POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

RansomScan: Extracting Intelligence from Ransomware Families

Relatore: Prof. Stefano Zanero
Correlatori: Dott. Andrea Continella
Dott. Michele Carminati

Tesi di Laurea di:
Giovanni Bucci, matricola 860880

Anno Accademico 2017-2018

Abstract

Ransomware is a particular category of malicious software that aims to encrypt the data of the infected users, making them unusable, and asking for a ransom in exchange for the decryption keys necessary to restore them. The attacks have become increasingly refined and difficult to counter, to the point that, even considering the different solutions designed against this threat, damage remains considerable, and ransomware keeps spreading. The exploitation of technologies aimed at keeping anonymity, such as the Tor network and payments via BitCoin, has also allowed the authors of the attacks to proceed by remaining untraceable. As a result, the spread of this type of attacks has led researchers and security firms to study prevention methods, mostly on two different fronts: on one side they focused on the study of samples collected from known infections, in an attempt to define effective methods of prevention, while on the other side they try to trace the root of the attacks, investigating on the authors and trying to find the source of the infection. Our work is focused on this last aspect of the research: in an attempt to provide more in-depth information useful for studying the ecosystem behind the attacks, we have developed RansomScan, a framework for fully automated extraction of data from ransomware samples. By running the samples in a monitored environment and collecting the artifacts left in the system at the end of the malicious activity, we can extract useful information such as BitCoin wallets addresses used for the payment of ransom or URL addresses that points to hosts controlled by attackers. In this way, we obtain more information about the sample and the family to which the sample belongs, and track payments by identifying potential aggregation wallets or even physical owners. The total flexibility and generality of the framework allows a customization of the analyses, and guarantees results on any working sample, regardless of family or age, minimizing the need for human intervention during its execution. We evaluated RansomScan performing analysis on 532 samples coming from different sources: from the

219 samples that were actually working, we were able to extract 117 payment addresses and more than 181 relevant URLs, thus extracting useful information from all the working samples.

Sommario

Ransomware é una particolare categoria di malware che si differenzia dalle altre per le singolari finalitá: i bersagli dell'attacco sono i dati sensibili delle vittime, che vengono cifrati e resi inaccessibili all'utente. Questi vengono resi di nuovo disponibili solo a fronte del pagamento di un riscatto. I danni economici procurati da questo tipo di virus sono ingenti, in termini sia di riscatti pagati che di disagi derivanti dall'immobilizzazione dei dati. Gli attacchi sono diventati sempre piú raffinati e difficili da contrastare, al punto che nonostante i diversi rimedi sviluppati sinora, il rischio di perdita dei dati é ancora alto. Le tecnologie di rilevamento sviluppate non si stanno rivelando efficaci contro questa minaccia, in quanto devono essere costantemente aggiornate per fare fronte alle numerose nuove famiglie che nascono. Lo sfruttamento di tecnologie mirate all'anonimizzazione, come la rete Tor e i pagamenti tramite BitCoin, permette agli autori degli attacchi di procedere rimanendo irrintracciabili. La diffusione di questo tipo di attacchi ha portato a una ricerca di metodi di prevenzione condotta su due fronti: da un lato abbiamo lo studio dei campioni raccolti dalle infezioni conosciute, nel tentativo di definire metodi efficaci di prevenzione, mentre dall'altro si cerca di risalire alla radice degli attacchi, investigando sugli autori e cercando di tracciare la fonte dell'infezione. É in questo ultimo ambito che si colloca il nostro lavoro: nel tentativo di fornire informazioni piú approfondite utili per studiare l'ecosistema presente dietro agli attacchi abbiamo sviluppato RansomScan, un framework per l'estrazione totalmente automatizzata di dati dai campioni di ransomware raccolti. Eseguendo i campioni in un ambiente monitorato e collezionando gli artefatti rimasti nel sistema al termine dell'azione del malware riusciamo ad estrarre informazioni utili come ad esempio indirizzi di portafogli BitCoin per il pagamento dei riscatti o indirizzi URL che puntano a domini utilizzati dai malintenzionati. Data la molteplicitá delle differenti fonti di informazione presenti nei campioni di ransomware, abbiamo tenuto conto dell'importanza di trattare ogni fonte

