POSTER: Locally Virtualized Environment for Mitigating Ransomware Threat

Manish Shukla
TCS Research
Pune, India
mani.shukla@tcs.com

Sutapa Mondal
TCS Research
Pune, India
sutapa.mondal@tcs.com

Sachin Lodha
TCS Research
Pune, India
sachin.lodha@tcs.com

ABSTRACT

Ransomware is one of the rising malwares in the crimeware family. It encrypts the user files and demands extortion money. From the perspective of an enterprise it is very crucial to detect and stop a ransomware attack. A well studied technique is to monitor file system behavior for suspicious activity. In this work we will show the gap in the existing state of art and describe a dynamic system which learns new behavior while under attack.

Keywords

Ransomware; Malware Analysis; Behavior Modeling

1. MOTIVATION & RELATED WORK

Ransomware is one of the most discussed malwares in the present time. According to Symantec [1], it is a dangerous cyberthreat to individuals and organizations alike. The losses due to it are running in multi-million US dollars. Till now most of the reported attacks were on individual users or small businesses due to lack of security hygiene, though recently some hospitals [2] were also affected. As per[1] IoT, health care sector and big businesses will be the next big targets as they provide volume, easy access and potential high profit. From the perspective of an enterprise it is a serious threat as this might result in loss of intellectual property or money or reputation or combination of them.

The first documented ransomware using symmetric key cryptography appeared in 1989. In 1996, Young et.al.[3] discussed its ineffectiveness and presented the idea of using public key cryptography for such attacks. The current generation of ransomware uses combination of symmetric and public key cryptography for taking control of user files. Newer and better strains of the ransomware are emerging due to its success with respect to development, deployment and returns. The latest such trend is to offer Ransomware-as-a-Service wherein skilled developers write the malware and offer it as a service. This is a concerning issue as it lowers the skillset needed for mounting an attack and introduces a human agent for more effective distribution.

However there are some technical challenges when it comes to mitigation of ransomware threat. First, the *attack detection* itself is difficult owing to indistinguishibility of the malicious process from a normal application. Secondly, the *culprit identification* is tricky as the malicious component might be a single process or it could have been injected into a trusted process or spread across multiple processes. Over the past few years many solutions were proposed to detect ransomware. Some existing techniques consist of signature based pattern detection, static code analysis of binaries and runtime analysis of process behavior. Unfortunately these methods tend to fail for newer variants with signficant changes as discussed in [4, 5, 6].

A more resilient technique is based on detecting behavioral traits of the malware. The method is specifically powerful as it abstracts the behavior from actual binary structure. Kharraz et.al.[7] have first proposed to use the sequence and types of I/O Request Packets(IRP) to the file system for detecting an ongoing attack. They extended their work in [5] to include an entropy based indicator for differentiating between normal and malicious process. A similar work is presented by Scaife et.al.[4] wherein they discussed few additional indicators for such differentiation. Even though their method is more general than the previous techniques [7, 5], it has some weak underlying assumptions. Table 1 shows the behavioral traits in the existing literature.

Features	Comments	Used In
Shannon Entropy	$E_\Delta = E_{Write} - E_{Read}$	[4, 5]
Mime	Change of mime type	[4]
Similarity Change	Fuzzy Hash$_{\{orig,final\}}$	[4]
Rate of Deletion	Secondary indicator	[4]
IRP Sequence	Order of events	[5, 7]

Table 1: Existing Behavioral Traits

This work is an extension of the work done by Shukla et.al [8] on file system event analytics. Here we first discuss the gap in the existing behavior based ransomware detection solutions. Following that, we discuss our distributed solution and its components.

CCS'16 October 24-28, 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4139-4/16/10.
DOI: http://dx.doi.org/10.1145/2976749.2989051

2. NEW BEHAVIORS

To show gap in the existing state of art we developed multiple new strains of ransomware with various different behavioral traits. For focused discussion we are assuming that ransomware is downloaded and running.

Zero Replacement: In this variant we read the original file and write it in a separate SQLite database. Once done we encrypt the database and delete the original files. As it does not create intermediate files and does not replace the original file with encrypted content, therefore, [7, 4, 5] overlook this attack.

