CS 486 – Capstone Project

Project Requirements

(Revision 1.0)

Submitted to

Dr. Doerry

By

Team Fugu:

Erik Wilson

Ben Atkin

Nauman Qureshi

Thad Boyd

On

February 18, 2004

Table of Contents

1 Introduction

The purpose of this document is for our team to convey the needs and wants of our sponsor, the United States Geological Survey (USGS) Astrogeology Research Program, regarding the requirements for our project – OS Tools for OpenBSD. Additionally, this document is quintessential in that it forms the basis for all of our future work, including the specifications and resulting implementation.

The USGS was originally created to perform a unique combination of responsibilities: "classification of the public lands, and examination of the geological structure, mineral resources, and products of the national domain." Our sponsor, the USGS Astrogeology Research Program, has a rich history of participation in space exploration efforts and planetary mapping, starting in 1963 when the Flagstaff Field Center was established to provide lunar geologic mapping and assist in training astronauts destined for the Moon. The Flagstaff Field Center has been involved with several important NASA missions, including the Lunar Orbiter and Mars Rover.

To support their robust networking environment there are several dozen enterprise class servers, many of which run OpenBSD for its renowned security. However, the issue with using OpenBSD for the USGS is that the installation or upgrading of the operating system is a manual process, and any security patching must be performed manually as well. Because of the required administrator interaction the time involved with these processes becomes greater as the number systems which are present increases. To solve this problem Team Fugu proposes to automate these tasks which otherwise would be chronologically inefficient to perform, therefore we will provide the following two items:

- **Automated Installation System for OpenBSD**
- **Automated Patching Tool for OpenBSD**

Team Fugu is excited to be a part of this excellent opportunity to assist the USGS in its duties as well as the contribution to the development of OpenBSD. We are confident in our abilities to produce reliable tools which will be effective time savers.

2 Problem Statement

2.1 Astrogeology Program Background

The Astrogeology Research Program is a team of over 80 research scientists, cartographers, computer scientists, administrative staff, students, contractors, and volunteers working to support the efforts to explore, map, and understand our solar system. Fields of particular interest are mapping, planetary geologic processes, remote sensing and monitoring, and scientific analysis, which leads to answers about our neighboring planets. Throughout the years, the program has participated in processing and analyzing data from various missions to the planetary bodies in our solar system, assisting in finding potential landing sites for exploration vehicles, mapping our neighboring planets and their moons, and conducting research to better understand the

origins, evolutions, and geologic processes operating on these bodies.

These research scientists rely heavily on the Flagstaff Field Center's secure and powerful networked computing environment. They use many different combinations of computer models, architectures, operating systems, and custom applications to perform their research. The system administrators for the Center must in turn build, secure, maintain, and provide user support for dozens of computer systems. The Information Technology (IT) department of the Flagstaff Field Center has relied on, among other operating systems, OpenBSD to provide the critical network services necessary to run their systems.

OpenBSD is a free Unix variant that is well known for the security that it provides, and is therefore the preferred operating system for enterprise class servers. In addition to being more secure, in terms of the number of vulnerabilities discovered compared to other Unix variants (such as Linux), OpenBSD also provides simplicity, reliability, and performance. However, the adoption of OpenBSD by the USGS has been slower and more expensive due to several shortcomings: namely that installations or upgrades, as well as security patching, must be performed manually.

2.2 Information Technology Issues

Rather than relying on a handful of expensive monolithic servers, the IT department instead deploys a multitude of smaller inexpensive servers which when combined can perform the equivalent duties of their larger cousins. However the trade-off is that when using OpenBSD these smaller servers require manual installation and patching. The time involved with performing a manual installation is prohibitive to the system administrator when the process occurs repeatedly. Patching a server also requires gaining special skills in the form of thoroughly understanding the patching system, which is unnecessary.

2.3 Value of a Solution

Creating an automated installation and patching system will save many person-hours of time. Installing OpenBSD on a group of systems should take the time it requires to create a configuration file and boot the installer, rather than an hour per node. Instead of spending hours patching systems the administrators will only need to spend the occasional few minutes checking email to ensure that the automatic patching occurred successfully. The amount of time it taken for manual installation and patching is as follows:

(Equation 1)

$$\frac{(1\,hour\,install + (\frac{(1\,hour\,patching)}{month}) * 12\,months)}{machine} = \frac{(13\,hours/year)}{machine}$$

Repeat this manual process for fifty machines and the equation becomes:

(Equation 2)

$$\frac{(13\,hours/year)}{machine} * 50\,machines = 650\,hours/year$$

Assuming the average system administrator is paid $28 per hour the cost per year for fifty machines using manual processes becomes:

