Introduction to Hacking Thick Clients: Part 3 – The File System and Registry

Introduction to Hacking Thick Clients is a series of blog posts that will outline many of the tools and methodologies used when performing thick client security assessments. In conjunction with these posts, NetSPI has released two vulnerable thick clients: BetaFast, a premier Betamax movie rental service, and Beta Bank, a premier finance application for the elite. Many examples in this series will be taken directly from these applications, which can be downloaded from the BetaFast GitHub repo. A brief overview is covered in a previous blog post.

Installments:

1. The GUI
2. The Network
3. The File System and Registry
4. The Assemblies
5. The API
6. The Memory

Information Disclosure

Many applications have been written by developers who could not resist the sweet siren song of storing sensitive information in the file system and/or registry. This has included social security numbers, credit card numbers, database connection strings, credentials, etc. Developers may encrypt this data for an added sense of security, but there could be an insecure or custom encryption implementation. If it is a necessity to store encrypted data, the key must always be stored in a separate location, such as securely in a database.

Focusing the Test

Testing for information disclosures in the file system and registry is quite simple – you just need to look at the files and registry keys used by the application and make sure there isn't any sensitive information there. But applications often make an insane number of calls to files and registry keys. That's why it is crucial to focus on the areas of the application most likely to write or read sensitive data.

Installer

If it's possible to test the installation, always use this as an opportunity to look for writes to the file system or registry. It's the perfect time for data to be written that the application will later read. Will the application be connecting to a database? Maybe the installation process writes the database connection string to a registry key. Maybe default administrator credentials are being stored in the file system.

Application

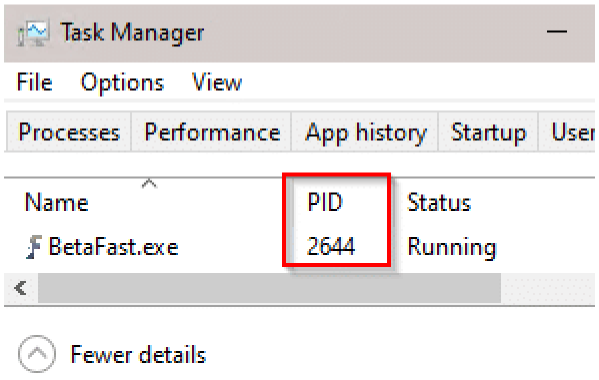
In the actual application, identify functional areas that may create or modify sensitive information. This always includes the initial database connection and a login form. Plenty of applications save some information for a Remember Me function, so determine how that's being stored and retrieved. In the authenticated portion of the application, look for areas that specifically handle sensitive data.

Tools and Examples

The Sysinternals suite contains many helpful tools for testing Windows applications. The two tools I will be highlighting are AccessEnum and Process Monitor, written by the CTO of Microsoft Azure. The vulnerable application I will be highlighting is BetaFast, written by the owner of 32 Arnold Schwarzenegger movies.

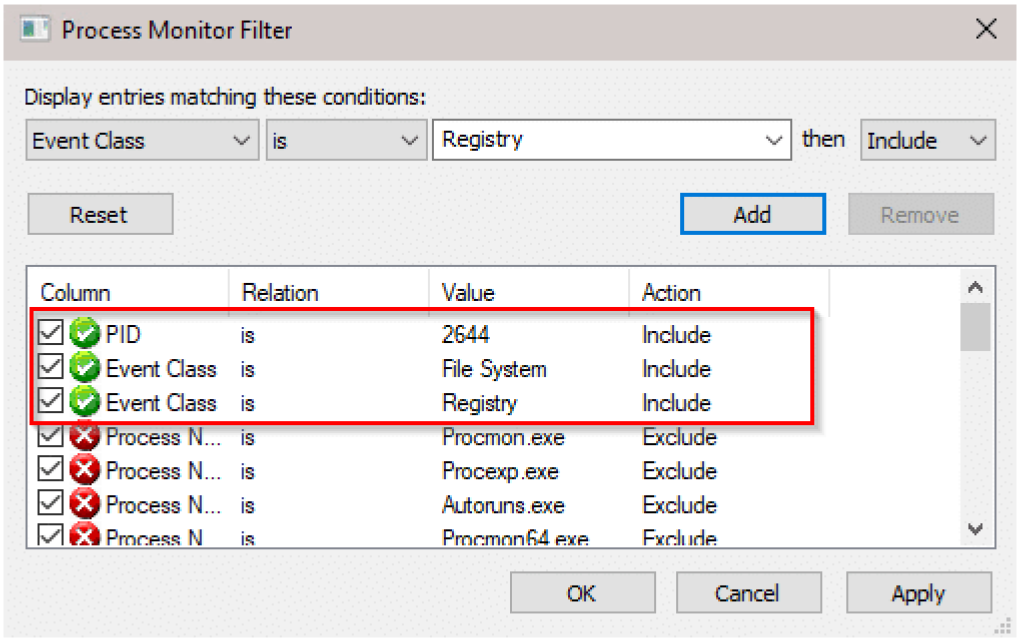
Configuration

Process Monitor will try and show you more information than you thought was possible, so the first step is to set up a filter. One item that Process Monitor can filter by is the PID of the application you're testing, so be sure to check for that in Task Manager.



Name	PID	Status
BetaFast.exe	2644	Running

In this example, I also filtered by Event Class.


