The Technical Landscape of Ransomware: Threat Models and Defense Models

by Barton P. Miller and Elisa R. Heymann

1 Introduction

Ransomware has become a global problem, striking industry, academia and government alike. These attacks affect the smallest businesses, the largest corporations, and have even shut down IT operations at entire universities.[1,2] While there have been many papers describing the threats and risks associated with ransomware, in this paper we take a more technical approach. We start with a discussion of the basic attack goals of ransomware and distinguish ransomware from vandalism. Our goal is to present the broad landscape of how ransomware can affect a system and suggest how to prepare to recover from such an attack. We present a canonical model of a computing system, representing the key components of the system. This system model forms the basis of our discussion on specific attacks. We then use the system model to methodically discuss ways in which ransomware can (and sometimes cannot) attack each component of the system. For each attack scenario, we describe how the system might be subverted, the ransom act, the impact on operations, difficulty of accomplishing the attack, the cost to recover, and the ease of detection of the attack. We also describe strategies that could be used to recover from these attacks. In this paper we are focused on recovery not prevention. As such, we are not discussing how the ransomware might enter a computer system. The assumption is that the attacker did enter the system and rendered it inoperative. These attacks might result from a human engineering attack, an unpatched known vulnerability, or a zero-day vulnerability. Note that this document represents our best understanding of the current threats and attacks. We actively solicit corrections, feedback, and contributions to make this document more accurate, complete, and timely. Please send your comments to the authors.

2 Ransomware Attack Goals

Our focus in this document is on ransomware, that is software that causes payment to be extorted or some penalty to be imposed. These penalties can come in two varieties:

1. The contents of the computer system are modified, typically encrypted or deleted, so that the system becomes inoperative. This is done in a way that the attackers can restore the system to normal operations after a ransom payment is made.

2. Data from the computer system is exfiltrated. The attackers

[1] "El Govern destina 3,5 millones a la UAB para recuperarse del ciberataque" ("The Government allocates 3.5 million to the UAB to recover from the cyberattack"), La Vanguardia, November 23, 2021. https://www.lavanguardia.com/vida/20211123/7883348/govern-destina-3-5-millones-uabrecuperarse-ataque-informatico.html

[2] Scott Jaschik, "College Closes After 157 Years", https://www.google.com/url?q=https://www.insidehighered.com/news/2022/04/01/lincoln-collegeillinoisclose&sa=D&source=docs&ust=1657816248673164&usg=AOvVaw2Jbax20QvbjxCxMKVxUXR

demand a blackmail payment to prevent the data from being revealed to the public.

Attacks can combine the above two varieties. We identify four basic operations for malware:

(ENC) Encryption. This common operation encrypts some portion of the storage of the victim system, promising to reverse the encryption if payment is made.

(LOC) Lockout prevents the victim from accessing some system functionality. A lockout might involve an operation such as changing a password, creating a password where none previously existed , or modifying critical code such as the BIOS or firmware.

(EXF) Exfiltrating data provides the attacker with potentially private, proprietary, or sensitive data taken from the victim system. The attacker then blackmails the system owner by threatening to reveal the private information.

(DEL) Deleting data prevents some or all of the normal system operation. For this to be ransomware, and therefore reversible, it must be combined with exfiltration.

We noted that (ENC), (LOC), and (DEL) are attacks on availability and (EXF) is an attack on confidentiality. We distinguish between a ransom attack and plain vandalism. Vandalism is an attack for which there is no meaningful payment option. While there are some relevant similarities, in this paper we are not discussing vandalism.

Consider the NotPetya attack in 2018[3] on the global Maersk shipping company that wiped out the contents of the disks on the tens of thousands of computers on the Maersk corporate network. At first glance appeared to be ransomware, but it offered no functional payment option. NotPetya turned out to be malicious vandalism on a global scale.