(Equation 3)

$$650\,hours/year * \$28/hour = \$18{,}200/year$$

Using an automated installation for fifty machines would result in the following equation:

(Equation 4)

$$2\,hour\ setup + \frac{(\frac{(5\,minutes\ patch\ verify)}{month} * 12\,months)}{machine} * 50\,machines = 52\,hours/year$$

Again assuming the average system administrator is paid $28 per hour the cost per year for fifty machines using automated processes becomes:

(Equation 5)

$$52\,hours/year * \$28/hour = \$1{,}456/year$$

The resulting monetary savings for using an automated installation is obtained by taking the difference between the manual costs and the automatic costs for fifty machines, this is as follows:

(Equation 6)

$$\$18{,}200/year - \$1{,}456/year = \$16{,}744/year$$ **total savings for 50 machines!**

In addition to saving time with installations or upgrades, the peace of mind in the uniformity of security maintenance is invaluable. There is no need to create a checklist of patched systems, and there is little possibility for human error to occur in the process. This creates a virtually "hands free" solution for an otherwise complicated processes.

2.4 Competitive Products

There currently aren't any tools that provide a flexible automated installation system for OpenBSD, although there is one in the planning stage called “BUMPSTART” which can be found at *http://sourceforge.net/projects/bumpstart/*. There does however exist a fairly good patch-management tool, called “Tepatche” (located at *http://www.gwolf.cx/soft/tepatche/*), but it needs a number of improvements to meet our sponsor requirements (especially the addition of the ability to patch from binaries). Because Tepatche performs most of the basic functionality required for our automated patcher we will modify it to suite our needs.

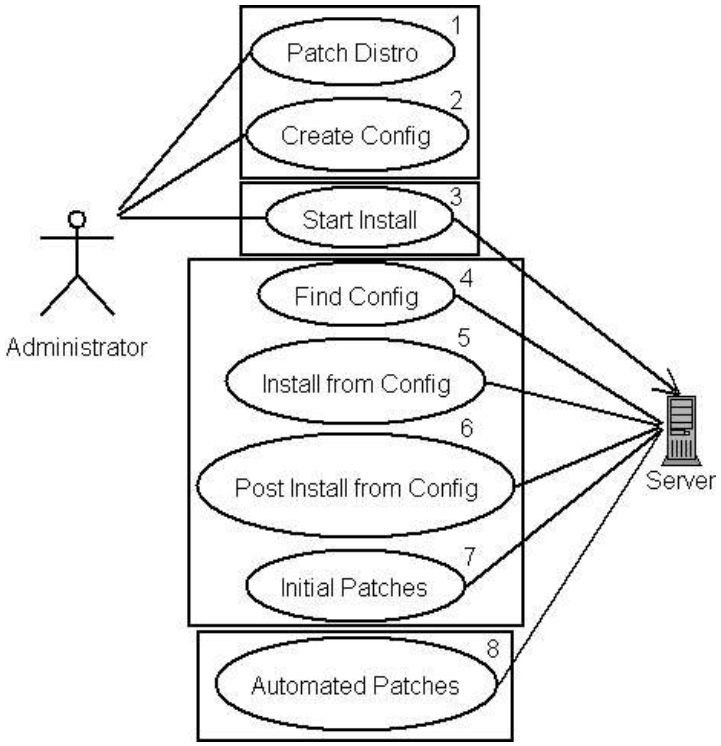
2.5 Business Environment

In general regarding server operating systems Unix is becoming more popular and Microsoft is becoming less popular. The need for automated installation and patching is always increasing in today's world. Remote administration is taken to the extreme and often occurs inter-continentally, often times where the capability to ship pre-built machines is unavailable. The need for our product can easily be found by performing the following simple Google search for 'openbsd automated installation':
http://www.google.com/search?q=openbsd+automated+installation

3 Solution Statement

The following Use Case Diagram depicts the high-level processes that will occur in order to use our implemented solution (details are provided on the next page):

(Figure 1)



Details for each step of the solution are as follows:

1. Modifying the current manual installation and upgrade system will create the automated installer.

2. For a class of machines the system administrator will create a configuration file.

3. The system administrator boots the new distribution. Initially, when the installer is booted there will be approximately a five-second timeout before the installation begins. During this period the user will be able to press a key and drop to a console, or do an interactive install.

4. If the automated installation isn't interrupted, the installer will search for the configuration file from a variety of locations, and then parse that file.

5. From this configuration file the installer will be able to partition the disk, format the partitions, configure networking, and install packages.