in modo specifico per sfruttarle al meglio, e quindi abbiamo adattato la procedura di estrazione in ogni singolo caso: per i file di testo teniamo in considerazione solo quelli leggibili (cioé contenenti caratteri ASCII), le immagini sono processate in modo da agevolare il processo di OCR, e ne viene poi analizzato il testo estratto, decodifichiamo il contenuto del dump della memoria cercando stringhe composte da caratteri leggibili, infine controlliamo tipo e contenuto dei pacchetti scambiati tramite la rete. Con i risultati ottenuti da questa analisi approfondita possiamo ottenere piú informazioni sul campione analizzato e sulla famiglia alla quale il campione appartiene, e tracciare i pagamenti individuando potenziali portafogli di aggregazione o addirittura proprietari fisici. La totale flessibilitá e genericitá del framework permette una personalizzazione delle analisi e garantisce risultati su qualsiasi campione funzionante, indipendentemente dalla famiglia o dall'etá, riducendo al minimo la necessitá dell'intervento umano nel corso della sua esecuzione. Abbiamo testato RansomScan su un dataset di 532 campioni di ransomware, provenienti da famiglie diverse e finestre di tempo diverse, verificando che su 219 campioni funzionanti siamo in grado di estrarre informazioni utili da ognuno di loro. Nei casi in cui non é stato estratto nulla abbiamo verificato che di fatto il campione non esegue azioni maligne, in quanto non in grado di connettersi al server di comando per iniziare le operazioni di cifratura. Nei 219 casi funzionanti siamo riusciti ad estrarre correttamente 117 portafogli BitCoin e 181 indirizzi Tor di siti utilizzati dagli autori degli attacchi, tra piú di 500 URLs sospetti. Abbiamo constatato che nei casi in cui riusciamo ad estrarre informazioni siamo in grado di farlo indipendentemente dalla famiglia del sample analizzato, riuscendo nell'obiettivo di mantenere piú generalitá possibile. Riuscendo a distinguere i casi in cui i campioni si attivano, abbiamo inoltre reso il framework decisamente performante, con una media di 21 minuti di analisi in caso di esecuzione positiva contro un tempo medio di meno di un secondo per analisi quando il malware non effettua operazioni. Collateralmente alle analisi, abbiamo inoltre condotto uno studio sulle performance degli strumenti per fare OCR a nostra disposizione, e abbiamo constatato che per l'estrazione di dati da immagini, dato il processo di binarizzazione personalizzato, il prodotto open-source Tesseract si rivela migliore delle alternative utilizzate (Google CloudVision e Kraken-Ocr), riuscendo ad individuare 5 portafogli contro i 2 individuati da CloudVision, mentre Kraken non ne ha rilevato nessuno. Purtroppo il formato dei portafogli non aiuta l'operazione di OCR, infatti i portafogli estratti in questo modo non sono del tutto corretti, ma differenziano dall'originale per uno o due caratteri. Al fine di rendere la lettura del lavoro piú agevole, riportiamo qui la struttura della tesi:

- Nel capitolo 2 presentiamo i concetti fondamentali per la comprensione del lavoro qui presentato. Riportiamo poi gli studi condotti su ransomware negli ultimi anni, e come ci rapportiamo ad essi. Spieghiamo inoltre la motivazione che ci ha portato a questo lavoro, cosa ci poniamo come obiettivo, e le sfide incontrate nel tentativo di raggiungerlo.

- Nel capitolo 3 presentiamo nel dettaglio l'approccio definito e seguito durante il lavoro, e dunque come abbiamo deciso di raccogliere, trattare e analizzare i dati a nostra disposizione.

- Nel capitolo 4 descriviamo come l'approccio presentato Ã¨ stato tradotto in uno strumento, chiamato RansomScan, con tecnologie usate, strumenti di terze parti utilizzati, e i dettagli implementativi.

- Nel capitolo 5 presentiamo tutto ció che riguarda il processo di analisi: il dataset considerato, l'hardware utilizzato. Presentiamo infine i risultati ottenuti e le informazioni ricavate da essi.

- Nel capitolo 6 sono elencate le limitazioni del nostro lavoro, i problemi imprescindibili dal nostro approccio e che non possono essere risolti senza venire meno al traguardo fissato.

- Nel capitolo 7 forniamo spunti su possibili sviluppi futuri sullo stesso argomento, e come il nostro strumento puó essere usato per migliorare i risultati.

Contents

Chapter 1

Introduction

During World Health Organization's AIDS conference in 1989, the biologist Joseph Popp handed out to the unsuspecting attendees a few thousand floppy disks labelled 'AIDS Information - Introductory Diskettes', containing the first ever documented ransomware, the AIDS Trojan [1].

Ransomware is the name given to a category of malicious software whose goal is to prevent the victim from accessing the infected machine for its standard usage, usually encrypting user's private files, and asking for a ransom (hence the name) to be paid in order to obtain the decryption keys. That first rudimentary attack gave birth to a new category of malware that has exploded six years ago and has grown considerably since then [2]. Figure 1.1 represents the explosion of the number of new ransomware families, mostly after 2016.

According to threat reports, published by main cybersecurity companies, in the last three years more than 600 million ransomware attacks were reported, from hundreds of different families, and for approximately 5 billion dollars of estimated economic damage [4, 5].

While it is possible to make in depth statistics about infections, it is typically more difficult to get information about the cybercriminals behind them. This lack of knowledge is mainly due to the use of anonymization techniques: cybercriminals hide destinations of the connections via Tor routing [6], and make transactions untraceable using cryptocurrencies, specifically BitCoin [7]. Ransomware samples offer these as main payment methods, because the victim's transactions are fast, reliable, and verifiable. Furthermore, transactions are associated only with sender and receiver addresses, making them theoretically traceable, but making it impossible to go back to the owner of the receiving address.

