Forum:Hack Forums

Thread: FileDestroyer Secure file removal TOOL

Immortal - *July 26, 2015, 11:26 pm*

Well I originally wrote this https://hackforums.net/showthread.php?tid=4920390

And well I kinda wanted to add a little bit more functionality. So I added Multiple file support, tried to optimize my code as best as I could and add multithreading to the delete file method

Video:
[Video: https://vimeo.com/134587060]

Q & A

Q:
(Jul 24, 2015 11:13 PM)d3agle Wrote: *[CITE]* Nice source, but I'm a little confused why you are encrypting the temporary file in memory, and only writing zeros to disk... Kind of defeats the purpose of the encryption, right? :P
A:
(Jul 25, 2015 10:41 AM)Imm?rtal Wrote: *[CITE]* wanted to make sure that no trails of data were left behind for lets say a forensic data analysis team. That way if there were tiny bits in the ram that didnt get changed to zeros do to some kind of anomaly the data is encrypted. So I take the size of the file to a byte[] and then encrypt the one in the memory and overwrite with 0's to make sure it's gone then delete the file.Q:
What size file will this support?
A:
Not sure the most I tested with was 15 files that were 25.6MB and a single file that was 996MB. The program did not crash it removed all the smaller files first and idled as it deleted the big one, Took about 3mins.
Links

Link to the source:
https://github.com/Immortal-/FileDestroyer/
Scan:

https://www.virustotal.com/en/file/542ec6dfea3f9ee30084ec495dc8f70bc078a74ef764a57690674c0d460b9195/analysis/1437977954/

Link to Exe:
http://www.mediafire.com/download/1gyt8kdaa6qgga8/FileDestroyer.exe
Scan:

https://www.virustotal.com/en/file/75b5f2ed3e4a8a3028fba74dd02309f123e235a0ea08958359994c837e32560d/analysis/1437976488/

FeatureBug - *July 27, 2015, 12:02 pm*

Could you release the code on Github or something? It's a legit program with a white-hat practical use. Would love to be able to pull the source directly instead of downloading a random zip!

Immortal - *July 27, 2015, 1:52 pm*

(Jul 27, 2015 12:02 PM)FeatureBug Wrote: *[CITE]* Could you release the code on Github or something? It's a legit program with a white-hat practical use. Would love to be able to pull the source directly instead of downloading a random zip!

Done and Done, Also working on some new cool features to make this application the perfect tool. Once I'm comfortable with the feature then I'm going to optimize the cod to the best of my knowledge again..

A sneakpeak of some of what im working on.

oh btw DO NOT TRY TO REMOVE 15 900MB files it will crash your pc. xD thats on my tofix list!

FeatureBug - *July 27, 2015, 2:05 pm*

(Jul 27, 2015 01:52 PM)Imm?rtal Wrote: *[CITE]* Done and Done, Also working on some new cool features to make this application the perfect tool. Once I'm comfortable with the feature then I'm going to optimize the cod to the best of my knowledge again..

Cheers man! Few changes I'd make, but it looks shit hot. Well done ;) I'll submit a merge request once I get my .NET dev environment set up on this machine... If I even should. I'm not running Windows!

tweety - *July 28, 2015, 7:09 am*

No offence Chris, but this is probably the 40598509486645th file destroyer we had in this section.

Cr1p - *July 28, 2015, 11:23 am*

It's a nice project you got going.

It looks like you only do 1 pass on each file. If you REALLY wanted to be secure you could do more. It'd take some serious work to get past one pass in my opinion.
https://en.wikipedia.org/wiki/Gutmann_method

It is believed data can be recovered from hard drives even after 3 passes.

Expanding security can be a small addition if you like coding for fun.

Edit: Some thing else you can look at: https://en.wikipedia.org/wiki/Data_recovery

Immortal - *July 28, 2015, 11:53 am*

(Jul 28, 2015 11:23 AM)Cr1p Wrote: *[CITE]* It's a nice project you got going.

It looks like you only do 1 pass on each file. If you REALLY wanted to be secure you could do more. It'd take some serious work to get past one pass in my opinion.
https://en.wikipedia.org/wiki/Gutmann_method

It is believed data can be recovered from hard drives even after 3 passes.

Expanding security can be a small addition if you like coding for fun.

Edit: Some thing else you can look at: https://en.wikipedia.org/wiki/Data_recoveryThankyou very much for this information this data removal and bytes are fascinating me currently.

Cr1p - *July 28, 2015, 7:49 pm*

(Jul 28, 2015 11:53 AM)Imm?rtal Wrote: [CITE] Thankyou very much for this information this data removal and bytes are fascinating me currently.No worries man - I was the same when I found out about this stuff.

Have fun with it before it gets boring.

Jopper - *July 29, 2015, 7:23 pm*

Not to be antagonistic towards the code, as I actually like it, but I would just like to point out the frustrating fact that the strings used in C# are pretty much plain-view. In response to d3agle 's question, I just felt it was important to note the (reasonable) possibility of finding some of the plain-text string's values somewhere down the line.

posixg0d - *July 31, 2015, 3:41 pm*

Wanted to note that within NTFS, files < 1024 bytes are also stored within the $MFT record. In short, don't entirely rely on file shredders especially not on anything less than a meg in size (which often text files are).