6. The automated installer will run a post-install script that is obtained from the configuration file.

7. Initial patches will be installed and the system will reboot into the newly installed or upgraded OS.

8. The patcher will be a regularly scheduled task, as opposed to a daemon process, to save system resources and take advantage of the task-scheduling mechanisms already in place. Most likely, it will be run both as a *cron* job and using */etc/rc* scripts, so updates will not be missed if a computer is shut off. It will be highly configurable, allowing for either standard source patches from the OpenBSD FTP site (and mirrors), or custom binary patches from the local network.

4 Requirements

4.1 Project Goals

Upon completion of the project, the following will be provided:

1. The Automated Installer will work by booting from any media and will run on any platform. The installation should handle partitioning disks, creating file systems, configuring the network, installing software, and any other task currently handled by the interactive install. It should also allow pre and post install scripts for additional software and hardware setup. As part of this tool the following will be provided:
    a. Scripts that can a create floppy and CD disk images with a given installer configuration file.
    b. A sample configuration files to provide a basis for creating new configurations.
    c. An interactive configuration file generating utility may be provided if time permits.
2. The patch management tool will be capable of installing binary or source patches from a given URL. This will most likely be derived from Tepatche, which already provides a good mechanism for downloading and building source patches. The patch management tool will, like the installer, be able to handle a configuration file made for a class of systems. As part of this tool the following will be provided:
    a. A package (compressed tar file) containing all of the source code for the patch manager.
    b. Scripts that will install the patch management tool. Once installed it will check for and install patches, as well as be regularly scheduled to for the patching process.
    c. An interactive configuration script may be provided if time permits, allowing the user to configure the patcher for such things as specifying an URL for download of the patches
3. Internet documentation for the two above systems, which will also be provided in such forms as a man page or README file on the system.

4.2 Functional Requirements

4.2.1 Auto-installer Requirements

Team FUGU understands the automated installation tool should fulfill the following functional requirements:

1. **Must be substituted into existing OpenBSD installation environment.**
   - A tool will be provided to modify the existing source distribution, which when compiled would provide automated installation functionality.
   - If possible a patch will be provided to modify ISO and floppy installation images.
2. **Must be able to handle either clean install or upgrade and existing OS installation.**
   - Installation type can be specified in the configuration file.
   - All preexisting functionality of installation or upgrade will be maintained.
3. **Must read an installation configuration from multiple sources.**
   - The configuration will contain sections related to specific steps of the installation process.
   - Installation will check floppy and CDROM.
   - Installation will check FTP and HTTP servers.
   - Machine specific configuration files will be searched for.
   - If multiple installation files exist a well-defined precedence will be used to determine which sections of the files to use.
4. **Must be able to handle all the installation steps currently performed by the manual installation process.**
   - The order for some steps, such as configuring the network, may be re-ordered to best facilitate the installation.
   - Extra functionality for systems such as partitioning the disks may be provided.
5. **The tool must also handle pre and post installation scripts.**
   - The scripts themselves or locations to these scripts may be provided in the configuration file.
   - These scripts must provide the capability for preexisting files to be transferred across the network or between partitions.

4.2.2 Auto-patcher Requirements

Team FUGU also understands that the automated patch tool should fulfill the following functional requirements:

1. **Arguments can either be passed in on the command line or set in a configuration file.**
    - The configuration file will provide all information needed to perform the patching.
    - Optional command line arguments are available to over-ride the defaults located within the configuration file.
2. **It must handle both source and binary patches.**
    - Locations for source and binary patches will be maintained independently of each other in the configuration file.
    - The correct architecture for binary patches will be selected for download.
3. **Binary patches must be able to run a pre-install, post-install, pre-uninstall and post-uninstall scripts, and contain install and uninstall processes.**
    - In the case of scripts these will be provided within the script itself.
    - Install and uninstall functionality will be provided by the patching system.
4. **The patch system must keep track of what patches have been installed.**
    - This may be maintained within a plain text file per machine, or residing on a network file system.
    - Location of database is determined in configuration file.
5. **The system must be able to send an email, using the standard UNIX mail system, to the system administrator(s).**
    - Email may be defined to be sent only when patch fails or also when successful patching occurs.
    - Email settings are defined in configuration file.
6. **A highly desirable feature is that the tool be able to run in the installation environment.**
    - Will occur prior to the post-installation script process.
    - Scheduling for patcher may be defined in installation configuration file.