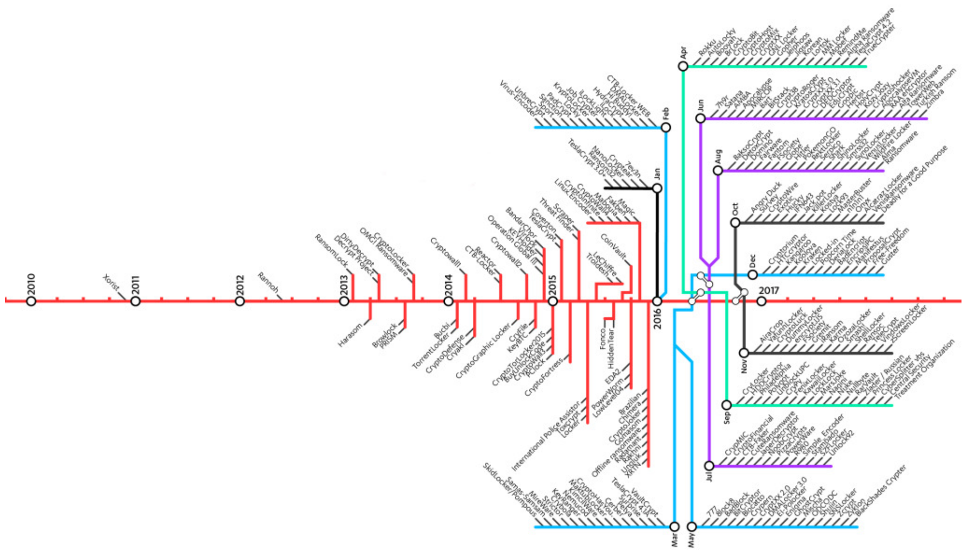


Figure 1.1: Evolution in Ransomware families [3]

Since the explosion of ransomware attacks in the last years, many researchers studied this phenomenon. Both university laboratories and big companies focused on different aspects of this kind of threat: from tracking transactions [8] to ransomware detection [9, 10, 11]. While it is certain that these studies are important and exhaustive in their sector, there is a lack of a definite approach when it comes to harvest data from samples; added to this is the necessity to gather information about the ransomware ecosystem behind the single sample. These motivations encouraged us to work on proposing a new methodology for ransomware analysis, from the design of the process to the development of the corresponding framework, aimed to extract all sort of useful data from post-mortem analysis of the samples, covering different sources of information. The idea behind this approach is to let the ransomware sample run in a controlled environment, then collect artifacts left from the execution, such as files, screenshot of the desktop of the emulated machine, memory dump, and examine the information left in order to find information useful for understanding the ransomware background and perform investigation (e.g. payment addresses, email accounts, URLs of malicious websites). Given the multiplicity of different sources of information analyzed, it is impossible to treat every source in the same way. We considered this aspect, and treated every source differently so we could make the most of it, and we changed analysis procedure for every of them accordingly:

- Text Files: We collect text files generated at runtime, of any format, but we keep only the readable ones.
- Images: Images are processed in order to facilitate text extraction: we apply resizing and custom binarization, then we use OCR techniques to extract text.
- Memory: We parse memory dump of the entire sample execution looking for readable strings, that form the corpus to analyze.
- Network packets: We sniff every packet generated by the sample, we detect type and content of it, and then if possible the packet is decoded.

Every piece of information found during analysis is then collected and we generate a report that is readable for people and parsable by other software. While ransomware has always been analysed in detail in his technical side, or divided by family, we propose here an innovative approach because this is the first time we try to extract clues for tracing people behind them in a most general and complete way. The described approach translates into an open-source framework, called RansomScan, that has been developed using state-of-the-art technologies. One of the distinctive traits of RansomScan is its ability to autonomously extract intelligence information from ransomware samples, exploiting different kinds of artifacts. We evaluated RansomScan analyzing 532 samples of different ages and belonging to different families: among these we detected that 313 were not executing, leaving us with 219 actually available samples. We were able to extract 117 payment addresses and more than 181 relevant URLs, thus extracting useful information from all the working samples.

In order to help the reader, we report here how this thesis is structured:

- In Chapter 2 we give an overview of the notions useful to fully understand the work presented here; we then report the studies conducted on ransomware in the last years, and how we relate to them; we finally state the motivation that lead us to this work, what we set ourselves as a goal, and the challenges met trying to achieve it.
- In chapter 3 we show the proposed approach, and thus how we collect, treat, and analyze data at our disposal.
- In chapter 4 we describe how we translated the approach presented in a tool, named RansomScan, with technologies used, tools included, and implementation details.

- In chapter 5 we present everything concerning the analysis process: dataset considered, hardware used; finally we present the results obtained and the insights derived.
- In chapter 6 we discuss the limitations of this work, the intrinsic problems of our approach that could not be solved without compromising the goals set.
- In chapter 7 we give ideas on possible future works on the same topic, and how to make use of our tool in order to improve results.

