AutoRun Logger

All your autorun data in one place…

Background

- The key point is to always assume your network is breached and therefore your security team should always be on the hunt for adversaries
- This presentation is aimed at introducing Info-Assure's AutoRun Logger (ARL) system
- Malicious software wants to have a persistence mechanism (Autoruns) so that it can survive reboots
- The go to tool for extracting autoruns is the SysInternals Autoruns software, written by Mark Russinovich (Microsoft)
- AutoRun-Logger is a three component system that allows organizations to have a network wide view of all of the autorun data of their Windows hosts.

Architecture

- The first component is a simple Windows service
- The service extracts the autorun data the first time it is run (e.g. boot up), and then periodically (every 24h
- The second component is the analysis server
- The analysis server parses the data and imports it into a database
- The server compares the new dataset with the previous, if any differences are identified then an **alert** is generated

Architecture

- The final component in the system is the user interface (UI) server
- The UI server display alerts and allows the analyst to drill down into the alert
- The UI server allows the analyst to view the current autorun data for a specific host
- From the UI server, various data can be exported such as files containing the SHA256/MD5 hashes of all current autoruns
- There is also a simple search facility to allow searches across the alert/autorun data for specific values.

Windows Service

- The Windows service is written using the Microsoft .Net framework (C#)
- The service uses a local copy of the SysInternals Autoruns command line software (autorunsc.exe) to extract the autoruns data
- The data is extracted in XML format for easier parsing

Windows Service

The **autorunsc** command line used is:

```
-a * -x -h -s -t *
```

Which gives the following:

–a *: Extracts all autorun types

–x: Output in XML

–h: Calculate file hashes

–s: Verify file signatures

–t: Normalise timestamps (YYYYMMDD-hhmmss)

*: Extract all user profiles

Windows Service

- The autorun data is compressed using GZIP and is sent via TLS to the server
- The client connects to the HTTPS port and sends data to the server. The request URL takes the form of:

  https://1.2.3.4:8000/domain/host

- The Windows service loads the servers TLS (self-signed) certificate and validates the connection (Certificate Pinning)

Analysis Server

- The server is written in Golang and is designed to run on Linux
- The server uses the PostgreSQL database
- The database holds two sets of data per **domain/host** combination e.g. current and previous
- When a new set of data is received, the data in the **current** table is moved to the **previous** table and the new data inserted
- The old data is purged from the **previous table**

Analysis Server

- The server then iterates through the current data set and attempts to match the previous autorun’s using the following:
  - ItemName
  - Location
  - Profile
  - FilePath
  - Launch String
  - SHA256

Analysis

- Using these fields will permit the flagging of items that are:
  - New
  - Deleted
  - Modified (including launch strings or files via hash)
- If an autorun has been identified as new/deleted/modified then a record is added to the **alert** table

Exports

- Every hour exports are generated that provide “uniqued” lists of data
- The exports currently generated are:
  - SHA256
  - MD5
  - Domains
  - Hosts

Exports

- The data is written to a timestamped file that represents the most current data for a particular day and data type. Each time the file is overwritten, then once the date changes, a permanent record of the last data set for that day will remain. An example of the file names used are detailed below:
  - export-sha256-2016-06-16.csv
  - export-md5-2016-06-16.csv
  - export-domain-2016-06-16.csv
  - export-host-2016-06-16.csv

- The files can be downloaded from the user interface (UI) server

Archive

- The server has a configurable **archive** option that stores a compressed archive of the autorun data
- The archives are stored in sub-directories in the form "domain-host"
- Each time a new set of data is received, the XML is compressed as a zip file, using the timestamp as the file name.
- The file is hashed and the hash compared to the last archive available, if the hash is different, then the file is kept.
- This method reduces the amount of archive data required. The archive files hash (MD5) is stored in a separate file with the same file name e.g. "/arl/archive/somedoman/host10938/2016-06-13T07:21:05Z.zip.md5"