automating the system integration process. Most USB flash drives are automatically detected and configured as a removable hard drive upon insertion into an active USB port. This is also true for Windows XP computers.

However, this same ubiquitous support for USB-based hardware under Windows 2000 may also make it difficult to enforce a security policy that forbids use of USB mass storage devices. My understanding is that Windows 2000 does not distinguish between devices such as USB mice, USB keyboards, or USB keychain hard drives when applying Windows 2000 local security policies. Thus, some of the finer-grained access control possible with OpenBSD and Linux for limiting USB keychain support, may be more challenging to implement under Windows 2000.

Removing your USB Keychain Hard Drive

Safe Removal from OpenBSD and Linux Systems

I have read several discussions about making sure you **umount** your USB keychain mass storage device before removing the USB keychain. Otherwise, you risk losing the information on the mobile hard drive.

I recommend shutting down your computer before you remove the USB keychain. This will insure the LED light will be off, which has been recommended before removal (Hermann 2003).

In theory, we should be able to get the USB keychain LED light to turn off, and remove it from the computer, without having to power down the entire computer system. Unfortunately, I have not been able to figure out how to do get the Memorex ThumbDrive USB keychain LED to turn off. Using **umount** will get the LED light to stop blinking, but the LED will remain lit. Others have had similar difficulty safely disconnecting their USB flash drives (Thach 2003), and there seems to be some consensus that it is okay to remove the USB keychain if it has been un-mounted with **umount** (Butcher 2002; Thach 2003). I have on occasion removed un-mounted, LED lit USB keychains from OpenBSD and Linux systems without loss of data or other ill effects.

Safe Removal from Windows Systems

The process of removing a USB keychain hard drive from a Windows system is straightforward. Under Windows 2000, an “Unplug or Eject Hardware” icon is displayed in the system tray, and clicking on it provides the option to “Stop USB Mass Storage Device...”. Choosing this option (often there are more than one that will accomplish this task) will result in the USB keychain's LED to shut off and a message to be displayed stating it is safe to remove the hardware.

I recommend removing the USB chain only after shutting down the device as described above, or after shutting down the entire computer system. On occasion, I have also accidentally removed a USB Keychain without following these steps, and while I immediately get Windows 2000 warnings, I have not lost any data.

Backup and Recovery Procedures

Backing up the USB Keychain hard drives is an important step in the DDC model. Backup of these systems should be done in accordance with established policies and procedures for storage of sensitive data. For example, it may be appropriate to compress and encrypt the portable hard drive data, and then burn this information to media such as CD-R before being stored in a designated location. The backup and recovery procedure should also account for concepts such as storage and retreival of encyption keys, and the possibility for unavailability, dismissal, or even death of the DDCM user/owner.

During my testing of USB Keychain, I only experienced one catastrophic failure. This occurred when the USB keychain was inserted in a USB port during the middle of a RedHat Linux bootup sequence. All partitioning information and data

on the USB keychain was lost. A variety of forensic disk utilities were employeed to try and recover the lost data, but all attempts failed. However, after data recovery efforts were abandoned and the USB keychain hard drive was reformatted, the USB keychain functioned fine without any future failures.

It is important to note that removal of the keychain without proper dismounting [under OpenBSD and Linux] or device stopping [under Windows 2000], did not cause any failures. However, I never attempted to remove a USB keychain in the middle of a data transfer.

While I feel failure of these small flash drives is relatively rare, it is still important to practice routine backups of the devices. One scenario I have imagined is that computer users have two USB keychains, and the information is mirrored between both disks. Thus, there is redundancy at the DDCM level.

Conclusions

In this paper, I put forward a new way to think about external media and how such media can be used to create detachable compartments, or DDCMs, for safeguarding the integrity and confidentiality of sensitive data. This concept was formally presented as Detachable Data Compartmentalization, or DDC, where hardened, sanitized mobile computing system functions soley to provide availability to compartmentalized, sensitive data stored on a DDCM.

In my discussion of DDC, I presented a prioritized, DDC layered defensive strategy for protecting mobile computing data. This layered strategy includes: 1) educated, security-aware users, 2) hardened, reproducible laptop configurations, 3) DDCMs, 4) addition of partition- or disk-level encryption, and 5) addition of token-based authentication and/or cryptography. I emphasized that while all layers can be used, the first three are the most important, and that balance needs to be exercised to build appropriate, “secure-enough” solutions.