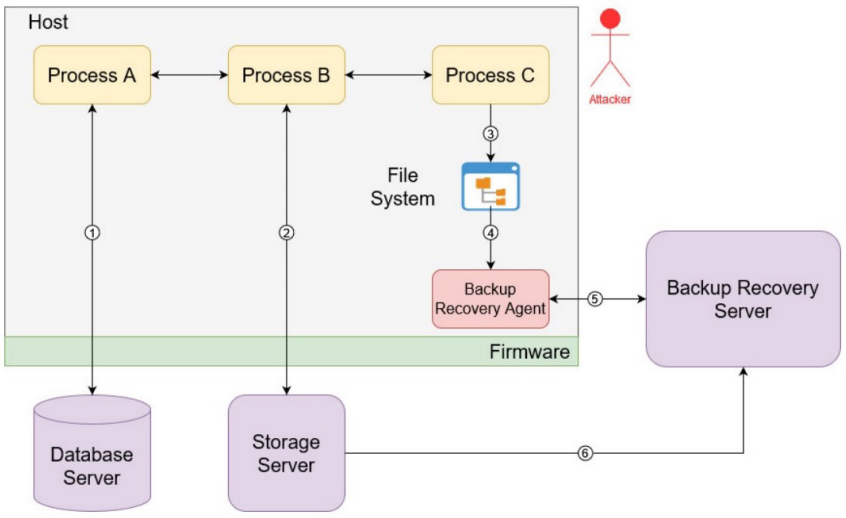

Figure 1: Canonical System

3 A Canonical System Model

We start with a model of the computer system that is being attacked, shown in Fig. 1. The goal of this model is to represent components of a system that might be attacked. The enclosing "Host" gray box represents a single computer system that is under attack. All components that are outside that box reside on different computer systems.

We start with three user processes, each of which is present to represent a different kind of attack.

Process A: User program accessing an external database service that might be in the local facility or remote.

Process B: User program accessing an external storage server (e.g., a file server or storage appliance) that might be in the local facility or remote.

Process C: User program accessing the local file system.

File System: Files that are stored on devices local to this host.

Backup Recovery Agent: Local service responsible for selecting files to back up and recover.

Backup Recovery Server: External service supporting the backup and recovery of files. It might be local or remote.

Database Server: External service running in the local facility or remotely, accepting queries from Process A.

Storage Server: External service running in the local facility or remotely, accepting file system requests from Process B.

Firmware: Semi-permanent software embedded in the devices associated with the host. These devices might include the motherboard (BIOS/UEFI and boot code), hard drives, or network card.

We illustrate both the components of the system and interaction of the components because an attack can operate on data while it is stored, data at rest, or data while it is being operated on or transferred, data in motion. We also distinguish between attacks that affect the system, which includes the operating system kernel and firmware, and those that affect user code, which includes any process running on the local host (in Fig. 1, Processes A, B, and C, and the Backup Recovery Agent).

[3] "NotPetya Technical Analysis", LogRhythm Labs, July 2017.

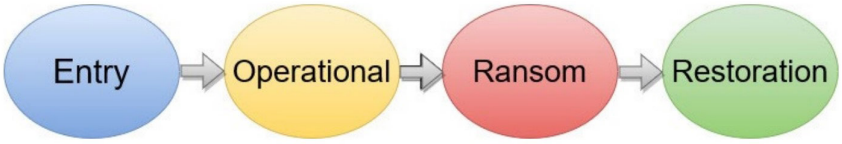

Figure 2: Workflow for a Successful Ransomware Attack

4 Attack Assumptions

This paper is focused on the recovery aspects of ransomware. As such, we are not discussing how the attacker enters the system. We assume that there has been a successful exploit and we are interested in how the attacker effects the ransom.

4.1 Attack Operations

Some of the basic operations that ransomware might use appear below:
Read, write, or create arbitrary files: These files might be on a local file system or on a remote server. The access could result in an exfiltration, encryption, or deletion of files. It could also result in modification of system configuration information. *Execute arbitrary code:* Executing any program on the system allows a wide range of control of the system. If you combine this operation with the ability to create or modify files, this means that any desired program or script can be created and executed.
Inspect the state of any process: Any information contained in the execution state of a process is available for viewing. Packages like Dyninst[4] simplify this access. *Modify the state of any process:* In the same way that a process' state can be read, it can also be modified, so, any existing running program can have its behavior changed. *Modify the state of the operating system:* A privileged attacker can modify the code or data within the operating system.

