

Forum: [Hack Forums](#)

Thread: [Extracting API keys from software](#)

Ash Ketchum - May 13, 2020, 12:16 am

Extracting API keys from software

Operational security is important even in software, hardcoded credentials or keys can lead to unintended consequences. How many of us have used Sysinternals? There is a variety of tools in Sysinternals, one of which is procexp, which is "Process Explorer". I must have run this software 100's of times yet there was something which I noticed, a functionality it used VirusTotal's API to submit hashes and check if currently running processes were malicious.

A bit of a foreword on VirusTotal, if you plan on using the API then you will need a key generally these can be free although limited, companies can apply for keys in order to allow multiple queries or even allow for downloading of samples. These are expensive though and not many of us have 10,000 dollars (US) to spend on checking VirusTotal or other fancy things.

I went to explore this functionality of procexp, I fired up my Windows 7 VM I ran fakenet wondering if perhaps it was retrieving a key from Microsoft, after using Wireshark to parse through the capture there was nothing related to retrieval of a key, instead it tried to contact VirusTotal's api. This gave me my first indication that the api key was hardcoded in procexp, also the simple fact that I was never prompted for a VirusTotal key helped cement this idea.

So my first step was to fire up Cutter, a free gui version of radare2 and load procexp. I began my search in the strings looking for "vt" or "api", vt turned out to yield a few hits. I began by following the XREF's for each time "vt" is referenced from there I would look over the function, it was not a large number of strings, functions or XREFs, so following the trail was not too hard. Pretty soon I stumbled upon this:

<https://ibb.co/CpcBVhw>

```
0x00637127    add byte [0x32303233], dh
;-- "4e3202fdb6953d628f650229af5b3eb49cd46b2d3bfe5546ae3c5fa48b554e0c":
0x00637128    .string
"4e3202fdb6953d628f650229af5b3eb49cd46b2d3bfe5546ae3c5fa48b554e0c" ; len=65
0x00637169    add byte [eax], al
0x0063716b    add byte [eax], al
0x0063716d    add byte [eax], al
0x0063716f    add byte [ecx + 0x70], ah
;-- "apikey=":
0x00637170    .string "apikey=" ; len=8
```

I had seen VT api's a few times I was confident I was in the right place. So a quick firing up of vt-cli we can check if this works:

<https://ibb.co/djpJHVH>

I got an odd message saying user was inactive, but uploads did work so I am guessing that while this key can make a large number of requests, such as uploading it is still limited. The use of API's

to streamline sites is common place and if an application with hardcoded credentials or keys it is possible to get them.

Credits to cyberstrike for helping me format this guide

Panagiota - May 16, 2020, 10:28 am

That was a very interesting find! Thanks for sharing it with us.

By the way, what kind of solution would you suggest to software developers in order to protect such api keys?

CyberStrike - May 19, 2020, 1:27 am

Hard coding credentials be it an ip or an api key is common place. Often though we can use built in encryption programmatically, but during run time this can be dumped unencrypted in memory, same thing with obfuscation. Using ROP gadgets one can obfuscate in memory and allow execution at a specific time. Have you seen Gargoyle project? Mixing this in with maybe a query to get the key for example using tls/ssl might be an option as well, although it'd just be more work and slow a determined person. BTW this was done while the program was not even being executed, had there been obfuscation or encryption then it may have slowed down OP.