Immortal - *July 31, 2015, 6:42 pm*

(Jul 31, 2015 03:41 PM)posixg0d Wrote: [CITE] Wanted to note that within NTFS, files < 1024 bytes are also stored within the $MFT record. In short, don't entirely rely on file shredders especially not on anything less than a meg in size (which often text files are).

I have a question about this, Would pumping a file full of random data to make it 2mb and then shredding it would that achieve the effect I'm looking for.

ubbelol - *August 1, 2015, 1:58 am*

Code var newData = new byte[fi.Length];

```
for (var i = 0; i < newData.Length; i++)
{
newData[i] = 0x0;
}

using (Stream s = File.Open(filePath, FileMode.Open))
{
s.Position = 0x0;
s.Write(newData, 0, newData.Length);
s.Close();
}
```

Why do you initialize the array before writing if you know it's just gonna be 00's? After looking over your code quickly this is most likely a big factor to why your application crashes. Additionally, why do you call GC.Collect(); (https://github.com/Immortal-/FileDestroyer/blob/master/FileDestroyer/FileDestroyer.cs#L351). I can almost guarantee you it doesn't do what you think it does and it's completely useless in your situation.

Immortal - *August 1, 2015, 9:27 am*

(Aug 1, 2015 01:58 AM)ubbelol Wrote: [CITE] Code var newData = new byte[fi.Length];

```
for (var i = 0; i < newData.Length; i++)
{
newData[i] = 0x0;
}
```

```
using (Stream s = File.Open(filePath, FileMode.Open))
{
s.Position = 0x0;
s.Write(newData, 0, newData.Length);
s.Close();
}
```

Why do you initialize the array before writing if you know it's just gonna be 00's? After looking over your code quickly this is most likely a big factor to why your application crashes. Additionally, why do you call GC.Collect(); (https://github.com/Immortal-/FileDestroyer/blob/master/FileDestroyer/FileDestroyer.cs#L351). I can almost guarantee you it doesn't do what you think it does and it's completely useless in your situation.

Why do you initialize the array before writing if you know it's just gonna be 00's?

The way I was explained arrays (no real .net code education) was when you initialize them you must provide how big the array is or is going to be so, to overwrite the whole file I got to offset 0x0 and then fill the filesize with zeros.

my question is how else would you do this?

Additionally, why do you call GC.Collect();

This was to clear the memory of the program, and it was not supposed to be in this version, The version I currently have has a graph that when it is being used it causes a weird memory bug that when the graph is on the ram just fills up to 100% and then crashes unless after each file I do GC.Collect();

this is a feature that will be removed evently.

d3agle - *August 1, 2015, 9:33 am*

(Aug 1, 2015 09:27 AM)Imm?rtal Wrote: *[CITE]* Additionally, why do you call GC.Collect();

This was to clear the memory of the program, and it was not supposed to be in this version, The version I currently have has a graph that when it is being used it causes a weird memory bug that when the graph is on the ram just fills up to 100% and then crashes unless after each file I do GC.Collect();

this is a feature that will be removed evently.

Good idea. I've learned calling GC.Collect() is never a good idea... The best way to handle this is to find the source of the issue, and properly handle it to the best of your ability.

ubbelol - *August 1, 2015, 12:03 pm*

(Aug 1, 2015 09:27 AM)Imm?rtal Wrote: *[CITE]* The way I was explained arrays (no real .net code education) was when you initialize them you must provide how big the array is or is going to be so, to overwrite the whole file I got to offset 0x0 and then fill the filesize with zeros.

my question is how else would you do this?

You're right you have to initialize an array with a size. What I meant was you don't need to write the entire buffer in one chunk like you do right now, that is memory suicide. Take advantage of streams and write it in smaller chunks.

(Aug 1, 2015 09:27 AM)Imm?rtal Wrote: *[CITE]* This was to clear the memory of the program, and it was not supposed to be in this version, The version I currently have has a graph that when it is being used it causes a weird memory bug that when the graph is on the ram just fills up to 100% and then crashes unless after each file I do GC.Collect();

this is a feature that will be removed evently.

There's almost never a reason to manually call GC.Collect, at least for the kind of stuff people do here on HF. If you have a memory leak then work on trying to find it instead of cheap workarounds.

Immortal - *August 1, 2015, 2:55 pm*

(Aug 1, 2015 12:03 PM)ubbelol Wrote: [CITE] You're right you have to initialize an array with a size. What I meant was you don't need to write the entire buffer in one chunk like you do right now, that is memory suicide. Take advantage of streams and write it in smaller chunks.So like File sectors?

// calc file sectors
//credit to
pseudobyte

PHP Code
<?php
double sectors = Math.Ceiling(new FileInfo(filename).Length / 512);

(Aug 1, 2015 12:03 PM)ubbelol Wrote: [CITE] There's almost never a reason to manually call GC.Collect, at least for the kind of stuff people do here on HF. If you have a memory leak then work on trying to find it instead of cheap workarounds.Yeah I figured this out and planned on removing it and or finding the leak and fixing it but I kinda started learning c++ (my books got here) So I might put a hold on this project or I might port it to c++ not sure yet.