4.2 Attack Workflow

A ransomware attack goes through four basic stages, as shown in Fig. 2.
The **entry stage** is based on the initial system exploit. The exploit might be based on a human engineering attack, a known vulnerability in software that has not been updated, or a zero day attack. This part of the workflow is out of the scope of our discussion. This stage needs to be stealthy and may happen well in advance of the operational stage.
The **operational stage** is when the major damage to the system occurs. It is in this stage that data is encrypted, overwritten, deleted, or exfiltrated. An attack that encrypts or removes stored data, will transition to the ransom stage, leaving the system non-functional. To be most effective, it should operate quickly to avoid detection and interruption.
An attack that encrypts the data in motion can allow the system to keep operating even though the data is encrypted. The system would be modified so that data is encrypted when written and decrypted when read. The attacker chooses the time of transition to the ransom stage by deleting the decryption key and shutting down the system.
Lockout attacks prevent future operation of the system by changing a password, creating a new one, or overwriting critical code. After the modification, the system typically continues to operate normally until the user logs out or the system restarts.
There are, however, types of attacks that will not disable the system at all. For example, a pure exfiltrate attack, whose main goal is blackmail to prevent the public release of the data, will not prevent continued system operations.
The **ransom stage** requires payment to restore operation or prevent the release of private information. There must be some form of trust in the attacker to cooperate once payment is made. However, it is in the best interest of the attacker to fulfill their side of the bargain or else they endanger payments from future victims.
For systems that were disabled, the **restoration stage** allows continuation of normal operations. If data was encrypted at rest or in motion, the attacker will provide a decryption key. If the data was deleted, the attacker will provide a restore program to download the files. If a password was modified, the attacker will provide this new password. If a system component was modified, then the attacker will provide a key for the modified component to return to normal operations.
Of course, any payment of the ransom does not guarantee that there will be no future demands for payment. Only independent recovery will take the attacker out of the loop. Of course, the source of the initial exploit must also be determined and prevented.

5 The Ransomware Threat Space

Given our system model, we create a collection of threat scenarios, examining the model one component at a time to understand how ransomware attempts to prevent recovery. For each scenario, we discuss how the ransomware might subvert that component and how difficult it would be to recover after a successful attack. We also evaluate how difficult it is to carry out the attack and how difficult it is to detect it.

5.1 File System Attacks (FSA)

Files are the most common target of a ransomware attack, whether it is for encryption, deletion, exfiltration, or lockout. A file system attack can come in many forms, some of which are common in the wild and some of

[4] "The DyninstAPI Binary Instrumentation and Analysis Toolkit", https://github.com/dyninst/dyninst/

which have not yet appeared. The FSA scenario can come in three forms, attacks on data at rest, data in motion, and file metadata.

5.1.1 FSA on Data at Rest

The most common form of this attack is simple: read in a file, encrypt the contents, and write it back out. Encrypting a large amount of data can be slow, and that increases the opportunity for the system owner to discover the attack before it completes. To counter that issue, ransomware FSA's will often encrypt only part of each file. An alternative to the encryption FSA is to exfiltrate a copy of data with the intent on releasing the data publicly if no payment is made. This type of an attack is more blackmail than ransom. A recovery strategy from this type of FSA starts by making regular file system backups to a remote and safe server. Backing up files is a well understood and widely recommended practice to allow recovery from this type of attack. To reduce the chance that this server will also be attacked, it should follow several best practices for backups:

1. Backups should be "write once". Once they have been created, the server will not allow them to be modified. This is the "secure storage" criteria from NIST 1800-25.
2. The backup server should be physically secure.
3. Authentication and access to the server should be separate from other hosts. There should be a limited number of people that have access to the system and there should be a separate access enforcement mechanism.
4. File recovery should be tested on a regular basis.
5. Separate authoriziations and permissions for each backup client's files.
6. Using monitoring tools to detect when parts of the file system appear to have suspiciously encrypted content.
7. Limit the rate of backups that a client can make to prevent denial of service attacks.

