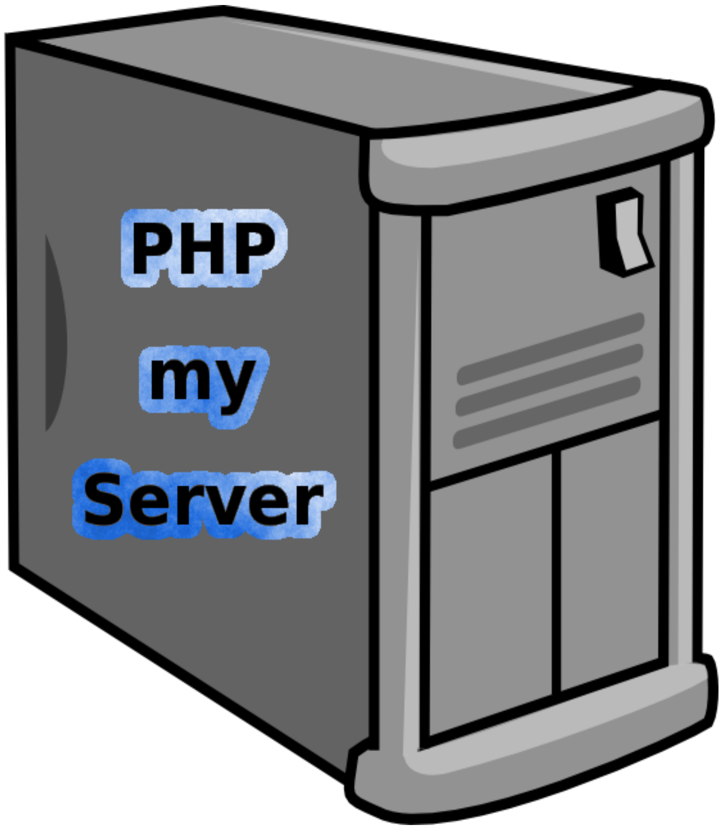
PHPmyServer

1.0 Release!



Author: Robert Nicholson

A free as in "Freedom" server control application for *nix based servers, running on PHP and Apache

Date: 10/8/09

"Final Report"

Table of Contents

Application Development

Project Description:

Edited as of December 8, 2009

PHPmyServer is a multi-platform server diagnostic utility for *nix based machines.

PHPmyServer allows remote status checks and some baseline commands such as restart, shutdown, and killing processes; more advanced features have been enabled through the extension of shell scripts. Administrators can create their own scripts to be executed through the Custom tab.

Currently, server administration software is specific in that usually a third party program must be installed on the machine the administering will be happening from. PHPmyServer allows an administrator to check on the status of servers and issue critical tasks by using a PHP interface to authorize the administrator, and then allow him or her to execute commands on a superuser level with minimal hassle and from any web-enabled machine.

Motivation:

I am currently aware of no software that allows an administrator to check up on his or her servers from any web-enabled machine, without fear of the password for the network or some other issue cropping up. This of course won't protect against key loggers or truly malicious individuals, however, a good administrator will know whether he or she should use a particular machine for such sensitive tasks.

What I would like to happen, is for PHPmyServer to use ssh and hashing to make sure that the administrator is really who he or she says that they are, and then allow only certain superuser level commands to be issued, and for others to be unavailable.

Ultimately I would like this to become a useful interface for handling server functions for anyone who has a *nix based machine that needs to make certain its always performing at its maximum. Possibly in the future I would also like to support the Windows platform, but this would require a lot of extra work, as the administrator would have to create custom scripts to allow the *nix commands to translate to compatible Windows commands.

Resources:
PHP
HTML
CSS
Apache Web Server
BASH (Born Again Shell)
Linux OS
http://www.w3schools.org/

Software Requirement Specifications

Number: 1
Statement: Web Based, Standards Compliant Interface
Evaluation: Interface can be accessed from any major web browser: Firefox, Internet Explorer, or Opera.
Depends Upon: None
Priority: Essential
Requirement Revision History: Appropriately formatted October 12th.