Process Monitor Filter

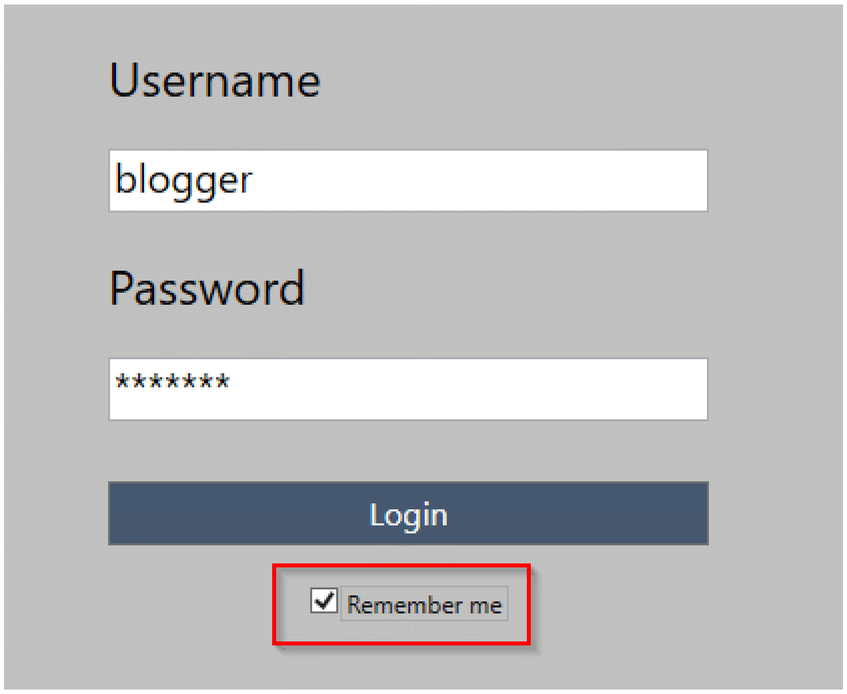
Display entries matching these conditions:

Event Class | is | Registry | then | Include

Column	Relation	Value	Action
✓ PID	is	2644	Include
✓ Event Class	is	File System	Include
✓ Event Class	is	Registry	Include
✓ Process N...	is	Procmon.exe	Exclude
✓ Process N...	is	Procexp.exe	Exclude
✓ Process N...	is	Autoruns.exe	Exclude
✓ Process N	is	Procmon64.exe	Exclude

Registry

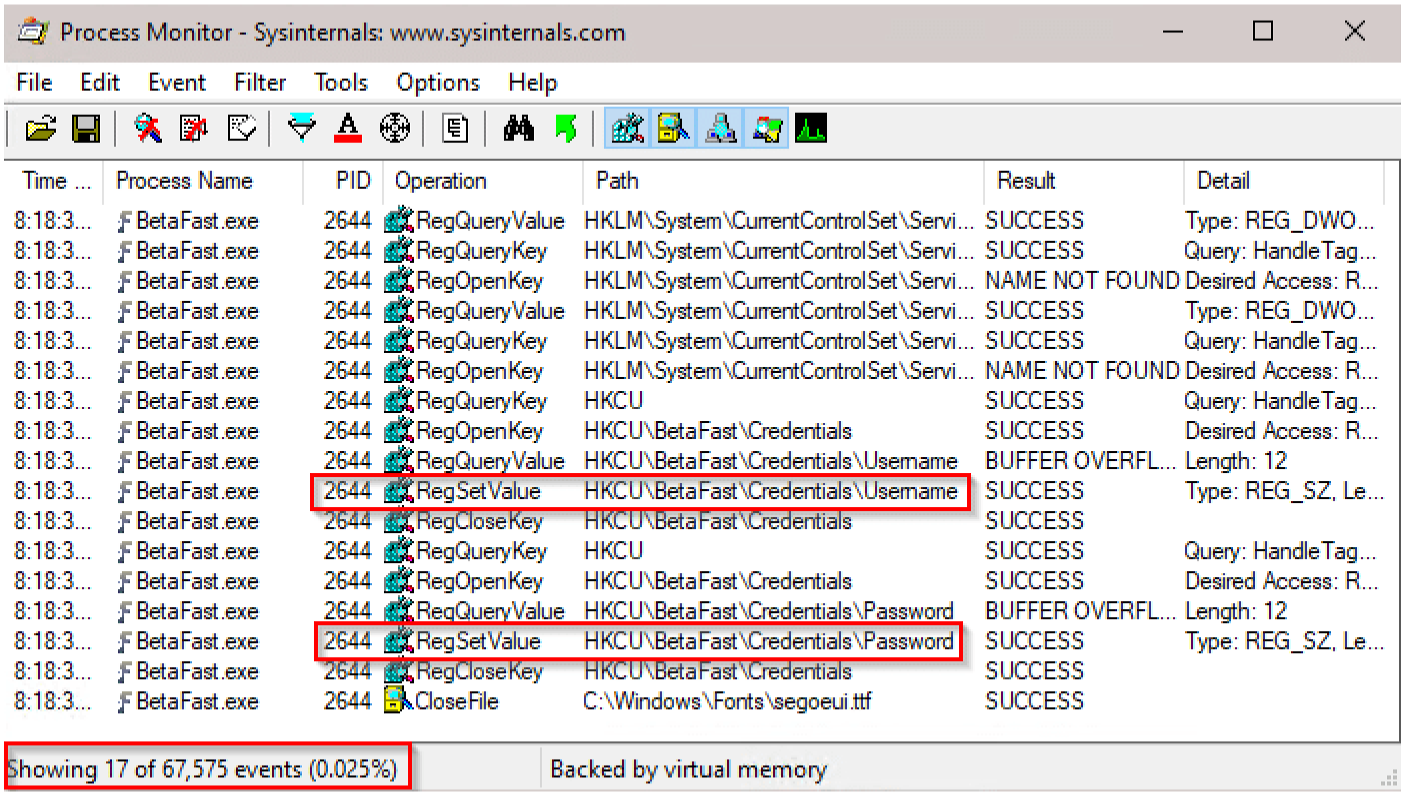
When opening BetaFast, a login form with a Remember Me function is immediately displayed.