Chapter 2

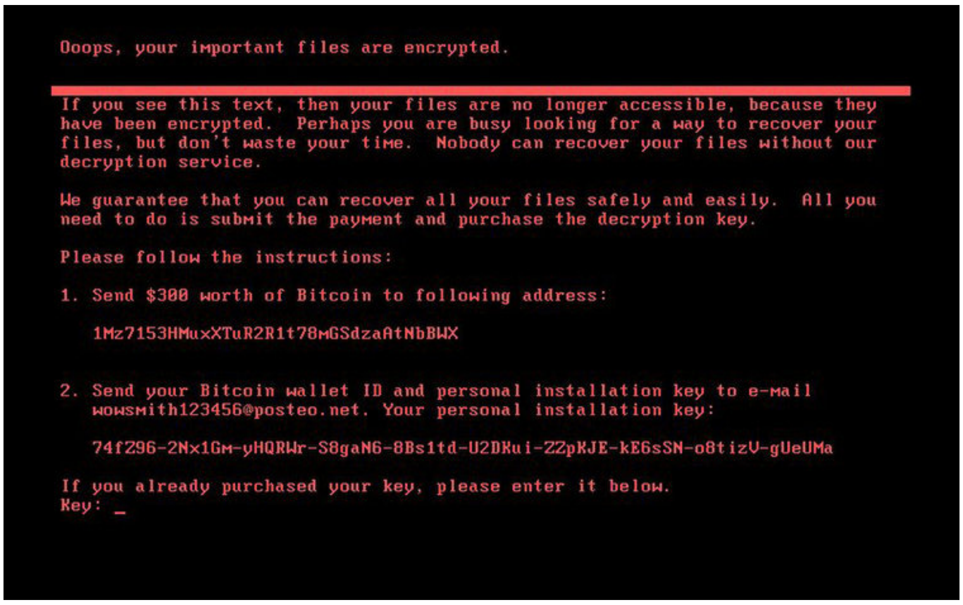
Background and Motivation

2.1 State of the Art

In order to understand the context in which this work is placed, and the proposed approach, it is important to have an overview on how ransomware behaves, and the economy behind this phenomenon. In Section 2.1.1 we describe in detail what is a ransomware, and give an overview of the ransomware ecosystem. In Section 2.1.2 we present BitCoin, the most common cryptocurrency used for ransoms payment. Then, in Section 2.1.3 we describe Tor network, that hosts most of the traffic generated by ransomware. In Section 2.1.4 we report the definition of Optical Character Recognition, and the techniques implemented for this task. We conclude with the overview on ransomware research: in Section 2.1.5 we present the state of the art regarding ransomware, and studies related to our work.

2.1.1 Ransomware

Ransomware is a specific category of malware that prevents users from accessing their files and demands a ransom to obtain them back. We can classify them in two different subsets. On one side we have locker-ransomware, that simply denies access to the device standard functionalities but does not influence files present on the system; the system is unresponsive, and a message is shown to the user asking for the ransom, to be paid usually in BitCoin or a similar cryptocurrency. This message uses social engineering techniques in order to scare the user and force the payment: they often include fake police messages announcing alleged crimes. On the other side we have cryptoransomware that, instead, encrypts user files, making them inaccessible even after malware removal. After the infection is completed, the



```
Ooops, your important files are encrypted.

If you see this text, then your files are no longer accessible, because they
have been encrypted.  Perhaps you are busy looking for a way to recover your
files, but don't waste your time.  Nobody can recover your files without our
decryption service.

We guarantee that you can recover all your files safely and easily.  All you
need to do is submit the payment and purchase the decryption key.

Please follow the instructions:

1. Send $300 worth of Bitcoin to following address:

   1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX


2. Send your Bitcoin wallet ID and personal installation key to e-mail
   wowsmith123456@posteo.net. Your personal installation key:

   74fZ96-2Nx1Gm-yHQRWr-S8gaN6-8Bs1td-U2DKui-ZZpKJE-kE6sSN-o8tizV-gUeUMa

If you already purchased your key, please enter it below.
Key: _
```

Figure 2.1: Example message from NotPetya ransomware

malware shows a message to the user asking for the payment, like what happens in the locker-ransomware case. One example of this kind of messages is shown in Figure 2.1.

This message is intended to warn the user about what happened to the system, and usually explains the user how to pay the ransom. It often includes coordinates for the payment, and a timer that shows the time remaining until the irreversible data loss, in order to create pressure. Given the conjunction of manipulation techniques and the severity of the damage dealt by this last type of ransomware, victims are more likely to pay for the requested fee. This higher earning probability makes cryptoransomware more lucrative [12], and thus more common, than their locker version.

During the last years we have observed an evolution of ransomware, from the first rudimental attacks to more sophisticated ones. The concept remained the same, but the techniques of infection changed, as well as the capabilities of the different ransomware families. The first major change was the targets of the attacks. Next to large-scale attacks aimed to infect as many victims as possible, more targeted attacks appeared, directed toward companies and organizations [13]. The use of software-specific exploits has been added to the infection methods, side by side with the well-known malicious emails: one remarkable example can be found in WannaCry ransomware attack that exploits a vulnerability present in the implementation of SMB protocol in various versions of Windows.

The encryption techniques for files changed as well: from flawed symmetric encryption algorithms, that let victims recover files extracting the key directly from the binary of the malware sample, ransomware converted to a hybrid encryption algorithm using two keys, one symmetric and one public. In this way the attacker only needs to send the public key with the sample, that encrypts the files using a symmetric key generated during execution, and encrypts the used key with the received public key. The victim is left with the cyphertext of the key, and can only send it to the attacker in order to obtain the decrypted symmetric key, essential to retrieve the files. This combination of different cryptography techniques makes the decryption of the files almost impossible without paying the ransom. Encryption mechanism is often regulated by a Command and Control server that is often in charge of controlling ransomware activation and sending the public key to the infected computer.

The massive diffusion of ransomware attacks started a research toward countermeasures to take in order to minimize the damage. The obstacles given by encryption make a posteriori solutions useless: without a backup it is hard to retrieve corrupted files. The only effective way to counter ransomware is to act proactively, trying and detecting malicious activity before it terminates. Modern antivirus software, even implementing different detection techniques, have several limitations and they are not always effective against new emerging families. One example of detection techniques used is represented by signatures exploitation. Signatures are defined as features of a malware, such as series of bytes in the code, or cryptographic hash of certain portions of it. Since this kind of features is unique to each sample of malware, signatures are constantly collected and used to identify the sample found and discern malicious software from benign one. Another important detection technique is the so-called behavior analysis that uses common operations within malicious software, like attempts to discover a sandbox environment, disabling security controls, installing rootkits, and uses them to label a program as suspicious and block it.