Number: 2
Statement: PHP Compliant Script Handling
Evaluation: The software must be able to handle PHP evaluations through a terminal.
Depends Upon: 1
Priority: Essential
Requirement Revision History: Appropriately formatted October 12th.

Number: 3
Statement: Reporting Uptime of Host Machine
Evaluation: Accurately reporting the uptime to the web interface of the host machine.
Depends Upon: 2
Priority: Essential
Requirement Revision History: Appropriately formatted October 12th.

Number: 4
Statement: Checking Processor Load of the Machine
Evaluation: Correctly reporting the processor load to the web interface of the host machine.
Depends Upon: 2
Priority: Essential
Requirement Revision History: Appropriately formatted October 12th.

Number: 5
Statement: Checking the Network Traffic of the Machine
Evaluation: Correctly reporting the network load to the web interface of the host machine.
Depends Upon: 2
Priority: Essential
Requirement Revision History: Appropriately Formatted October 12th.

Number: 6
Statement: Authenticate a Super User access through PHP to the Host Machine.
Evaluation: Correctly allow or deny access securely to the host machine.
Depends Upon: 2
Priority: Essential
Requirement Revision History: Appropriately Formatted October 12th.

Number: 7
Statement: Issuing a Reboot or Shutdown Command to the host machine.
Evaluation: The host machine correctly accepts and processes the command.
Depends Upon: 2
Priority: Essential
Requirement Revision History: Appropriately Formatted October 12th.

Number: 8
Statement: Allow parsing of custom shell scripts from a folder on the host machine.
Evaluation: Correctly controlling access to the shell scripts and executing them via the host machine's shell.
Depends Upon: 2
Priority: If Time Permits
Requirement Revision History: Appropriately Formatted October 12th.

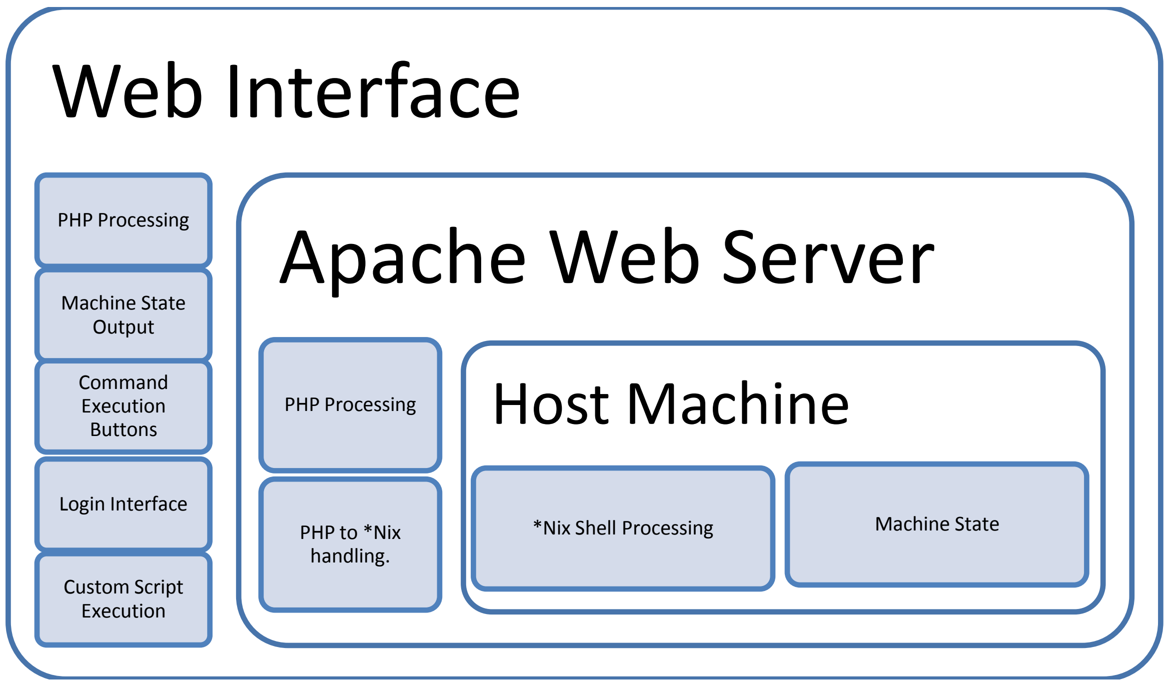
(Edited October 12th.)
Vision:

