

# Ragnarok Stopper: development of a vaccine

By Administrador

Published: 2020-05-04 · Archived: 2026-04-05 19:34:06 UTC

The field of reverse engineering and specifically malware analysis within the [Incident Response](#) process is of vital importance. Beyond the analysis of logs, events, network connections, alerts generated by IDS and firewalls, etc., experience tells us that a preliminary and quick analysis of a suspicious binary (whenever possible) can offer high-value intelligence, not just to get more context about an incident (TTP, C2, persistence, timeline, etc.) but to develop tools or techniques to mitigate a campaign still underway. The following case is an example of this.

In a recent incident we obtained a binary packaged with **Ragnarok**, a malware widely used in the last months in [various campaigns](#). The aim of this post is to describe a possible **vaccine** that can stop the threat posed by some of the samples that we have analyzed so far.

## Developing the vaccine

Just before harmful actions start (basically encrypting files with RSA 2048 + AES) the malware computes a fingerprint based on 5 values:

- MachineGUID
- Product Name
- Username
- Hostname
- A string concatenating the previous four elements

For each of these values, an 8-byte ID is generated, which are joined by a “-“. The fingerprint obtained (a string with the following form: XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX) is used to create an event object via `CreateEventW` with which to check if an instance is already running.

This sort of actions is very common in malware to guarantee the execution of a single process, generally with Mutex and Event objects. What is interesting is that these checks are very useful for creating [vaccines that prevent malware from running](#). The following screenshot shows the logic previously described.

```

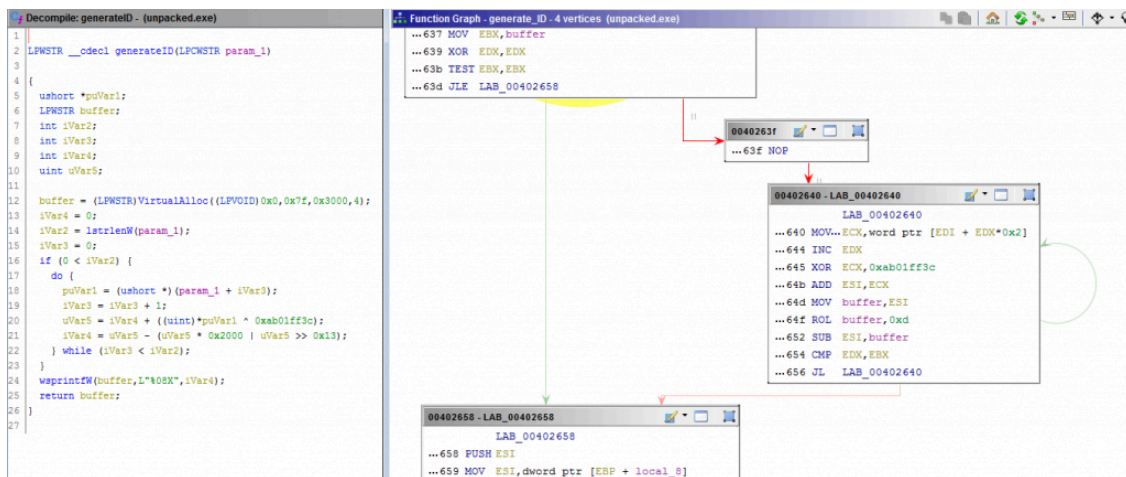
136 GetComputerNameW(hostname, &local_5c);
137 GetUserNameW(username, &local_60);
138 lstrcpyW(local_17f0, L"SOFTWARE\\Microsoft\\Cryptography");
139 machineGuid = (LPCWSTR) FUN_00402590(L"SOFTWARE\\Microsoft\\Cryptography", L"MachineGuid");
140 productID = (LPCWSTR) FUN_00402590(L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion",
141                                 L"ProductName");
142 lstrcpyW(all_concat, machineGuid);
143 lstrcatW(all_concat, productID);
144 lstrcatW(all_concat, username);
145 lstrcatW(all_concat, hostname);
146 hostname_ID = (SC_HANDLE) generateID(hostname);
147 username_ID = (LPENUM_SERVICE_STATUSA) generateID(username);
148 machineGuid_ID = generateID(machineGuid);
149 productID_ID = generateID(productID);
150 concat_ID = generateID(all_concat);
151 pcVar10 = wsprintfW_exref;
152 wsprintfW(bot_ID, L"%s-%s-%s-%s", machineGuid_ID, productID_ID, username_ID, hostname_ID, concat_ID);
153 machineGuid_ID = GetCommandLineW();
154 local_30 = 0;
155 local_20 = CommandLineToArgvW(machineGuid_ID, &local_30);
156 if (local_30 == 1) {
157     iVar9 = 0;
158     do {
159         hObject = CreateEventW((LPSECURITY_ATTRIBUTES) 0x0, 1, 0, bot_ID);
160         dwBytes = GetLastError();
161         pcVar10 = wsprintfW_exref;
162         if (dwBytes != 0xb7) break;
163         CloseHandle(hObject);
164         if (iVar9 == 0x8000) {
165             uExitCode = 0x29a;
166             hObject = GetCurrentProcess();
167             TerminateProcess(hObject, uExitCode);
168         }
169         iVar9 = iVar9 + 1;
170         pcVar10 = wsprintfW_exref;
171     } while (iVar9 < 0xfae9);