The paper concluded with a technical discussion of how to configure a Memorex 256mb ThumbDrive USB keychain as a DDCM under either OpenBSD, Linux or Windows 2000. Minimal additional testing suggests that Lexar Media 128mb JumpDrive Secure USB keychains may function under Linux or Windows 2000, but not under OpenBSD.

In closing, I would like to share a story of a recent laptop theft, where data belonging to Bank Rhode Island would have been protected had the laptop used the DDC security model described in this paper. This stolen laptop contained “the names, addresses and social security numbers of about 43,000 customers”, and the data on the laptop hard drive was not encrypted (Mearian 2003; SANS NewsBites 5/51). This theft and the loss of such highly sensitive information underscores the need to physically de-couple mobile data from mobile

computing systems using a process such as DDC. The overwhelming response to this theft is that the data on the laptop should have been encrypted. While I agree that the data should have been encrypted, I would also advocate that the data should only have been stored on a DDCM and that the laptop itself be free of any sensitive information.

The concept of Detachable Data Compartmentalization [DDC] and the use of Detachable Data Compartmentalization Modules [DDCMs] provide powerful, innovate methods to add new layers of defense for protecting mobile computing data. I look forward to their widespread use in computer information security.

References

Aladdin. 2003. "Aladdin eToken Portable USB Authentication Devices". URL: http://www.ealaddin.com/etoken/default.asp (December 2003).

Artner, Bob. 2002. "Lessons Learned from the Great TechRepublic Laptop Theft". URL: http://techrepublic.com.com/5100-6314-1051250.html (December 2003).

Authenex. 2003. "Authenex: Affordable Strong e-Security". URL: http://www.authenex.com (December 2003).

Best, Steve. 2003. "Halt! Who Goes There? Access Control Lists Are Far More Flexible". Linux Magazine, September 2003 (vol 5/issue 9), pp. 24-29, and 64.

Blaze, M., Feigenbaum, J., and Moni Naor. 1998. "A Formal Treatment of Remotely Keyed Encryption (Extended Abstract)". In Eurocrypt 1998. Helsinki. LNCS 1403.

Blaze, Matt. 1993. "A Cryptographic File System for Unix". In Proceedings of the 1st ACM Conference on Computer and Communications Security. November 1993.

Blaze, Matt. 1996. "High-Bandwidth Encryption with Low-Bandwidth Smartcards". In Cambridge Workshop on Fast Software Encryption. February 1996.

Bouyer, Manual. 1999. "Re: user mount". URL: http://mail-index.netbsd.org/netbsd-help/1999/11/06/0000.html (August 2003).

Bryson, David. 2002. "The Linux CryptoAPI: A User's Perspective". URL: http://www.kerneli.org/howto/index.php (November 2003).

BUPD. Boston University Police Department. 2002. "Laptop Theft Alert (25 Feb 2002)". URL: http://www.bu.edu/police/advisory/wanted/laptop_theft_advisory.htm (December 2003).

Butcher, Matt. 2002. "Configure GCT's USB Palm Key in Linux". URL: http://www.gctglobal.com/Download/3rd_LED/PalmKey/palmkey.html (August 2003).

Butcher, Matt. 2002. "Configure GCT's USB Palm Key in Linux". URL: http://www.gctglobal.com/Download/3rd_LED/PalmKey/palmkey.html (November 2003).

Cole, E., Fossen, J., Northcutt, S. and Hal Pomeranz. 2003. SANS Security Essentials with CISSP CBK Version 2.1. Volume One. Published by SANS Press.

Cole, E., Fossen, J., Northcutt, S. and Hal Pomeranz. 2003. SANS Security Essentials with CISSP CBK Version 2.1. Volume Two. Published by SANS Press.

Cole, Eric. 2001. Hackers Beware: Defending Your Network from the Wily Hacker. Published by New Riders Publishing.

DCID 6/3. Director of Central Intelligence Directive 6/3. 1999. "Protecting Sensitive Compartmented Information Within Information Systems: Policy". URL: http://www.fas.org/irp/offdocs/DCID_6-3_20Policy.htm (November 2003).

de Raadt, T., Hallquvist, N., Grabowski, A., Keromytis, A. D., and Niels Provos. 1999. "Cryptography in OpenBSD: An Overview". In Proceedings of the USENIX Annual Technical Conference, FREENIX Track. June 1999.