Once the attacked host has been cleared of the attack, then the file system data can be restored using normal file restoration procedures.

5.1.2 FSA on Data in Motion

Data in motion attacks will encrypt, delete, or exfiltrate data *as it is being written* to the file system. This attack would require the file system code of the operating system to be modified, which makes this attack more difficult to implement. This attack could result in a situation *where recovery, even with best practices in backup, would be extremely difficult*.

This attack modifies the file system so that all data that is written is encrypted before it is stored. When data is read back, it is decrypted so that the attack is not visible until the moment of the attackers choosing. In the background, the existing stored data is encrypted with the same key. The attack could be scheduled to be triggered at a certain time. Triggering the attack would cause the encryption/decryption key to be deleted from the computer's local memory. At this point, all file reads would return encrypted data.

Note that since the system keeps operating while the files are encrypted, the backed up files will also be encrypted. This attack becomes more effective if the system is left to run for a longer period of time because the longer that the attack persists, the greater the change in the file system since the last unencrypted backup.

This attack might be discovered by tools that detect the presence of a large presence of anomalous or encrypted data in the file system. Such detection might also allow for the discovery of the encryption key before it was detected by the attacker.

Recovery from this type of attack is problematic. If the attack was to persist for an extended period of time then the backup best practices described in Section 5.1.1 would not be effective. Such an attack will likely result in potentially significant data loss.

5.1.3 FSA on File System Metadata

A file system attack is not limited to modifying the data stored in a file; it could also modify the information that describes how the data is stored, often called the *metadata*. Examples of metadata that could be modified as part of an effective attack include the file names and access permissions. The most effective file name attack would be encryption of the file names. While the file contents would remain intact and accessible, such an attack would make finding the files problematic.

Conceivably, a tool could be constructed that would compare the shape of the file system tree and file contents of the attacked file system to its most recent backup. Such a tool should be able to recover most of the file names.

5.2 Storage Server Attacks (SSA)

These attacks are similar to the FSAs: We are assuming that a privileged process can have arbitrary access to the files on the server in the same way as it would have access to local files. As such, most of the discussions from Section 5.1 apply to SSAs. We note that many of the FSAs are also SSAs. One way that this assumption is not true is that in an SSA, the server process is running on a different host, so it cannot modify the system software on that host. This limitation means that a comprehensive data-in-motion attack is not possible. While the attacker could intercept the reads and writes from the exploited host, it would not be able to intercept requests from other hosts. Under the data-in-motion attack, 7 after data is written to a file in encrypted form but before the ransom act, file reads need to transparently decrypt the data.

In addition, depending on how the storage server is configured, the attacked host may not be able to access all the files on the server nor have administrator access to that

server.

5.3 Database Server Attacks (DSA)

There are many similarities between a client process accessing a database server and a client process accessing a storage server. The main difference is the access protocol. In the storage server case, access follows the basic open/read/write/close semantics of a file system. In the database server case, access follows a more structured protocol such as SQL.

With a DSA, we can still have attacks that encrypt the contents of the database, exfiltrate the data, or remove it. We can also attack the database metadata by renaming relations or attributes and changing access permissions. In addition, we can intercept requests made by the client to the database server, so it can affect a data-in-motion attack. However, as with the SSA, we can only control the behavior of the clients on the attacked host and not those running on other hosts. This limits the effectiveness of such an attack.

5.4 Backup System Attacks (BSA)

Backup systems play a key role in providing system reliability both in response to normal system and device failure and in response to an attack. Given this key role, the backup system itself becomes an attractive target for attack. From Fig. 1, we can see that backups can be written to locally mounted disks or to a remote backup server. In both cases the attack has the same effect. The point of attack is the software on the local computer that identifies the files to be backed up and then writes them to storage.

A backup system attack modifies the data that is being written to the backup storage device or service by modifying the behavior of the Backup Recovery Agent. For this modification to be a ransom activity and not vandalism, it must be reversible. For it to be an effective ransom activity, it must be difficult to reverse without special knowledge.