2.1.2 BitCoin Transactions

BitCoin has become the standard form of ransom payments, due to its pseudo-anonimity properties. Conceived for the first time in 1998, BitCoin is the first decentralized peer-to-peer payment network [7]. It is a virtual currency, and all transactions ever made are stored in the so-called blockchain, a public ledger where every money exchange is documented. It is designed to be secure, not subject to manipulation by single entities or centralization of power. In practice, every transaction, in the form “payer X sends Y BtC to payee Z” is broadcast to all nodes in the network. Every node, possessing a copy of the ledger, can validate the transaction adding it to a block. It then proceeds to broadcast the new version of the ledger with the block appended to all the other nodes: after this operation the transaction is official and recognized by all the network. Every block is created by the so-called process of mining: every time a block is created, it must contain a proof-of-work, i.e. an arbitrary value whose calculation requires the computer a certain amount of work. This is a measure to prevent spam and denial-of-service attacks, making the currency more trustworthy.

Transactions using BitCoin as a currency have several advantages: thanks to the structure of the network it is possible to transfer money anywhere within a short span of time, with neglectable fees. Furthermore, transactions are pseudonymous: every wallet is not tied to owner’s identity, so even if all transactions are public it is difficult to associate a payment to a real person. Given the possibility of creating transaction with multiple input and output addresses, and with the advent of mixers (services that help obscuring trails left from potentially identifiable funds), tracing money through the ledger has become hard.

2.1.3 Tor Network

Tor (that stands for The Onion Router) [6] is a project born with the idea of creating a free tool for online anonymous communication. This is achieved directing internet traffic through a worldwide overlay network consisting of more than seven thousands relays. Packets are encrypted with a tecnique called “Onion Routing”: every message exchanged through the network is wrapped with a number of successive layers of encryption equal to the number of the nodes that the packet must visit before reaching its destination. Every time the packet reaches a node, only the outermost layer is decrypted, and the current node receives only information about the previous node and the node that will be visited next by the packet. Both the previous and the next nodes are not characterized, so that the current node does not know if the packets comes from the origin, or it will reach the destination with the next forwarding. Only the destination node will receive the original content of the packet. Wrapped in this way packets are bounced between relays multiple times, reducing drastically the likelihood for sites to trace actions and identity back to the user. The path followed by packets through the network is randomly generated by the sender every time, so it is virtually impossible to follow back the data stream to the original user. The traffic viewed from the destination appears in fact to come from the last node visited (called exit node). Given the difficulty in tracing this kind of traffic, attackers widly uses Tor Network to host command and control servers, or web pages to guide the user through the steps necessary to pay the ransom.

2.1.4 Optical Character Recognition

Optical Character Recognition (OCR for short) refers to the conversion of images containing text, electronic, printed or handwritten, into digital text. Early examples of OCR technologies involved creating reading devices for the blind [14], but with the diffusion of more accessible technologies and tools, OCR became used for various application. The goal is to digitalize text in order to allow different kinds of processing (editing, searching, storing, etc.).

Currently text recognition is performed using many different techniques: Pattern Matching [15], for example, converts each character read in a pattern of ones on a matrix, and then compares it with known patterns previously built from character models. Other techniques involve extraction of different features from images and confront the obtained information with the known set; examples of such attributes are height, width, density of lines, loops or straight lines. The technique that is now widely used exploits neural networks [16] in order to simulate the way the human brain works. It takes the features extracted in precedent approaches as input, and uses them to recognize the possible characters shown in the image.

Best OCR tools commonly used are mainly proprietary software [17], developed by specialized companies, but it is possible to find also open-source projects, like Tesseract [18] or Kraken [19].

2.1.5 Related Work

Many studies have been conducted about ransomware. Given the difficulties in restoring systems status after an attack, a great section of the work done is focused on prevention, and in particular ransomware detection before or during encryption, in order to stop attacks before completion. Following this concept, recent techniques that have been conceived and developed for the first time by Kharraz et al. [20], are based on the idea of using activity of the filesystem as a solid ground to understand the behavior of analyzed software. They demonstrated that with an examination of file system activities, looking at I/O requests and monitoring Master File Table entries, it is possible to detect and prevent a significant number of ransomware attacks. Following this idea, Continella et al. developed ShieldFS [9], an add-on driver that makes Windows immune to ransomware, using a copy-on-write mechanism in order to keep an always-fresh, automatic backup of the files modified in the short term. This tool is composed by two different layers for both detection of ransomware and reversion of modifications on the attacked system. The first layer is in charge of monitoring I/O request packets to detect malicious activity, while the second operates periodic backups in order to allow files restoration after possible attacks. In this way they found a very efficient method to avoid common attacks. The system, however, will be still vulnerable to multiprocess malware, that wouldn't be detected since it spreads activity on different processes, or advanced encryption techniques. In parallel to this work, other detection approaches, respectively UNVEIL [21] and CryptoDrop [22], were developed. UNVEIL is based on the idea that in order to carry on an attack, a ransomware sample must tamper with a user's files or desktop. The tool automatically generates an artificial user environment, and detects when ransomware interacts with user data. In parallel, the tool tracks changes to the system's desktop, that may indicate ransomware-like behavior. The biggest limitation is that a malware could detect the artificial environment created, or detect dynamic analysis going on, and thus stop or slow down malicious activity. CryptoDrop, on the other hand, tried to classify malware activity in different major behaviours, and checks if new processes belong to one of those classes, allowing early warnings for ransomware activity. The downside of this method is that there is no way to understand the intent in determinate actions, for example it is not possible to determine if an encryption has been issued by an attack, or intended by a user. While these works help us understanding better ransomware behaviour and mechanics, they cover a different side of the ransomware analysis, the prevention of attacks. In our case we will actually infect systems on purpose in order to collect insights of the ecosystem with post-mortem analysis.