Username: blogger

Password: *******

[✓] Remember me

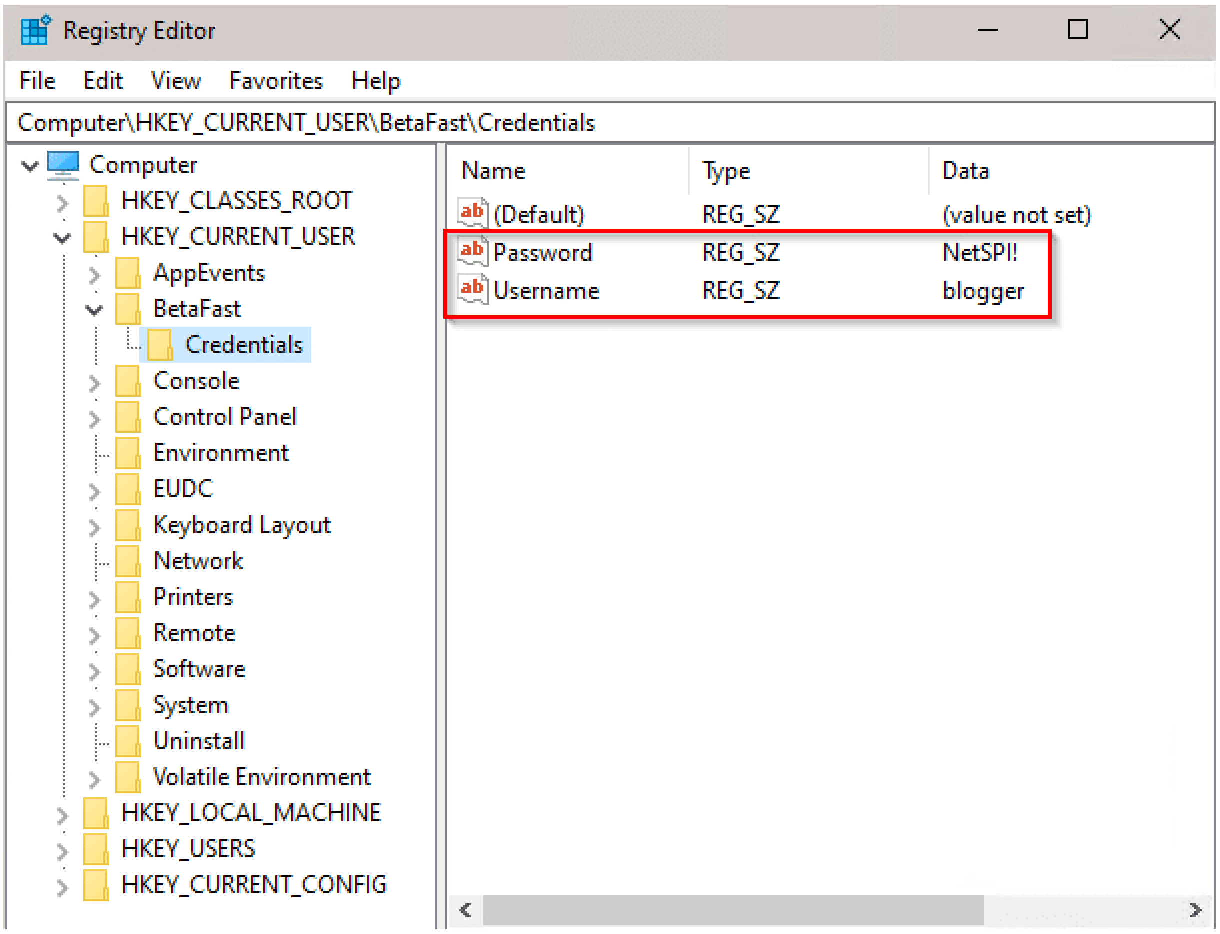
With Process Monitor running, authenticate to the application and allow it to remember your credentials. Notice that two RegSetValue operations are performed – one for a username, another for a password. Also note how many events actually took place there but were ignored thanks to the filter.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
8:18:3...	BetaFast.exe	2644	RegQueryValue	HKLM\System\CurrentControlSet\Servi...	SUCCESS	Type: REG_DWO...
8:18:3...	BetaFast.exe	2644	RegQueryKey	HKLM\System\CurrentControlSet\Servi...	SUCCESS	Query: HandleTag...
8:18:3...	BetaFast.exe	2644	RegOpenKey	HKLM\System\CurrentControlSet\Servi...	NAME NOT FOUND	Desired Access: R...
8:18:3...	BetaFast.exe	2644	RegQueryValue	HKLM\System\CurrentControlSet\Servi...	SUCCESS	Type: REG_DWO...
8:18:3...	BetaFast.exe	2644	RegQueryKey	HKLM\System\CurrentControlSet\Servi...	SUCCESS	Query: HandleTag...
8:18:3...	BetaFast.exe	2644	RegOpenKey	HKLM\System\CurrentControlSet\Servi...	NAME NOT FOUND	Desired Access: R...
8:18:3...	BetaFast.exe	2644	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
8:18:3...	BetaFast.exe	2644	RegOpenKey	HKCU\BetaFast\Credentials	SUCCESS	Desired Access: R...
8:18:3...	BetaFast.exe	2644	RegQueryValue	HKCU\BetaFast\Credentials\Username	BUFFER OVERFL...	Length: 12
8:18:3...	BetaFast.exe	2644	RegSetValue	HKCU\BetaFast\Credentials\Username	SUCCESS	Type: REG_SZ, Le...
8:18:3...	BetaFast.exe	2644	RegCloseKey	HKCU\BetaFast\Credentials	SUCCESS	
8:18:3...	BetaFast.exe	2644	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
8:18:3...	BetaFast.exe	2644	RegOpenKey	HKCU\BetaFast\Credentials	SUCCESS	Desired Access: R...
8:18:3...	BetaFast.exe	2644	RegQueryValue	HKCU\BetaFast\Credentials\Password	BUFFER OVERFL...	Length: 12
8:18:3...	BetaFast.exe	2644	RegSetValue	HKCU\BetaFast\Credentials\Password	SUCCESS	Type: REG_SZ, Le...
8:18:3...	BetaFast.exe	2644	RegCloseKey	HKCU\BetaFast\Credentials	SUCCESS	
8:18:3...	BetaFast.exe	2644	CloseFile	C:\Windows\Fonts\segoeui.ttf	SUCCESS	