This attack proceeds through the stages described in Section 4.2 (Fig. 2). During the entry stage, the attack modifies the backup software to encrypt all data that is backed up. During the operational stage, any backups that are produced will be encrypted in such a way that the user cannot use them. The longer the system runs, the more data will be stored in an encrypted, and therefore useless backup. The backup software would also be modified so that any recovery requests made during the operational stage properly decrypt the data. This recovery behavior ensures that the attack continues to be stealthy until the ransom phase is triggered.

The ransom phase is triggered by deleting the primary copy of the files from the file system and deleting the decryption key from the host. At this point, the files are gone and the backups are encrypted.

Preventing a BSA is based on limiting the damage that can occur. Such limiting requires that we can detect when backup data is unexpectedly encrypted. Such detection might be accomplished by using a file system monitoring tool as described in Section 5.1. Recovery for such an attack is problematic as the primary data is gone and the secondary data is encrypted. The longer that this attack is stealthily present in the computer, the larger the percentage of data that is likely to be encrypted.

5.5 Firmware Attacks (FWA)

Firmware is the software that is provided by a device manufacturer and runs inside a device to control that device. It is separate from the operating systems and applications that run on the computer and is stored in separate memory local to the device it controls.

5.5.1 FWA Modifying the Firmware

There have been significant firmware attacks in recent years.[5, 6] In a ransomware context, taking control of a device's firmware has serious security consequences, such as:

- Taking control of the disk firmware, preventing booting the system.
- Taking control of the keyboard firmware, allowing an attacker to set a boot password that would also prevent booting.
- Taking control of the NIC firmware, isolating a computer, or allowing remote access and control.
- Taking control of the battery firmware[7], causing shutdown of the computer at will.

The good news is that modern systems provide significant defenses against such attacks, starting with processor-based security mechanisms that provide cryptographically strong storage of keys. Unless you can open the chip and defeat its anti-tampering mechanisms, the data stored can be considered reliable and secure. The encrypted keys and certificates, combined with signing of each software update delivered to the computer from the vendor, make it difficult to replace any system component.

While a successful firmware attack can be difficult to do, recovery from such an attack can be extremely labor intensive. Such a recovery can require reprogramming the EEPROM or FLASH memory on the motherboard or in the devices themselves. While the labor to recover a few computers is manageable, the cost to recover a large number of computers, such as found in a data center or corporate network, can be prohibitive.

Best practices for prevention and recovery include:

1. Ensure that your operating system is updated to the most recent release. The newest versions of the major operating systems require signed software and (mostly) signed firmware and up to date hardware.
2. Ensure that Secure Boot has not been disabled.
3. Ensure that the Trusted Platform Module (TPM)[8] has not been disabled.

5.5.2 FWA Setting a Boot Password

A standard security feature of modern computer systems is the ability to set a boot-time password. These passwords require that the password be typed on the keyboard. The BIOS/UEFI will enforce "proof of presence" at the keyboard to accept a password. Subverting protections on setting a boot password could be done by subverting the keyboard firmware. If the keyboard says that a person is present and entering a password, then the operating system is likely to accept this entry. The boot password is stored in separate volatile CMOS RAM on the motherboard. The battery on the motherboard that powers the RAM needs to be physically disconnected to reset any security data stored in this RAM. Such disconnection may involve unsoldering the battery connection. A best practice for prevention is to have a boot password already set on your computer.

5.6 Operating System Attacks (OSA)

5.6.1 OSA on the Boot Loader and Boot Image

The boot loader is the software responsible for initial loading of the operating system kernel. As described in Section 5.5, there is a cryptographically secured chain of steps that ensures that only software that originated from the vendor will be booted. A successful attack on the boot loader or operating system boot image will prevent the operating system from starting. Until this situation is repaired, the computer will be unusable until it can be booted from an alternative device.

The Secure Boot feature, along with Boot Guard or Hardware Validated Boot, will prevent an attacker from replacing the boot loader or operating system boot image. However, it will not prevent a vandalism attack that overwrites these items with non-functional code, such as was done for NotPetya attack on Maersk's shipping network.