Another macro-area of study is focused on understanding the economy behind ransomware, where do payments go, how to trace them. This lack of information led to the study of BitCoin-specific tracing methods, with the purpose of keeping track of suspicious movements of funds. Meiklejoh et al. were the first to try and follow BitCoin transactions with a forensic approach [8]. They concentrated on a small number of manually labeled transactions, and they were able to identify major institutions and the interactions between them, and demonstrated that this approach based on heuristics can shed light on the structure of the BitCoin economy and how it is used. Another, more statistical, approach is presented by Ron and Shamir in [23]. Here we have an overview of the numbers involved in BitCoin network and the structure of the transactions flow, in order to better understand the phenomenon. The first attempt of semi-automatic analysis of the blockchain comes with BitIodine [24], a tool which parses the blockchain, clusters addresses that are likely to belong to a same user (or group of users), classifies such users and labels them, and finally visualizes complex information extracted from the network. We can consider this kind of study as the main beneficiary of our work: extracting malware addresses automatically, we generate information about wallets owned directly or indirectly by attackers, helping detecting payment patterns, aggregation wallets, and allowing better tracing of ransoms payments throughout the network.

The lack of ties between ransomware attacks and real people is not an unfamiliar problem and since the ransomware explosion many approaches to fill this knowledge gap have been studied. The most relevant and comprehensive is the tracking of ransomware payments end-to-end conducted by Google [25]. They conducted a study lasted two years and similar to the one showed here. They collected information on infected machines in order to find the BitCoin address showed for payment, and used it to trace ransoms on the BitCoin network. The differences between their study and ours are many: first of all, they focused mainly on the tracing part, and their information collection was limited to wallets address search. We focused instead on the collection and extraction part, not limiting ourselves to wallets but extracting all kind of interesting information left by the malware, specifically focusing on everything that could be helpful for intelligence. After that, while they focused only on certain ransomware families, we tried to maintain a high level of generalization, in order to design an approach valid in all cases, making the resulting framework useful for all kind of samples, even for future ones.

2.2 Motivation

The problem that led to this study is the lack of a detailed and automated approach for information collection about the world behind the ransomware proliferation, from the real identity of the attackers to the sources of infection. During years of research, there has been a greater effort in studying the techniques used by malware samples rather than in discovering more about the ransomware ecosystem and transactions. Gathering as much information as possible is helpful to go back to the root of the attacks. With more information available it is possible to prevent the spreading of the infections, or in some cases even find and stop the people carrying on the attack.

2.3 Challenges and Goals

We aim to give our contribution to the research and collection of data regarding connections between ransomware and the real attackers, and for this purpose we want to provide a tool for intelligence and investigation that extract all possible endpoints left from attackers: BitCoin addresses, email contacts and URLs. This tool is conceived to be used as a black box, receiving one or multiple samples as inputs, and returning a report with all data extracted. We want it to be easily customizable, in order to let users select only certain functions over the complete analysis. The development of this tool is not trivial, given the challenges that researchers have to face in this field. The first critical issue we had to face is the heterogeneity of information: we have to consider all different possible sources and formats, in order to be sure to extract everything from the sample without losing important details. Given the quantity of information shown by ransomware via the message left on victims screens, it is worth focus in detail on extraction from images: given the OCR state of the art, and the variety of ransomware interfaces, it is impossible to achieve high accuracy in recognizing useful parts of the text from images. Another important issue is represented by the difficulty of discerning information actually important and left by the malware from the multitude of data of the same kind already present on clean systems. For example, it is difficult to automatically distinguish between a common URL address and a malicious one, without human intervention. This causes several cases of false positives that we tried to reduce during our work.

Chapter 3

RansomScan: Approach

In order to confront, at least partially, the problem stated in Chapter 2, we aim to extract information from ransomware samples that are useful for identification and tracing of attackers.

We developed a framework, named RansomScan, following a precise approach described in this section. The idea is to receive as an input one or more ransomware samples eligible for analysis, run them in a supervised environment in order to let them start their malicious behaviour, then extract the artifacts left in the environment. Finally the results are returned in a readable way, useful for both human understanding and further automated operations. Given the multiplicity of information sources that we aim to analyze, and since every source is associated to a different kind of extraction method, we decided to separate the approach in phases, for better understanding of the process. This will result also in a modular structure of the system. The modularity of the resulting tool makes it open to customization: if the usage as a black box for analysis is not sufficient, it is possible to insert the separate modules in a different customized pipeline.

3.1 Approach Overview

In Figure 3.1 we represent the framework architecture. We receive one or more ransomware samples as inputs, and we perform dynamic analysis on them, as described in Section 3.1.1. The output of this first phase is represented by the set of artifacts that we manage to extract, i.e. files and images created, screenshots, network packets and the memory dump. These artifacts are collected, in the way presented in Section 3.1.2, and become the input of the pre-processing phase, described in Section 3.1.3.