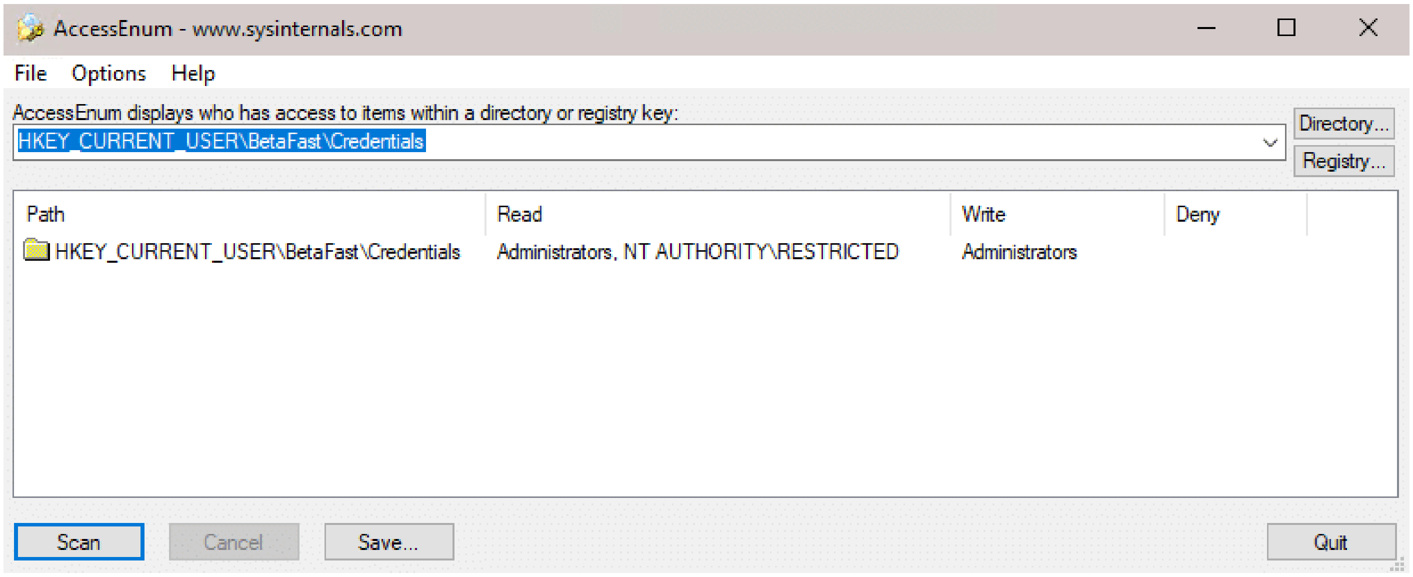
Showing 17 of 67,575 events (0.025%)   Backed by virtual memory

The registry paths might not be as obvious as Credentials\Password, so it’s more important to look for specific operations rather than paths. Registry Editor can be used to view the paths from the previous step, displaying cleartext credentials.



Name	Type	Data
(Default)	REG_SZ	(value not set)
Password	REG_SZ	NetSPI!
Username	REG_SZ	blogger

AccessEnum allows us to determine who can read and write registries and directories. The Credentials registry can only be read by Administrators and NT AUTHORITY\RESTRICTED. Since this is in HKEY_CURRENT_USER, these credentials would only be exposed to highly privileged accounts and the current user. But they would still be exposed.