4.3 Performance Requirements

Team FUGU understands the OS tools should fulfill the following performance requirements:

1. **User interaction should involve as minimal amount of time as possible.**
   - An administrator must download, patch, or rebuild an installation image.
   - An administrator must create configuration files for a class of machines.
   - Once the installation process begins there should be only a nominal amount of interaction.
2. **There is no requirement for interactive processes to configure the automated installation or patching systems.**
   - These servers are used within an enterprise environment, usually without the use of an X-Windows system.
   - Qualified system administrators are capable of configuring the system without the aid of an interactive process.
3. **There is no requirement for the amount of time the automated installation process must occur in.**
   - Installation time is dependent on the speed of the processor.
   - Installation time is also dependent on the speed of the network.
4. **There is no requirement for the amount of time the automated patcher must apply the patches.**
   - Patching time is dependent on the speed of the processor.
   - Patching time is also dependent on the speed of the network.

4.4 Constraints

One of the main advantages of OpenBSD is that it is easily installed and can run on legacy systems. The ramdisk installation method requires a minimal amount of memory to load the ramdisk (approximately 8 megabytes), however there are other installation methods to circumvent this requirement. OpenBSD can run on Alphas, HP300 (and above), Intel's i386 (and above), and many other such architectures. This takes care of most of the hardware constraints. There are no other form of constraints applicable other than the ones mentioned above.

4.5 Business Philosophy

To encourage future improvements, we are creating this software under the revised BSD license. This license states all the rights that Team Fugu has under which the development of automated tools is being done. Redistribution of source code must at all times bear the copyright notice of Team Fugu. Also the redistribution of the code in binary form should also bear at all times the stated copyright notice. When products derived from this software are promoted and endorsed with the names of the authors, permission prior to that is strictly enforced and suggested.

5 Feasibility

5.1 *Economics*

Our project is expected to save hundreds of person-hours over a period of months. The bottom line in our project is that it will allow for the installation of multiple copies of OpenBSD without human input. Ideally, where an administrator would have previously had to work an hour on each installation, he may now spend an hour on a single configuration file to be used for all installations.

5.2 *Technical Feasibility*

We intend to build our project based on existing open-source OpenBSD code, under the Revised BSD License (see 5.5, legal feasibility). In some places, we expect to rewrite code from the ground up, and in others we we will merely build on existing code. We intend to use existing and freely available libraries and sources for as many tasks as possible.

5.3 *Risk Analysis*

Risk	Description	Probability	Severity	Mitigation Strategy
1. Does not perform to standards	Tools do not perform their tasks correctly. For example, installer does not partition disk correctly or patching system does not update correctly.	Low	Critical	Test thoroughly. Make absolutely sure that programs perform to specifications.
2. Interface unusable	Tools are inconvenient and actually increase operation time rather than save it.	Low	Critical	Test early and often. Communicate with sponsor and send test versions for feedback.
3. Cannot back up existing files	Installation procedure is only suitable for completely wiping systems and is not equipped to back up important files (SSH keys, etc.) where necessary	Medium-low	High	Begin research on this as soon as possible and ask sponsor for specifics. This should be a trivial bit of coding as long as we know which files we need to look for and back up.

Risk	Description	Probability	Severity	Mitigation Strategy
4. Programs not fully automated	Tools save operation time but still require user input.	Low	High	Write code to operate automatically first and worry about accepting user input later.
5. Limited or no hands-on functionality	Tools are completely automated and do not allow user input.	Low	Medium	Write automatic code first as it is the priority, but code expecting to add user input overrides and to make doing so as easy as possible.
6. GUI incomplete	Tools run only from command line, no GUI is available.	Medium	Low	As the primary purpose of our project is to create hands-off functionality, GUI generation, while potentially useful in some cases (e.g. scheduling for patching program), is our bottom priority. We will only begin work on a GUI after all project requirements have been satisfied, and if no GUI is completed, our project can still be considered a resounding success.

5.4 *Resource Availability*

We intend to build these tools based on existing code, freely available libraries, and our own expertise. This is an open-source project and will use only open-source code and tools.

Our tools must run on Intel i386 and Sun Sparc64 machines. Officially, we will use the Intel Pentium and Sparc Ultra as our minimum systems, as these machines are much lower-end than the systems we expect our software to run on, but as our tools are console-based they do not require sophisticated machines and we expect they would run effectively, though very slowly, on much older legacy systems.

Each team member is to have an i386 test machine at home, and the team requires several test machines, both i386 and Sparc64, in the CET building. Test machines are being provided by the CET IT Department and by USGS where necessary.

5.5 *Legal Feasibility*

We are confident in our ability to produce a piece of software which is versatile and powerful enough to become widely used in the OpenBSD world. As OpenBSD developers

tend to be hostile toward the GNU GPL (General Public License), we intend to license our software under the Revised BSD License (Appendix A). This precludes using any GPL code in our project, as any code derived from GPL works must itself be licensed under the GPL.

Appendix A: Revised BSD License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.