```

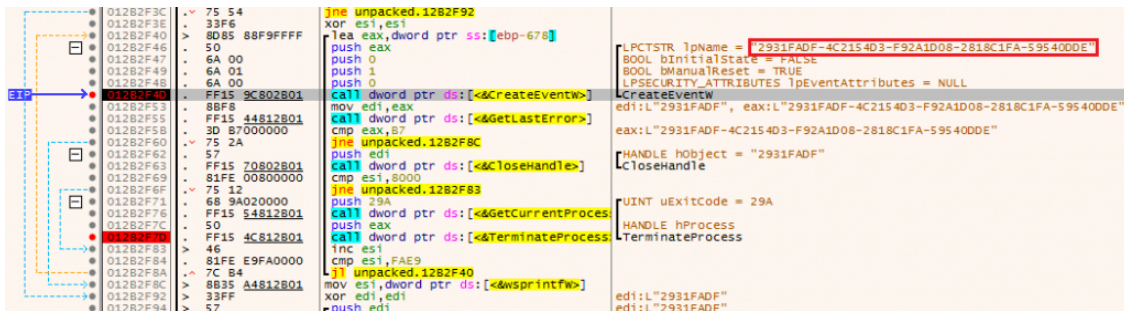
### Ragnarok running instance check

In this case, note that once the identifier is generated, it is used to create an event object with that name. If the returned handle belongs to an existing object ( `GetLastError() == ERROR_ALREADY_EXISTS` ) it retries it for a total of 0x8000 times. If this counter reaches this value, `TerminateProcess()` is invoked, thus avoiding the destructive actions of the [ransomware](#).

With this information creating a vaccine that forces the ransomware to finish its execution is trivial. It would be enough to create an event object whose name follows the format used by Ragnarok. The following function shows the logic used to generate the ID from each string.



### Fingerprint generation



### Fingerprint generation

A cleaner version of the `GenerateID` function is shown below:

```
#define __ROL__(x, y) _rotl(x, y)
inline unsigned int __ROL4__(unsigned int value, int count) { return __ROL__((unsigned int)value, count); }
inline unsigned int __ROR4__(unsigned int value, int count) { return __ROL__((unsigned int)value, -count); }

LPWSTR GenerateID(LPWSTR lpString)
{
    unsigned int inc;
    int len, i, acu;
    LPWSTR codeID;

    inc = 0;
    codeID = (LPWSTR)VirtualAlloc(0, 0x7Fu, MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);
    len = lstrlenW(lpString);
    for (i = 0; i < len; inc = (acu ^ 0xAB01FF3C) + inc - __ROL4__((acu ^ 0xAB01FF3C) + inc, 13))
        acu = lpString[i++];
    wprintfW(codeID, L"%08X", inc);
    return codeID;
}
```

The code of the PoC Ragnarok Stopper can be [found in our repository](#). It should be noted that this is just a vaccine that applies to some of the samples that we have analyzed so far. New variants of Ragnarok may use other criteria to identify a running instance.

<https://youtu.be/xUC07prttZs>

Source: <https://www.tarlogic.com/blog/ragnarok-malware-stopper-vaccine/>