SQL Forensics

If you owned a convenience store, or other business that was likely to get robbed you would likely have surveillance cameras with the intention of having evidence to provide the police with the information that they need in order to track down and prosecute the person who may rob the store.

Most of our databases have far more value to our company than the entire contents of a small convenience store, and whether you are willing to admit it or not they are probably more likely to get hacked than a convenience store is to get robbed. But we don't have the surveillance system in place to monitor what happened when and why.

Once you have been hacked it is very challenging to find out what someone may have done to your databases, especially when you don't find out for several months.

As a DBA, imagine your manager coming to you saying that 3 months ago between the 10th of the month and the 26th of the month the IT team has determined that a hacker obtained unauthorized access to the network. They have a log of all files that were transferred out of the network. What we need from you is to know that parts of the database did the change or modify. Did they install a job anywhere that checks for confidential information and emails it to them on a regular basis, did they alter a stored procedure to move money into their account? Did they add another user to the database that will allow them unlimited access whenever they want? How would you answer these questions.

It would be nice if we could just ask SQL Server for a forensic analysis of what occurred between the dates in question and get a quick summary of what was changed, who was in the system and what might have been done. Unfortunately SQL Server doesn't have a good way to do this, you could get a product that would scan the transaction logs to show what may have changed, if you still have the transaction logs from 3 months ago. You could get some of what you are looking for

from Extended Events, maybe some from SQL Source control if you are using a source control product.

All those after the fact solutions are like working with a sketch artist for a convenience store robbery because you didn't have a functioning camera.

If your company is large enough to have a data center, or to use a data center co-location facility there are probably plenty of cameras. But none of the cameras will catch the hacker with remote access to your systems.

Many of us when asked what changed on our SQL Server 3 months ago over a 2 week period would probably do some research and eventually come up with the "I don't know" answer. Management may see it like this... "You don't know if anyone did anything to compromise the integrity of your database, What good are you?"

How do we solve this. Plan ahead, it can actually be easier that installing video cameras. Anyone who has ever installed surveillance cameras knows how painful it can be to get them in the right place, the right angle, the right view.

Github Open Source Project

Introducing SQL Forensics, an open source project that I have just started hosted at Github. https://github.com/SteveStedman/SqlForensics

The reason that I have created this as an open source project is to have the highest level of

transparency in the project. All the source code is there so look it over, don't trust me, confirm for yourself everything that it is doing to monitor your system, and if you don't like something it is doing, then just change it.

What does it do today... Not much since I just started working on it yesterday. But here is where I plan to go with it.

- Tracking of who changed what and when
    - Stored Procedures
    - Functions
    - Tables
    - Triggers
    - Foreign Keys
    - Schemas
    - Users
    - Logins
    - Permissions
    - Add/Remove a Database
    - Database configurations
    - Instance / Server configurations
- Alerting
- Investigation Tools

So far if you install the SQL Forensics database it tracks any changes made to the global configuration settings for the current server, sp_configure. You can see the baseline, and any changes by querying the [Log] table.

This shows that sp_configure 'show advanced options' was changed 3 times, but reconfigure was never run. The value column shows the configuration value, as well as the value currently in use.

What would you need to know?

What do you need to know for a proper SQL Forensic investigation to know if someone is in the system? Drop me an email or post a response to this message with your needs.

`Links:`

- Github project: https://github.com/SteveStedman/SqlForensics