Strain 1 Zero Replacement Behavior

```
procedure RansomwareMain(void)
    create a sqlite database                          ▷ Random path
    files := enumerate all interesting files on volume
    for file in files do
        (content, metadata) := read file
        write content & metadata in sqlite database
    end for
    encrypt sqlite database
    // Delete after encrypting. Reduces suspicion.
    for file in files do
        delete file
    end for
end procedure
```

Friends in Deed: It creates multiple processes consisting of reader, writer and remover. The reader process can choose to handle only similar set of files. Once active, the malware was able to bypass funneling filters as mentioned in [4] and IRP sequencing as used in [7, 5] as different stages were delegated to different processes. Also, this behavior discourages any per process entropy change detection.

Strain 2 Friends in Deed Behavior

```
procedure ReaderProcess(extensions)
    files := enumerate files based on extensions
    for file in files do
        (content, metadata) := read file
        pass content & metadata to writer process
    end for
end procedure

procedure WriterProcess(content, metadata)
    buffer := encrypt content & metadata
    write buffer in random location
    update path database                     ▷ db of path mapping
end procedure

procedure RemoverProcess(files)
    for file in files do
        delete file
    end for
end procedure
```

Ledger Manager: In this variant we are doing an additional high entropy read for every low entropy read, therefore, $E_\triangle \approx 0$ in the following equation.

$$E_\triangle := E_{Write} - E_{Read}$$

Both [4, 5], failed to detect this strain as [4] expect $E_\triangle \geqslant 0.1$ and [5] expects writing on the same offset.

Strain 3 Ledger Manager Behavior

```
procedure RansomwareMain(void)
    high_entropy_file := create high entropy file
    files := enumerate all interesting files on volume
    for file in files do
        (content, metadata) := read file
        buffer′ := read high_entropy_file
        buffer := encrypt content & metadata
        file′ := create new file     ▷ Fixed or random path
        write buffer in file′
        update path database with file′
        high_entropy_file := file′
    end for
    create a remover process
end procedure
```

Apart from these, we created other variants depicting multitude of different behaviors, including one which operates as a kernel filter driver.

3. SYSTEM DETAILS

System: Our solution, called FileTracker, has a distributed client-server architecture. The client consists of a user mode analytics component and an event monitoring kernel module, shown in Figure 1. The kernel module is further composed of a file system filter driver and a process monitoring driver. The filter driver provides the I/O event, access to raw read/write buffer and capability to virtualize the I/O calls. The process monitoring module is responsible for notifying process related events. Also, it performs event based integrity scan of the running processes and file system filter layers. The FileTracker server aggregates all the system events from each node and builds a global model of normal and abnormal file access and modification behavior. These behavior models are then shared with each local node at regular intervals.

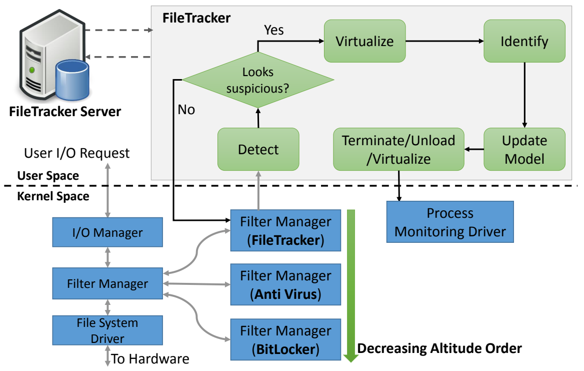

Figure 1: Client Component

Feature Extraction & Selection: Feature selection is one of the important point as it directly affects the accuracy and performance of the system. We spent a months time in observing 50 users for building our normal behavior model. Whereas, the malicious behavior model was built by running existing and new families of ransomware on systems with our solution installed. Based on our observation we have divided the suspicious features in three broad categories. **Category 1 (C1)**, consists of features directly related to files (subject),

Category 2 (**C2**), consists of features related to a process (actor), and finally, **Category 3** (**C3**), consists of features related to impact (action). Features shown in Table 2 are the most commonly exhibited behavioral traits.

Attack Detection: A kernel mode ransomware at lower altitude than the detector is much harder to detect as all the read and writes appear normal to the detector. Although, on 64 bit Windows 7 and above, this mode is only possible if the OS is running in testing mode and User Account Control (UAC) is disabled or bypassed. To detect this kind of ransomware we use our process monitoring component which looks for any filter driver hierarchy change. For detecting user mode attacks, which are much more common, we applied a two pronged strategy consisting of behavioral and structural analysis. The detector module looks for the behavioral traits listed in Table 1 and 2. A process is considered potentially harmful if it exhibits at least 3 traits. Once a candidate is found then structural analysis is performed for detecting known patterns or extracting new ones. For differentiating between normal and malicious mass deletions we use path diversity along with mime diversity. Path diversity helped us in reducing false positives as normal processes tend to deal with less diverse paths. Similarly, entropy density indicates the accumulation of encrypted content in a folder. All anomalous behaviors are notified to the user and the server for corrective actions and model update.