PHPmyServer will provide network administrators a quick, and easy way of checking on servers and rebooting them from a distance without hassle. People who depend on these servers for daily tasks need not fear simple overloads and minor problems interfering with their work flow. Now all the hassle of checking up on minor problems can be done from a distance to save time and money.

(Edited October 12th.)
Scope:

PHPmyServer will include a slick interface to server states and a secure method of executing a shut down or reboot command from any web enabled computer with a standard browser. If time permits, it will even allow custom commands to be executed via scripting, but is primarily a tool of diagnostics and not a full maintenance utility.

System Design and Architecture



The above nested diagram shows the layout of the PHPmyServer overall design. Functionally, the program will be a PHP Script. In totality, it will require a smooth cooperation between the Web Browser, Apache Web Server, and the Host *Nix machine.

The administrator will login. PHP will process the login through the host machine through string handling and a form of encryption not yet decided upon. The vital statistics of the machine will be updated via PHP executing a built in *Nix command or series of commands such as TOP or UPTIME. The host machine's terminal will display the information and PHP will return the information to the web browser, where it will be formatted and displayed to the screen for the administrator to see.

Implementation (List of Files Included and Basic Overview)

Index.php: Main php page. This page contains the menu and the introduction statement, as well as some helpful info in the comments.

Login.php: Used to check to see whether the user was Admin, deprecated. Now the authentication is left up to the user.

Memory.php: Memory script readout page. This page displays the host machines current memory load in an up to the second manner.

Resources.php: This page reads out the current physical storage space and any devices and what nodes those devices are mounted on.

Rootaccess.php: This reads a log file and scans it for all root accesses.

Styles.css: This is the cascading styles sheet file.

User.php: This checks the host machine for the current users logged in and displays them.

Dmesg.dump: This is the temporary file used for dumping all of dmesg so we can search it for what we want.

Custom.sh: This is the example custom script.

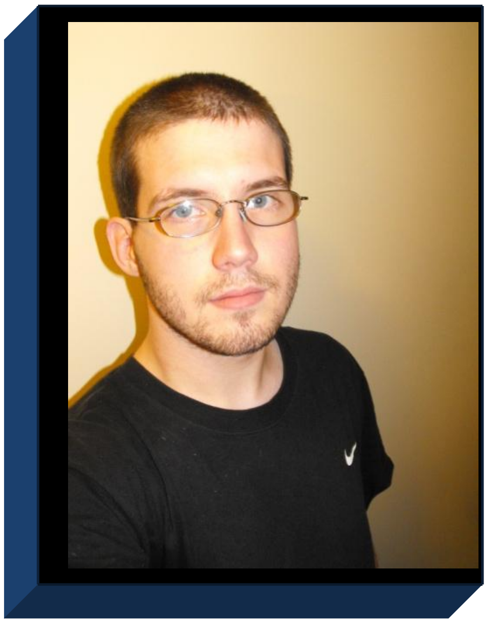
Custom2.sh: This is a second example script.

Custom.dump: This is an example dump file, in case a command generates too much garbage to sort through on the fly, users can see how to dump the info and then search it.

Custom.php: This is the page that calls custom.sh and uses that script to generate output based on the contents of custom.dump.

Commands.php: Some predefined php functions I created to make coding easier.

Auth.log: a hard link to the systems Auth.log file for active searching. You should delete this, as it will not work on your machine and may cause errors.



About the Developer

Robert Nicholson is a Computer Science major at Berea College. Now a senior, Robert enjoys the challenges that come from the everyday life of a college student. This has provoked an interest in the aspect of interface and security for him, ultimately leading to this project.