DesLock. 2003. "DesLock USB Token". URL: http://www.deslock.com/about.e.htm (December 2003).

DocBook. 2003. "Simplified DocBook Document Type Definition". URL: http://www.docbook.org (December 2003).

Firebird. 2003. "Mozilla Firebird: OpenSource Web Browser". URL: http://www.mozilla.org/products/firebird (December 2003).

Friedl, Stephen. 2003. "Analyze This: Using Nmap and Nessus to Find Security Risks". Linux Magazine, May 2003 (volume 5, issue 5), pp. 20-25.

Gallagher, Michael J. 2001. "Centralized Backups. SANS InfoSec Reading Room: Backup Strategies". URL: http://www.sans.org/rr/papers/index.php?id=515 (December 2003).

Ghost. 2003. "Norton Ghost 2003". URL: http://www.symantec.com/sabu/ghost/ghost_personal/ (December 2003).

Graff, Mark G. and Kenneth R. Van Wyk. 2003. Secure Coding: Principles and Practices. Published by O'Reilly & Associates.

Grant, Chris. 2003. "Defense-In-Depth Applied to Laptop Security: Ensuring Your Data Remains Your Data. SANS InfoSec Reading Room: Best Practices". URL: http://www.sans.org/rr/papers/index.php?id=1263 (December 2003).

Gulati, Radha. 2003. "The Threat of Social Engineering and Your Defense Against It. SANS InfoSec Reading Room: Social Engineering.". URL: http://www.sans.org/rr/papers/index.php?id=1232 (December 2003).

Harbour, Thomas. 2003. "Defence in Depth on the Home Front. SANS InfoSec Reading Room: Home & Small Office". URL: http://www.sans.org/rr/papers/26/1033.pdf (December 2003).

Hermann, Uwe. 2003. "How To Use The Aiptek PenDisk On A GNU/Linux System". URL: http://www.hermann-uwe.de/pendisk.php (August 2003).

Hiebert, Bob. 1997. "Airport Laptop Theft. Laptop Thieves Part I - The UL. Laptop Thieves Part II - UL Propagation". URL: http://www.urbanlegends.com/misc/airport_laptop_theft.html (December 2003).

JHSPH. Johns Hopkins Bloomberg School of Public Heath Information Systems. 2003. "Laptop Theft Prevention". URL: http://intra1.jhsph.edu/Administration/InformationSystems/AdviceTraining/laptopTheft.html (December 2003).

Judge, James C. 2001. "Steganography: Past, Present, Future. SANS InfoSec Reading Room: Steganography". URL: http://www.sans.org/rr/papers/index.php?id=552 (December 2003).

JumpDrive. 2003. "Lexar Media JumpDrive Family of USB Products". URL: http://www.lexarmedia.com/jumpdrive/index.html (December 2003).

Kay, Russel. 2001. "Blowing the Whistle on Laptop Theft. Computerworld". URL: http://www.computerworld.com/mobiletopics/mobile/story/0,10801,59091,00.html (December 2003).

Keromytis, A. D., Wright, J. L., and Theo de Raadt. 2003. "The Design of the OpenBSD Cryptographic Framework". URL: http://www.openbsd.org/papers/ocf.pdf (December 2003).

Lavigne, Dru. 2001a. "Dividing Your Data. O'Reilly Network: FreeBSD Basics". URL: http://www.onlamp.com/pub/a/bsd/2001/02/21/FreeBSD_Basics.html (December 2003).

Lavigne, Dru. 2001b. "Understanding Unix Filesystems. O'Reilly Network: FreeBSD Basics". URL: http://www.onlamp.com/pub/a/bsd/2001/02/28/FreeBSD_Basics.html (December 2003).

Linux Online. 2003. "Linux Online". URL: http://www.linux.org (December 2003).

Lucas, Michael. 2002. Absolute BSD: The Ultimate Guide to FreeBSD. Published by No Starch Press.

Marimba. 2003. "Marimba Configuration and Patch Management". URL: http://www.marimba.com (December 2003).

McClure, Stuart, Joel Scambray and George Kurtz. 2003. Hacking Exposed: Network Security Secrets & Solutions. 4th Edition. Published by McGraw-Hill/Osborne.

Merian, Lucas. 2003. "BankRI Customer Information Stolen Along with Laptop. Computerworld (19 December 2003)". URL: http://www.computerworld.com/printthis/2003/0,4814,88443,00.html (December 2003).