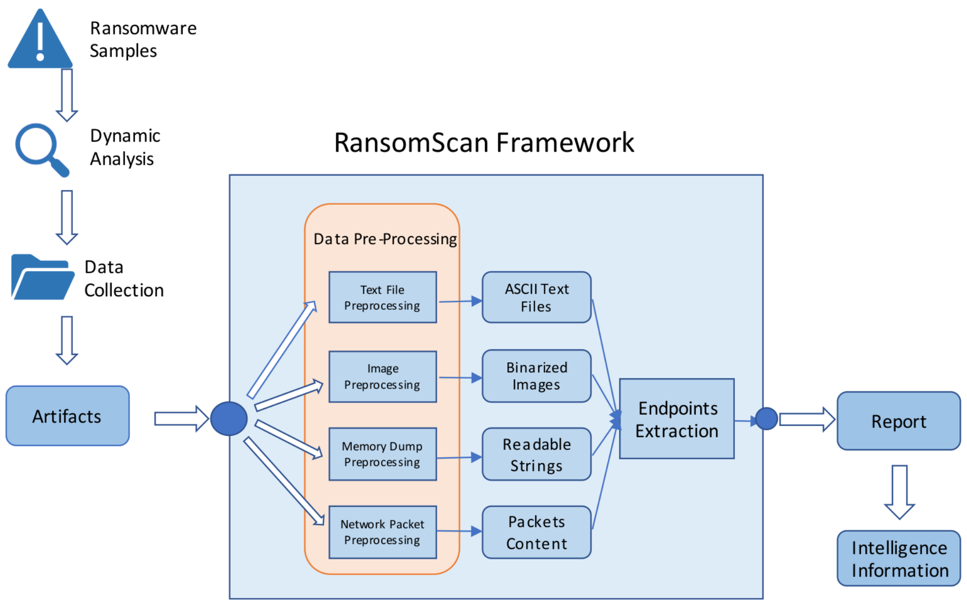


Figure 3.1: Framework Architecture

The pre-processing phase represent one of the main parts of the core of the framework: in this phase we operate a selection and preparation of the artifacts for the actual analysis. We are now able to properly analyze data. We perform this operation following the procedure defined in Section 3.1.4, and after this phase we collect the results in a structure that is easy to read and access, as explained in Section 3.1.5. The output of the collection phase represents also the output of the whole process of analysis.

3.1.1 Dynamic Analysis

Every ransomware sample is executed in a controlled environment in order to let it operate and create the characteristic artifacts to be analyzed. We use a sandbox in which we let malicious process running while we monitor its behaviour. This is performed letting the virtual environments communicate with the host in a client-server fashion, where the host behaves as a server, and messages are exchanged through the network. In Figure 3.2 we show the infrastructure model used for dynamic analysis. Inside every machine we install an agent, that allows comunication between host and guests using packets sent through the network. This agent uses API hooking to trace API calls, in order to check what kind of operations the process is doing. The agent is also in charge of keeping records of file operations, so it is also possible to control operations like file creation, deletion, and download, all useful

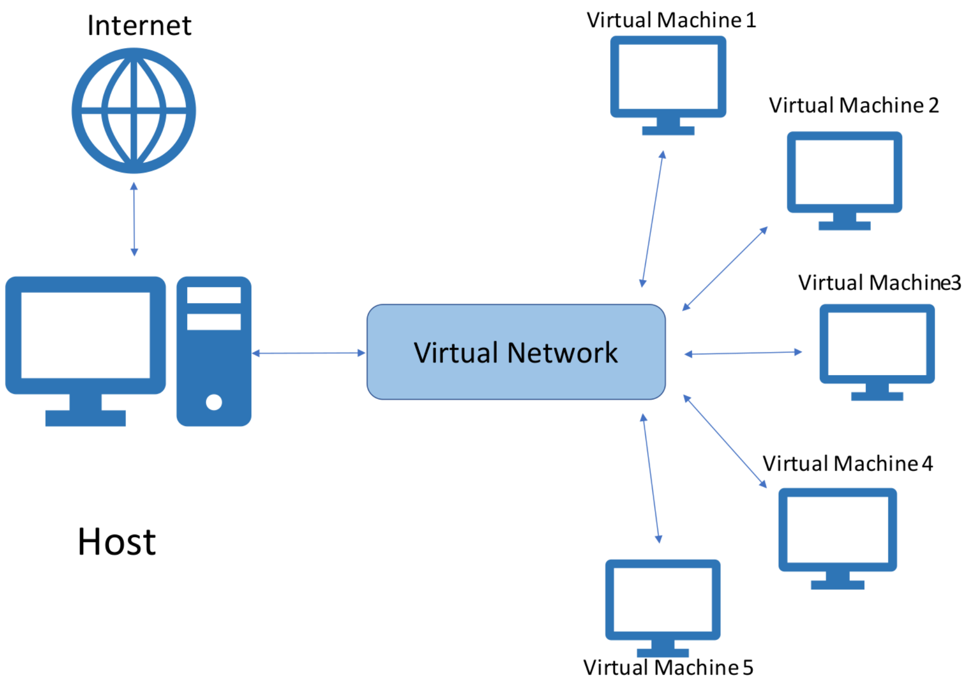


Figure 3.2: System Architecture for Dynamic Analysis

for the aforementioned artifacts collection. Every detail of the execution is sent to the host, that is in charge of collecting packets containing logs of the operations performed by the malware, the files created on the machine, and the screenshots taken. During execution, in fact, we supplement data left by the malware with a series of screenshot, taken every time there is a significant change in what the desktop is showing. We also intercept traffic from and to the controlled environment, in order to monitor any communication between the sample and an external source. The network traffic is routed through the virtual interface that allows us to control traveling packets.