Robert’s goals within the field of Computer Science include becoming a Systems Engineer and consultant for enterprise level networks, and to work on the human-hardware interface level of Computer Science to make the technology more accessible for any individual, without limiting functionality. Particularly, touch screen, vocal recognition, and optical biometrics are of interest to him, as well as direct brainwave manipulation of hardware, and vice-versa.

Less Technical Stuff

Interests Outside of Computer Science: Practical Martial Arts (Law Enforcement, Personal Defense, Health, Etc…), Ballistics, Photo Editing, Hiking, Camping, Neurochemistry (Especially involving the endocrine system and neurotransmitters such as serotonin and noradrenalin and adrenaline.), Native Cultures (Religious customs and ideologies, anything with a story.), Biking, Amateur Photography, Art (Paintings, sculptures, all of it.), Music Festivals, (Terrapin Hill, Burning Man, Bonnaroo), Good Science Fiction (Heinlein, Asimov, Bradbury, Clarke etc…)

Contact Info:

Phone	(859)985-6218
Email	bobbynicholson@gmail.com

Test Plan

Name	Proper PHP Processing
Requirement	None
Preconditions	Apache is installed and configured on the host machine. PHP is installed and configured on the host machine.
Steps	1. Go to 127.0.0.1 on the host machine's web browser. 2. The page should display the interface, which is in PHP
Expected Results	The web browser displays the interface correctly, if PHP is not enabled and configured properly, the program cannot possibly function.

Name	Standards Compliant Web Programming
Requirement	Proper PHP Processing
Preconditions	The root /www folder must be set up correctly. Apache and PHP must be installed. The phpmyadmin.php file should be in /www The machine is equipped with a standards compliant browser.
Steps	1. Go to 127.0.0.1 on the host machine. 2. Visit the page in Firefox, Internet Explorer, Opera, and Chrome
Expected Results	The page will be displayed correctly in all four major browsers, and any new browser that is standards compliant.

Executive Section

Week 2-3 Memo:

During week 2 and 3, no progress was made other than the finalization of the platform, and the scope of the project. I was ill, and was in no shape to be working on the project. This document was updated, and the overall direction of the project was defined more thoroughly.

For the following week, implementation of some sort of user login credentials will be in development, as well as the possibility of issuing superuser commands through the PHP interface, and whether that interface can be encrypted without crippling accessibility.

PHPmyServer
Date: Tuesday, October 13, 2009
Author: Robert Nicholson
Project Total Time: 20 Hours

The past two weeks have been largely devoted to revision of the SRS, Vision and Scope, and learning PHP and how it interacts with the host web server. Here is the breakdown:

Studying PHP:	4 Hours
Using PHP to Issue Commands to a Host Machine:	4 Hours
Formatting Output Returned from Issued Commands:	2 Hours
Reformatting Project Documentation	3 Hours

I’ve spent about 13 additional hours these past two weeks on the project itself. Overall things are going very well, and I expect the time use to increase as the project goes along and the actual construction of the code and testing SRS components begin. I would like to complete all SRS components and have time to implement the Scripting function, which would greatly improve the functionality of the program itself, but I feel that to be out of the scope unless I am certain it will fit into the time frame of the project.

In the following week I would like to have a developed schema as to exactly what I would like the authentication system to function like, whether it will use hashing or encryption and exactly how secure versus usable do I want it to be. Ultimately I would like to have the authentication system finished by the end of next week.

PHPmyServer
Date: Tuesday, October 20, 2009
Author: Robert Nicholson
Project Total Time: 26 Hours

This past week has been devoted to the use of PHP and working with Sessions. Currently a weak grasp of PHP Sessions and coding practice is understood, but strong headway has been made in the area of PHP variable handling and sessions. The next step will be to break the PHP code down into functions, clean up the code, and implement encryption for authentication of sessions, and logging into the host machine.

PHP Session Coding (index.php)	4 Hours
Report Maintenance	2 Hours

The following week should be significantly more productive, and authentication should be completed.