Most operating systems offer the ability to boot from removable media or the network. Once this is done, then the boot loader or operating system image can be restored. Such operation requires physical presence at the computer, so it is reasonable for recovering individual computers but expensive for large facilities or data centers. Best practices for this situation are the same as those described for firmware attacks in Section 5.5.

5.6.2 OSA on Account Passwords

A simple attack is to change the passwords for users and administrators. Such an attack will prevent normal access to the computer though it may not stop services from starting on booting the system.

As mentioned before, most operating systems offer the ability to boot from removable media or the network. Once this is done, then the password file(s) can be restored. Such operation requires physical presence at the computer, so it is expensive for large facilities.

Best practices here include:

1. Make sure that files storing login authentication data are included in the backups.
2. Ensure that you have escrowed the disk encryption keys for all the storage devices on all your systems.

6 Conclusion

We have described a framework for how a ransomware attack can affect a computer system, described the risks associated with such attacks, and presented some best practices for prevention and recovery. This document should be considered only a starting point for a longer technical discussion on ensuring that recovery from a successful ransomware attack can be prompt and effective.

[5] "Sean Metcalf, "Thunderstrike: EFI bootkits for Apple MacBooks via Thunderbolt & Option ROMs", https://adsecurity.org/?p=854

[6] Pavel Shoshin, "Malware delivery through UEFI bootkit with MosaicRegressor", https://usa.kaspersky.com/blog/mosaicregressor-uefi-malware/23419/

[7] C. Miller, "Battery Firmware Hacking", DEF CON 19, Las Vegas, August 2011.

[8] Trusted Computing Group, "Trusted Platform Module (TPM) Summary", https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/

Barton Miller is the Vilas Distinguished Achievement Professor and the Amar & Belinder Sohi Professor in Computer Sciences at the University of Wisconsin-Madison. He is a co-PI on the Trusted CI NSF Cybersecurity Center of Excellence, where he leads the software assurance effort and leads the Paradyn Tools project, which is investigating performance and instrumentation technologies for parallel and distributed applications and systems. His research interests include software security, in-depth vulnerability assessment, binary and malicious code analysis and instrumentation, extreme scale systems, and parallel and distributed program measurement and debugging. In 1988, Miller founded the field of Fuzz random software testing, which is the foundation of many security and software engineering disciplines. In 1992, Miller (working with his thenstudent Prof. Jeffrey Hollingsworth) founded the field of dynamic binary code instrumentation and coined the term “dynamic instrumentation”. Miller is a Fellow of the ACM and recent recipient of the Jean Claude Laprie Award for dependable computing.
Miller was the chair of the Institute for Defense Analysis Center for Computing Sciences Program Review Committee, member of the U.S. National Nuclear Safety Administration Los Alamos and Lawrence Livermore National Labs Cyber Security Review Committee (POFMR), member of the Los Alamos National Laboratory Computing, Communications and Networking Division Review Committee, and has been on the U.S. Secret Service Electronic Crimes Task Force (Chicago Area).

Barton P. Miller, Vilas Distinguished Achievement Professor, Sohi Professor in Computer Sciences,
NSF Cybersecurity Center of Excellence, University of Wisconsin-Madison
bart@cs.wisc.edu

Elisa Heymann is a Senior Scientist on TrustedCI, the NSF Cybersecurity Center of Excellence at the University of Wisconsin-Madison, and an Associate Professor at the Autonomous University of Barcelona. She co-directs the MIST software vulnerability assessment at the Autonomous University of Barcelona, Spain. She coordinates in-depth vulnerability assessments for NFS Trusted CI, and was also in charge of the Grid/Cloud security group at the UAB, and participated in two major Grid European Projects: EGI-InSPIRE and European Middleware Initiative (EMI). Heymann’s research interests include software security and resource management for Grid and Cloud environments. Her research is supported by the NSF, Spanish government, the European Commission, and NATO.

Elisa Heymann, Senior Scientist, Associate Professor
NSF Cybersecurity Center of Excellence, University of Wisconsin-Madison, Autonomous University of Barcelona
elisa@cs.wisc.edu