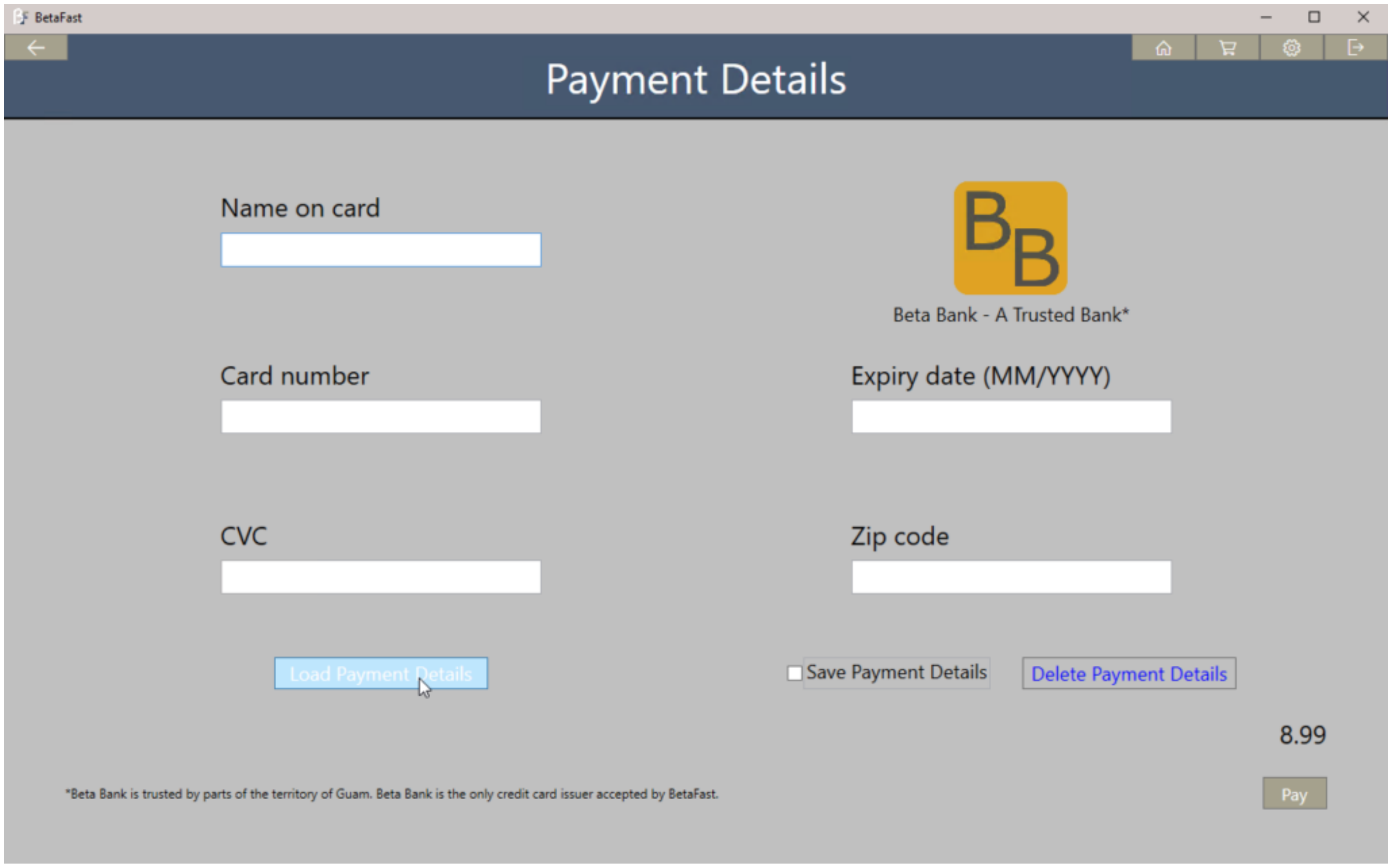


AccessEnum displays who has access to items within a directory or registry key:
HKEY_CURRENT_USER\BetaFast\Credentials

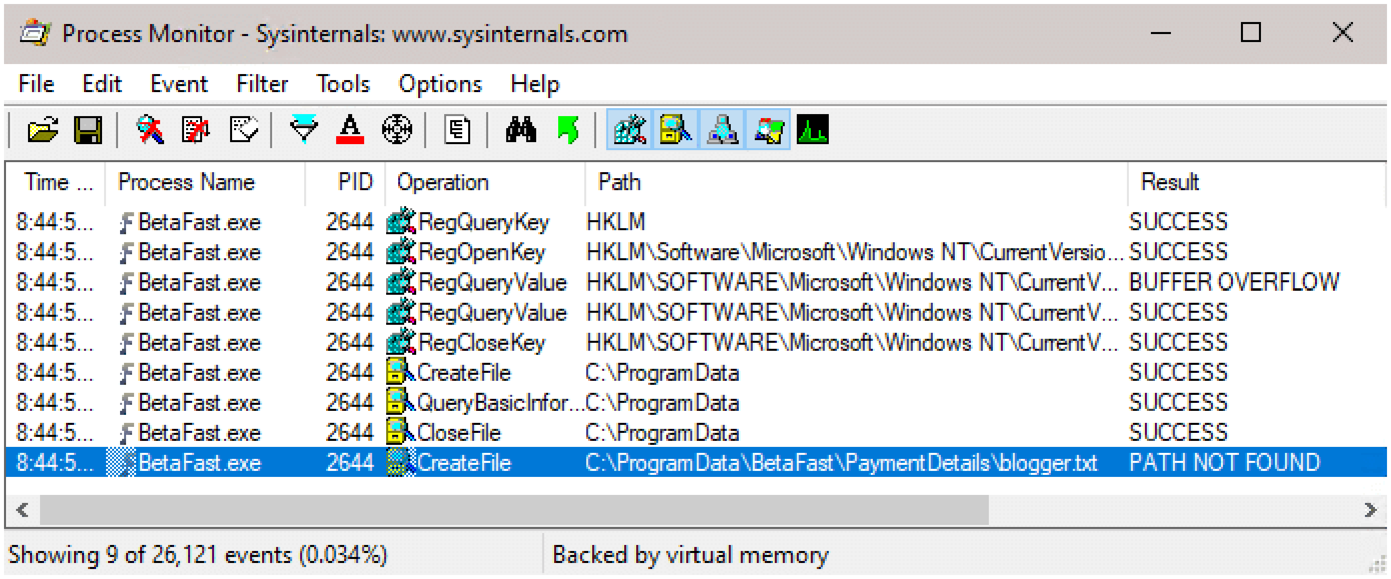
Path	Read	Write	Deny
HKEY_CURRENT_USER\BetaFast\Credentials	Administrators, NT AUTHORITY\RESTRICTED	Administrators	

File System

BetaFast also has a payment details form. Process Monitor can analyze what the Load Payment Details button is attempting to load.