Merle, Renae. 2003. "Trial Due in Boeing Wrongful-Termination Suit. Washington Post. Wednesday, December 10, 2003: Page E01". URL: http://www.washingtonpost.com/wp-dyn/articles/A50910-2003Dec9.html (December 2003).

Microsoft. . "Microsoft Windows 2000 Professional Workstation". URL: http://www.microsoft.com/windows2000/default.asp (December 2003).

Milo. 2001. "OSdata.com Website: File Systems". URL: http://www.osdata.com/holistic/connect/filesys.htm (December 2003).

Mitnick, Kevin D., and William L. Simon. 2003. The Art of Deception : Controlling the Human Element of Security. Published by John Wiley & Sons.

Mozilla Forum. 2003a. "Running Thunderbird on a USB Drive". URL: http://forums.mozillazine.org/viewtopic.php?t=34245 (December 2003).

Mozilla Forum. 2003b. "Portable Hard Drive". URL: http://forums.mozillazine.org/viewtopic.php?t=32998 (December 2003).

Mozilla Forum. 2003c. "Portable Hard Drive". URL: http://forums.mozillazine.org/viewtopic.php?t=32384 (December 2003).

Mozilla Forum. 2003d. "Changing Locations Where Profiles Are Stored". URL: http://forums.mozillazine.org/viewtopic.php?t=33687 (December 2003).

Mueller, Andrew. 2001. "Laptop Security: Past, Present. SANS InfoSec Reading Room: Travel Security". URL: http://www.sans.org/rr/papers/index.php?id=409 (December 2003).

Nemeth, Evi, Garth Snyder, Scott Seebass, Trent R. Hein. 2001. UNIX System Administration Handbook. Third Edition. Published by Prentice Hall PTR.

OpenBSD. 2003. "OpenBSD Secure Operating System". URL: http://www.openbsd.org (December 2003).

OpenOffice. 2003. "OpenOffice: OpenSource Office Suite". URL: http://www.openoffice.org (December 2003).

Palmer, Thomas. 2001. "Basic Travel Security Revisited. SANS InfoSec Reading Room: Travel Security". URL: http://www.sans.org/rr/papers/index.php?id=410 (December 2003).

Parrish, Scott. 2001. "Security Considerations for Enterprise Level Backups. SANS InfoSec Reading Room: Backup Strategies". URL: http://www.sans.org/rr/papers/index.php?id=515 (December 2003).

PatchLink. 2003. "PatchLink Patch Configuration Management". URL: http://www.patchlink.com (December 2003).

Perera, Rick. 2002. "USB Tokens Offer Pocket-Sized Security: Small, cheap devices plug into PC's ports to identify users, but not everyone is convinced of their merits. IDG News Service, 15 March 2002". URL: http://www.pcworld.com/news/article/0,aid,89263,00.asp (December 2003).

Petullo, Mike. 2003. "Implementing Encrypted Home Directories". Linux Journal, August 2003 (issue 112), pp. 62-68.

PGP. Pretty Good Privacy. 2003. "International Pretty Good Privacy". URL: http://www.pgpi.org (December 2003).

Provos 2000: Provos, Niels, Encrypting Virtual Memory, 2000

PuTTY. 2003. "PuTTY: A Free Win32 Telnet/SSH Client". URL: http://www.chiark.greenend.org.uk/~sgtatham/putty/ (December 2003).

Raak Technologies. 2003. "T8 USB Smart Token". URL: http://www.raaktechnologies.com/products/t8.htm (December 2003).

RedHat Linux. 2003. "RedHat Linux". URL: http://www.redhat.com (December 2003).

Renfro, Scott. 2002. "Secret Handshakes & Telltale Fingerprints: Cryptographic Solutions for System Security". Linux Magazine, September 2002, (volume 4, issue 9), pp. 24-29.

RSA. 2003. "RSA SecureID: 6100 USB Token". URL: http://www.rsasecurity.com/products/securid/datasheets/dsusbtoken.html (December 2003).

Ryder, Josh. 2001a. "Laptop Security. Part One: Preventing Laptop Theft". URL: http://www.securityfocus.com/infocus/1186 (December 2003).

Ryder, Josh. 2001b. "Laptop Security. Part Two: Preventing Information Loss". URL: http://www.securityfocus.com/infocus/1187 (December 2003).