PHPmyServer
Date: Tuesday, November 3, 2009
Author: Robert Nicholson
Project Total Time: 39 Hours

For the last week I have began to restructure the underlying elements of PHPmyServer. What this means is, I have decided to recode the PHP functions into a separate php include file which I plan to use to keep the code clean and separate from the general html I will use to stylize the page once the code is complete. I have also nearly completed the login system, which will generate an MD5 hash based on a password located on the server and a randomly generated number combined with the user's IP address. This combo will ensure that only the administrator is accessing his or her machine, and also avoid any unnecessary complexity associated with having a database just for authentication purposes.

The structure will look a little like this:
On the Guest machine
MD5(IP Address + Randomly Generated Key (Sent to user by Server) + Password)
= Guest_Unique_ID

On the Host machine
MD5(IP of Guest + Randomly Generated Key (Generated by Server) + Password)
= Host_Unique_ID

If the Host_Unique_ID = Guest_Unique_ID the accessing account will be given full permissions, and the sudo command suite will be unlocked.

Time Spent on PHP and Restructuring: 6 Hours
Time Spent on Report: 3 Hours

By next week I would like to begin working on the interface design, and hopefully implement some basic server control commands such as showing directory contents and displaying processes running on the machine.

PHPmyServer
Date: Tuesday, November 10, 2009
Author: Robert Nicholson
Project Total Time: 46 Hours

This week was spent struggling with getting the PHP to properly cooperate with the terminal interface. User authorization functions, and is control is passed off appropriately, however, the ‘su’ or ‘sudo’ commands take one or more arguments in order to be ran, and the shell( ) function in PHP seems to only pass commands as a single argument and then kill the connection. I have to find some way to circumvent this and authenticate as root on the host machine, or I may have to write a supplementary C++ program that executes and takes a single string argument, breaks it up, and then executes it in the shell. That way I could catch the shell( ) PHP function with the C++ program like a net, break up the arguments in the string, and pass them one by one as god intended.

Struggling with PHP and BASH	7 Hours
Looking for a way to authenticate as root	2 Hours
Banging my head against the wall	Always

PHPmyServer
Date: Tuesday, November 17, 2009
Author: Robert Nicholson
Project Total Time: 57 Hours

This was a frustrating but fairly productive week. I’m still a little behind. Thankfully the testing phase will be much shorter for my particular project, and I can get that knocked out over the Thanksgiving break. Success has been had! There is no interface to speak of yet, but HTML and PHP are quite easy to generate a nice, useable and attractive interface that will give the user plenty of options. As for the project itself, the software will now correctly format text to be displayed and can execute a series of commands to enable rebooting, network traffic reports, processor load, etc…etc… All through the super user interface after being verified via JavaScript and PHP. This was a big hurdle, and although the project doesn’t have a ‘face’ so to speak just yet, the really meaty bits are all together and working like a dream. At this point, all that’s left to do is slap a pretty paint job on it and test the hell out of it. Then videos, posters, etcetera. Most folks won’t care about this that isn’t interested in administering machines, but the ones who do; I want to take it seriously.

Getting PHP to pass commands correctly to a shell:	7 Hours
Getting the login info to store in a session cookie:	2 Hours
Preliminary layouts for testing and interface design:	2 Hours

This is exciting. While I have no interface yet, that is the next step, and essentially icing on the cake where I’m concerned. At this point, I have a fully functioning web enabled interface to any *nix machine. Making the project open source will allow this to really take off. I hope to see someone adapting this to Windows platforms, and others. If things go smoothly with the interface and testing during Thanksgiving, I should be able to implement the custom scripting abilities as well. Cross your fingers.

PHPmyServer
Date: Tuesday, December 8, 2009
Author: Robert Nicholson
Project Total Time: 70 Hours

The day has finally come. The last bit has been updating documentation and uploading the video, polishing up the interface and getting ready for the show.

Time Spent on Video	3 Hours
Time Spent on Poster	2 Hours
Time Spent on Final Touches	8 Hours

I think I'd like to accomplish some sleep next week.

Link to software demo: http://www.youtube.com/watch?v=pXsG2NFkVRc

README and CodeBase are inside the zip file this file is in.