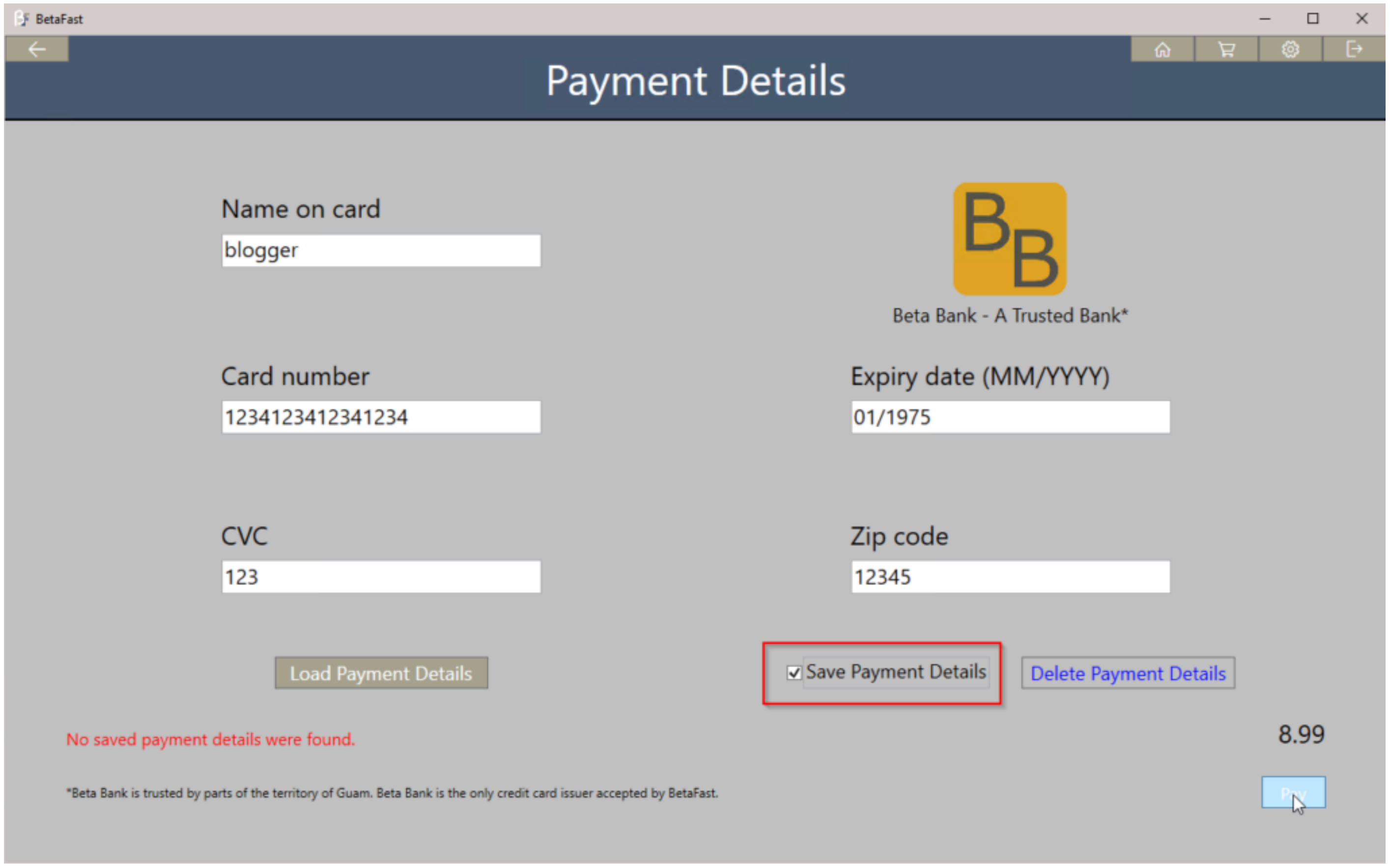
The application is looking for a file in C:\ProgramData\BetaFast\PaymentDetails with the current user's name.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result
8:44:5...	BetaFast.exe	2644	RegQueryKey	HKLM	SUCCESS
8:44:5...	BetaFast.exe	2644	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersio...	SUCCESS
8:44:5...	BetaFast.exe	2644	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentV...	BUFFER OVERFLOW
8:44:5...	BetaFast.exe	2644	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentV...	SUCCESS
8:44:5...	BetaFast.exe	2644	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentV...	SUCCESS
8:44:5...	BetaFast.exe	2644	CreateFile	C:\ProgramData	SUCCESS
8:44:5...	BetaFast.exe	2644	QueryBasicInfor...	C:\ProgramData	SUCCESS
8:44:5...	BetaFast.exe	2644	CloseFile	C:\ProgramData	SUCCESS
8:44:5...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	PATH NOT FOUND

Showing 9 of 26,121 events (0.034%) Backed by virtual memory

The next step is to submit the form with Save Payment Details checked and see what's happening in the file system.