Features	**Strain 1**	**Strain 2**	**Strain 3**
File Attributes (**C1**)		✓	✓
Path Diversity (**C2**)	✓✓	✓	✓✓
Process Hierarchy (**C2**)	✓	✓✓	
Bytes Read (**C2**)	✓✓	✓	✓✓
Bytes Written (**C2**)	✓✓	✓	✓✓
File Handles (**C2**)	✓✓	✓	✓✓
Entropy Density (**C3**)	✓✓	✓	✓
Rate of Creation (**C3**)		✓✓	✓
Rate of Modification (**C3**)	✓✓	✓	
Rate of Size Change (**C3**)	✓✓		
Rate of Mime Change (**C3**)			✓✓

✓✓ : Observed Behavior, ✓ : Possible Behavior

Table 2: Additional Behavioral Traits

Virtualization: The virtualization aspect is provided by the file system filter driver and operates in local or quarantine mode. In local mode, it reroutes all I/O to a predesignated folder. Whereas in quarantine mode it creates a sparse file and reroute all I/O to it. In case of suspicious file system activity the filter driver creates a virtual view of the resources under attack for the suspected process. For all the reads from the suspected process, it first checks whether there exist a version in the virtual view, if yes then that is presented, otherwise original file is presented. The file write system calls always go to the virtual view. The virtualization of file system prevent further losses and helps in discovering new behavioral traits in real time.

Culprit Identification: Some variants, including the existing ones, operate by injecting themselves into a normal process. If the host process turns out to be a system process then its termination may destabilize the complete system. For known processes our system currently supports detection of injected module by comparing the checksum of all the loaded modules with the checksum of modules of uninfected process. In case of positive result we try to unload the injected module or terminate if it is not a system process. Furthermore, we also analyze process command line and the payload if it is available. This last step helps us in detecting virtual machine based variants and allows us to show more helpful message to user and enhance our global model.

4. RESULTS & CONCLUSION

We executed all ransomwares on real machines as some of them have inbuilt protection against virtual environments. We profiled our solution for CPU, memory and disk usage on machines with Intel i5 4310U @2 & @2.6 GHz with 4 GB RAM and running Windows 7 64 bit with *TESTSIGNING* ON. The preliminary results seem promising as we were able to detect all of the existing and the new variants without losing more than 20 files. The CPU usage was not more than 1-2% and memory usage not more than 40 MB. There was slight delay observed, in magnitude of milliseconds, in disk activity as we rerouted I/O calls to user mode for analysis. Refer to Table 2 for behavioral traits exhibited by the Strains 1, 2 & 3. We got optimal detection performance by using category 3 as screening, category 2 as confirmation and category 1 as cross validation step. We have reduced false positives from applications like 7z archiver by using path diversity and observing it in virtual view.

In this work we have shown that new variants of ransomware are possible. We have extended the existing literature by adding new behavioral traits for new variants. As ransomware will evolve more new behaviors are bound to be seen. The current usage of static behavior indicators will either result in high false positives or missed detection. To mitigate that, we have used file system virtualization to minimize losses and learn new behaviors during an attack. In the future work we will harden our identification logic and will make the virtualization layer more robust.

5. REFERENCES

[1] Symantec. An ISTR Special Report: Ransomware and Businesses 2016. http://goo.gl/CjH90k, 2016.

[2] Washington Post. LA Hospital Pays Hackers After Ransomware Attack . https://goo.gl/IVx60L, 2016.

[3] Young et.al. Cryptovirology: Extortion-based security threats and countermeasures. In *Symposium on Security and Privacy*, pages 129–140. IEEE, 1996.

[4] Scaife et.al. Cryptolock (and drop it):stopping ransomware attacks on user data. In *International Conference on Distributed Computing Systems*. IEEE, 2016.

[5] Kharraz et.al. Unveil: A large-scale, automated approach to detecting ransomware. USENIX Security Symposium, 2016.

[6] Ma et.al. Shadow attacks: automatically evading system-call-behavior based malware detection. *Journal in Computer Virology*, 8(1-2):1–13, 2012.

[7] Kharraz et.al. Cutting the gordian knot: a look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015.

[8] Shukla et.al. Poster: Winover enterprise dark data. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015.