SANS NewsBites 5/40. "Canadian Tax Department Computers Stolen" (30 September 2003). SANS NewsBites: 08 October 2003, (volume 5, number 40). SANS Institute. URL: http://www.sans.org/newsletters (December 2003).

SANS NewsBites 5/43. "Judge Dismisses Data Theft Lawsuit" (21 October 2003). SANS NewsBites: 29 October 2003, (volume 5, number 43). SANS Institute. URL: http://www.sans.org/newsletters (December 2003).

SANS NewsBites 5/45. "Australian Defense Minister Enumerates Department Security Breaches" (10 November 2003). SANS NewsBites: 12 November 2003, (volume 5, number 45). SANS Institute. URL: http://www.sans.org/newsletters (December 2003).

SANS NewsBites 5/47. "Wells Fargo Offers $100,000 Reward in Computer Theft Case" (24 November 2003). SANS NewsBites: 26 November 2003, (volume 5, number 47). SANS Institute. URL: http://www.sans.org/newsletters (December 2003).

SANS NewsBites 5/48a. "Hatch Staffer on Administrative Leave After Computer Document Theft Allegations Surface" (28 November 2003). SANS NewsBites: 03 December 2003, (volume 5, number 48). SANS Institute. URL: http://www.sans.org/newsletters (December 2003).

SANS NewsBites 5/48b. "Wells Fargo Customer Data Thief Arrested" (27 November 2003). SANS NewsBites: 03 December 2003, (volume 5, number 48). SANS Institute. URL: http://www.sans.org/newsletters (December 2003).

SANS NewsBites 5/51. "Stolen Bank Laptop Contains Customer Data" (19 December 2003). SANS NewsBites: 23 December 2003, (volume 5, number 51). SANS Institute. URL: http://www.sans.org/newsletters (December 2003).

SecureInfo. 2003. "Enterprise Vulnerability Remediation [EVR], Risk Management System [RMS], and Enterprise Vulnerability Management [EVM]". URL: http://www.secureinfo.com (December 2003).

Silman, Joshua. 2001. "Steganography and Cryptanalysis: An Overview. SANS InfoSec Reading Room: Steganography". URL: http://www.sans.org/rr/papers/index.php?id=553 (December 2003).

Singh, Simon. 2000. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Published by Anchor Press.

Skoudis, Ed. 2001. Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses. Published by Prentice Hall PTR.

Starling, Gan. 2003. "PNY Attache 256MB USB 2.0 Flash Drive: Key Fob Memory Thingie Okay for NetBSD". URL: http://starling.us/gus_netbsd/gus_netbsd_pny_attache.html (August 2003).

Thach, Le Ngoc. 2003. "Re: How to stop a flash disk?". URL: https://listman.redhat.com/archives/psyche-list/2003-July/msg00047.html (August 2003).

ThumbDrive. 2003. "Memorex: USB ThumbDrives (ThumbDrive™ 256MB, Part Number: 32507256)". URL: http://www.memorex.com/products/product_display.php?cid=180&pid=505&oid=588 (December 2003).

Thunderbird. 2003. "Mozilla Thunderbird: OpenSource Email Client". URL: http://www.mozilla.org/products/thunderbird (December 2003).

Traugott, S., J. Huddleston and J.C. Traugott. 2003. "Infrastructures.org Website". URL: http://www.infrastructures.org (December 2003).

Traugott, S., J. Huddleston and J.C. Traugott. 2003. "Infrastructures.org Website: Disaster Recovery". URL: http://www.infrastructures.org/bootstrap/recovery.shtml (December 2003).

Traugott, Steve and Joel Huddleston. 1998. "Bootstrapping an Infrastructure". URL: http://www.infrastructures.org/papers/bootstrap/bootstrap.html (December 2003).

Urban, Michael and Brian Tiemann. 2003. FreeBSD Unleashed. Second Edition. Published by SAMS Publishing.

WinCVS. 2003. "WinCVS Graphical User Interface [GUI] to the CVS Version Control System". URL: http://www.wincvs.org (December 2003).

Zakrzewski, Miroslaw and Ibrahim Haddad. 2002. "Linux Distributed Security Module: The Implementation of Mandatory Access Control Through a Linux Kernel Module That Is Targeted For Linux Clusters". Linux Journal, October 2002 (issue 102), pp. 20-28.