BetaFast

Payment Details

Name on card: blogger

Beta Bank - A Trusted Bank*

Card number: 1234123412341234

Expiry date (MM/YYYY): 01/1975

CVC: 123

Zip code: 12345

Load Payment Details

[✓] Save Payment Details

Delete Payment Details

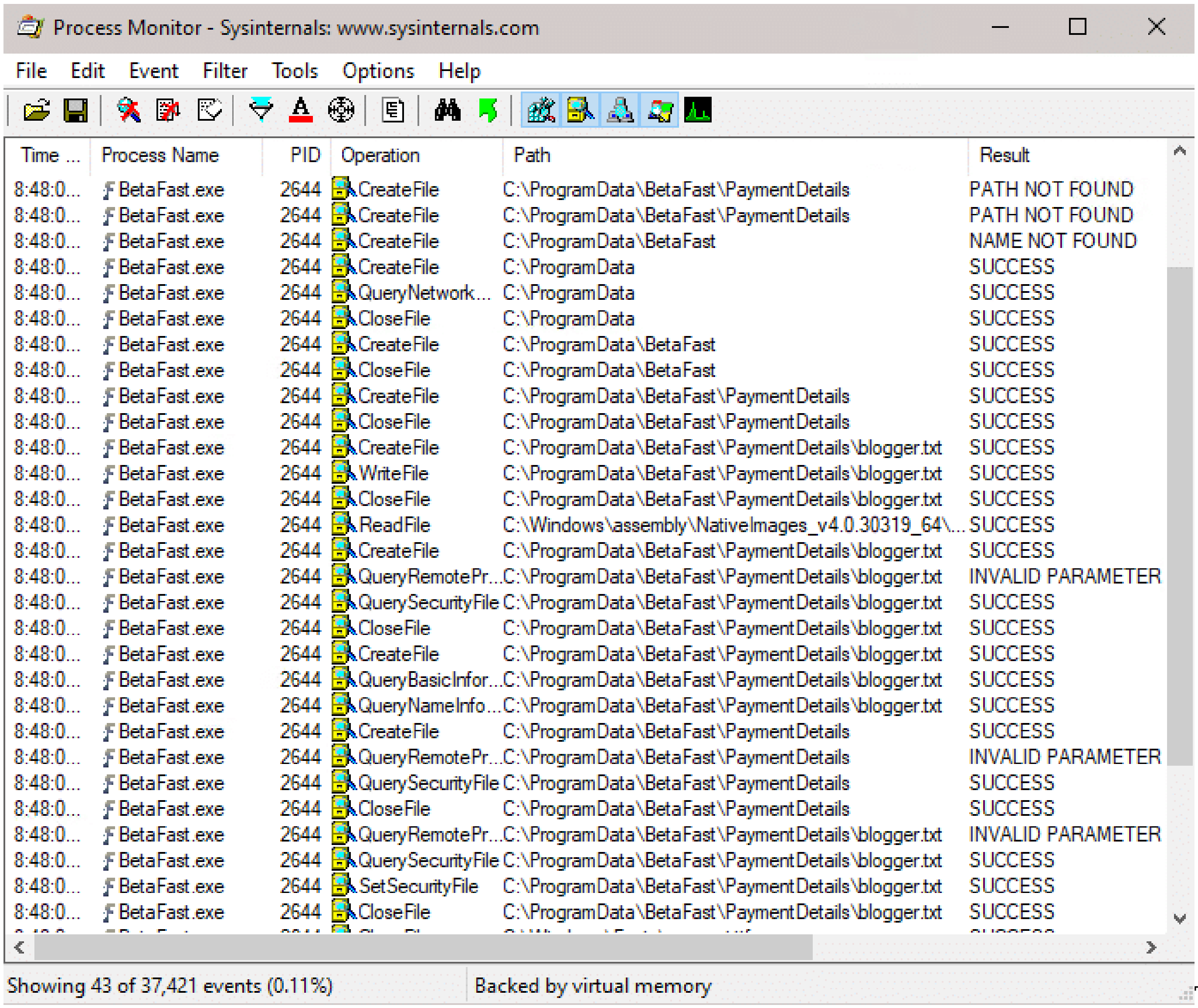
No saved payment details were found.

8.99

*Beta Bank is trusted by parts of the territory of Guam. Beta Bank is the only credit card issuer accepted by BetaFast.

Pay

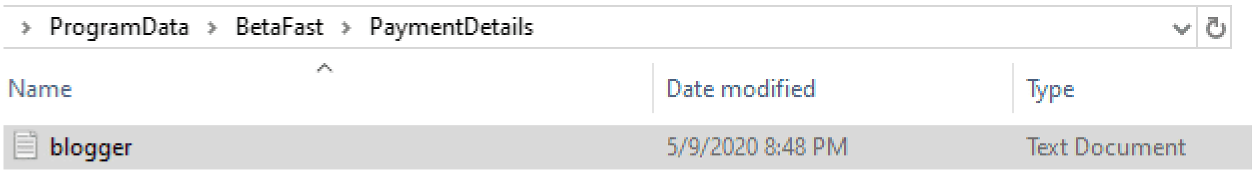
Unsurprisingly, the file that Load Payment Details was looking for is now saved.



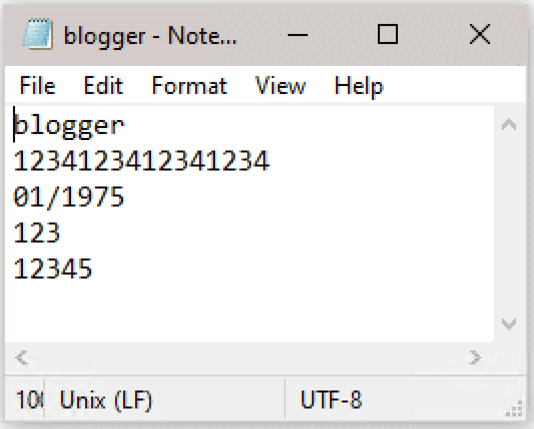
Time ...	Process Name	PID	Operation	Path	Result
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails	PATH NOT FOUND
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails	PATH NOT FOUND
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast	NAME NOT FOUND
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData	SUCCESS
8:48:0...	BetaFast.exe	2644	QueryNetwork...	C:\ProgramData	SUCCESS
8:48:0...	BetaFast.exe	2644	CloseFile	C:\ProgramData	SUCCESS
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast	SUCCESS
8:48:0...	BetaFast.exe	2644	CloseFile	C:\ProgramData\BetaFast	SUCCESS
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails	SUCCESS
8:48:0...	BetaFast.exe	2644	CloseFile	C:\ProgramData\BetaFast\PaymentDetails	SUCCESS
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	WriteFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	CloseFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	ReadFile	C:\Windows\assembly\NativeImages_v4.0.30319_64\...	SUCCESS
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	QueryRemotePr...	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	INVALID PARAMETER
8:48:0...	BetaFast.exe	2644	QuerySecurityFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	CloseFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	QueryBasicInfor...	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	QueryNameInfo...	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	CreateFile	C:\ProgramData\BetaFast\PaymentDetails	SUCCESS
8:48:0...	BetaFast.exe	2644	QueryRemotePr...	C:\ProgramData\BetaFast\PaymentDetails	INVALID PARAMETER
8:48:0...	BetaFast.exe	2644	QuerySecurityFile	C:\ProgramData\BetaFast\PaymentDetails	SUCCESS
8:48:0...	BetaFast.exe	2644	CloseFile	C:\ProgramData\BetaFast\PaymentDetails	SUCCESS
8:48:0...	BetaFast.exe	2644	QueryRemotePr...	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	INVALID PARAMETER
8:48:0...	BetaFast.exe	2644	QuerySecurityFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	SetSecurityFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS
8:48:0...	BetaFast.exe	2644	CloseFile	C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	SUCCESS

Showing 43 of 37,421 events (0.11%) Backed by virtual memory

The file also unsurprisingly contains cleartext credit card information.

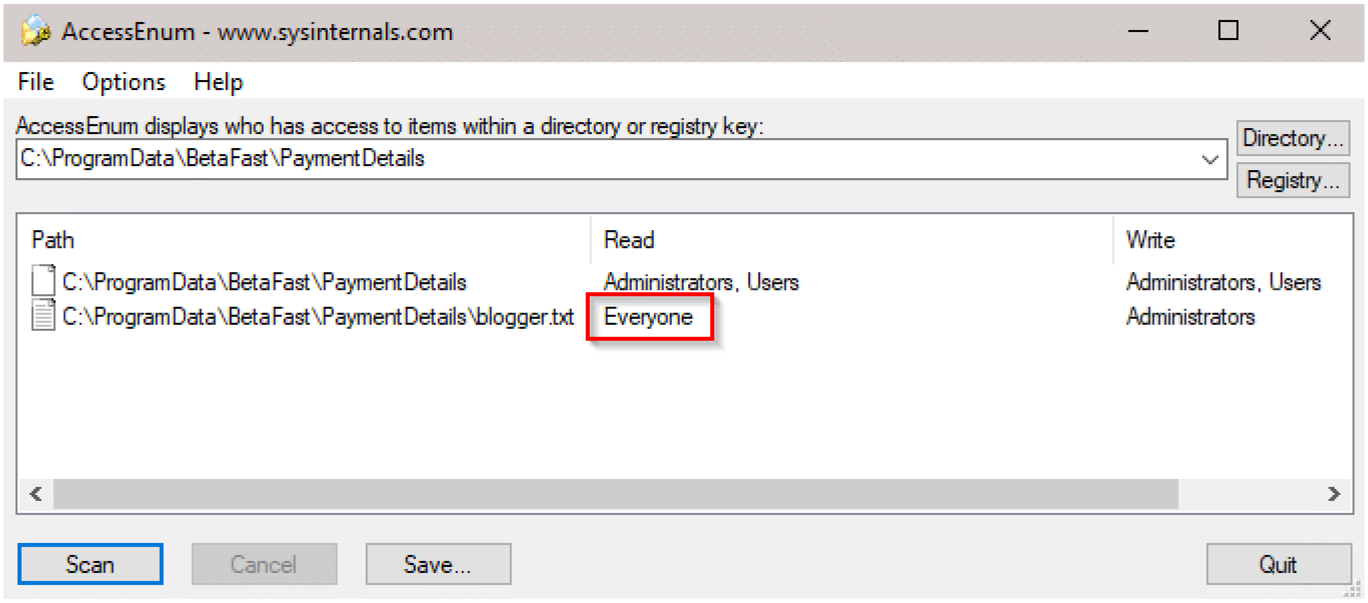

› ProgramData › BetaFast › PaymentDetails

Name	Date modified	Type
blogger	5/9/2020 8:48 PM	Text Document



```
blogger
1234123412341234
01/1975
123
12345
```

AccessEnum shows that everyone can read this file.



AccessEnum displays who has access to items within a directory or registry key:
C:\ProgramData\BetaFast\PaymentDetails

Path	Read	Write
C:\ProgramData\BetaFast\PaymentDetails	Administrators, Users	Administrators, Users
C:\ProgramData\BetaFast\PaymentDetails\blogger.txt	Everyone	Administrators

Of course, this shouldn't be surprising either. ProgramData is not user specific while the AppData folder is. Storing cleartext sensitive information in AppData would still show up in one of our reports, but it at least wouldn't be readable by everyone on the system.

DLL Hijacking

This topic could be a blog or two of its own, and we even have one from 2012. But it also has to do with the file system, so it's worth a brief overview here as well.

1. Applications will search for DLLs to load.

2. Some applications will not specify a fully qualified path.
3. Windows will search a predefined order of directories for this DLL.
4. Using Process Monitor, an attacker could identify this DLL because the application would be trying to open a .dll file that could not be found.
5. An attacker could then place a malicious DLL in one of the directories in the search order, causing the application to execute malicious code.

The problem with DLL Hijacking is in step 5. The issue mostly comes down to configuration issues. Microsoft highlights a few different attack scenarios and their severities in this article. They also have an article on how to prevent and identify these attacks. For even more additional reading on attack scenarios, check out this article as well as this one.

The main point I wanted to cover is that file system vulnerabilities don't just come from writing data – they sometimes stem from what's being read. Also, Process Monitor is again very useful.

Referenced Articles:

- https://blog.netspi.com/testing-applications-for-dll-preloading-vulnerabilities/
- https://msrc-blog.microsoft.com/2018/04/04/triaging-a-dll-planting-vulnerability/
- https://support.microsoft.com/en-us/help/2389418/secure-loading-of-libraries-to-prevent-dll-preloading-attacks
- https://liberty-shell.com/sec/2019/03/12/dll-hijacking/
- https://itm4n.github.io/windows-server-netman-dll-hijacking/