On the virtual systems we install components useful for monitoring such as the sanbox management agent, create fake files, and visited different websites in order to try and emulate a real system usage. We also change the default values set for the virtual machine in plausible values of a real computer. This is performed in order to make the system indistinguishable from a real one, and thus avoid malware evasion controls. This process is replicated on multiple identical virtual machines: in this way we can run parallel analyses and thus increase the process throughput. After these operations, we save the state of the system in order to revert to it every time an analysis is performed, avoiding manual restoration of the machine status, and thus reducing the time between two consecutive analyses on the same system. Before starting any analysis, we create a baseline memory dump applying the whole dynamic analysis described above to an harmless program, in order to list all information already present in the operative system, or that is generated during standard execution. Then, at the end of execution the hosts generates the memory dump, and the file with the captured network traffic.

3.1.2 Data Collection

In this phase we collect data left from ransomware execution from the different sources: files created or modified, screenshots taken during execution, memory dump of the process, network traffic. Files created are collected using the agent controlling the sanbox, that transfers all new files via networking to a dedicated folder. The agent can also control the desktop content in order to take the screenshots, that are transferred, in same way as files created, to a specific folder on the sandbox host. Network traffic is captured directly by the host, that acts like a man-in-the-middle and allows us to filter and log every packet. We also collect the memory dump, that is saved at the end of the process execution. We elaborate every one of the sources in a different part of the process, described more in detail in the dedicated sections.

3.1.3 Data Pre-Processing

The data gathered in the previous step is divided in groups based on the file type, in order to process every typology in different specific ways and exploit them to extract as much information as possible.

Text Files Processing

The fundamental operation for this category of files is the separation between readable and non-readable files: the distinction is based on the file header, that reveals file type, and character encoding. In this way we select only files containing ASCII characters. In this case we don't alter the content of the single files.

Images Processing

We process images, coming from files or from screenshots, before trying to extract text from them. We first filter them in the same way we did for text files; in this case we are interested in the standard images format, that are more likely to contain useful information. Images selected pass through a pre-processing phase that consist in resizing and binarization: we enlarge them in order to reduce the amount of information loss during manipulation. Then we apply a mode filter and we convert the image in greyscale. Finally we implement a custom binarization process in order to increase OCR performance. We parse the image selecting sequentially small windows and applying this heuristic operation: we convert to white the most common color in the window, and to black the rest of the colors. This is based on the assumption that the most common color in the selected area is most likely to belong to a background section, while the rest can be considered content. In this way we achieve the conversion of the image from colored to black-and-white, with black content on white background. In order to clean possible imperfections, we apply another filter in order to sharpen the image, and the image is now ready for analysis. In Figure 3.3 we confront the original image, containing the screenshot of a ransomware message, with the binarized image after the process. The image now contains only black content and elements on a white background. While the result is not perfectly clean, it is sufficient to help character recognition.

Memory Dump Processing

We extract the memory dumps coming from malware execution, decoding them, and searching in them for strings composed of more than four readable characters. We then operate a differential analysis comparing the strings found in this way with the ones extracted in this same way from the baseline dump. In this way we can filter out information present on the memory by default and select almost only relevant information, left on the system by the malicious process.

Network Packets Processing

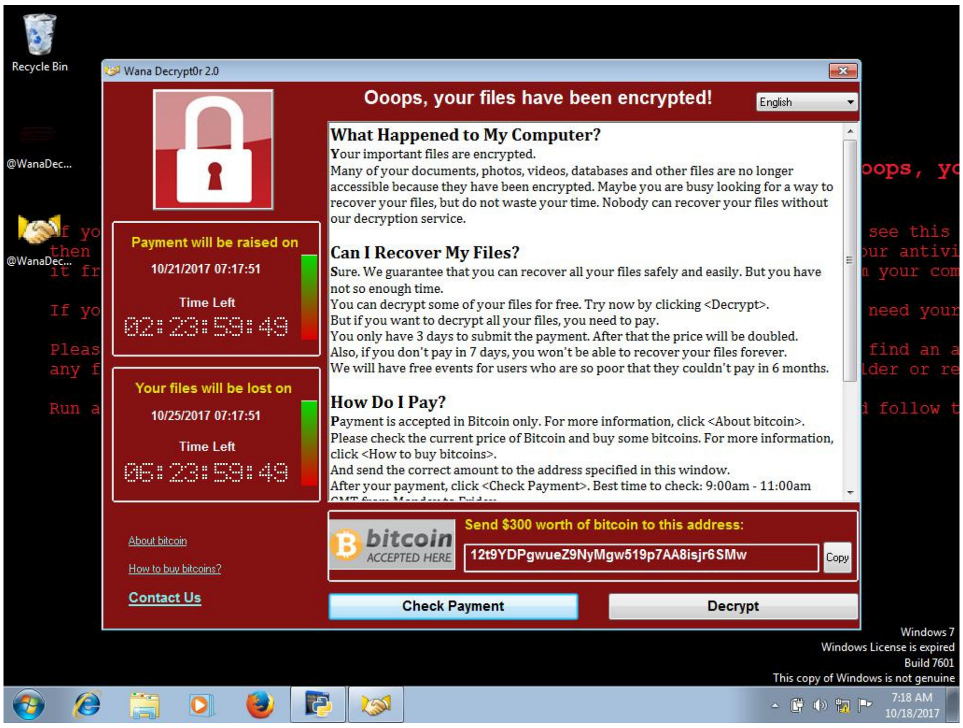
We also open and dissect captured network packets: we make a list of the sessions captured during network sniffing, and for each packet exchanged we use the HTTP headers to understand if it is a packet containing content, and decode the packet payload. We can continue parsing the responses obtained searching for potential information exchanges between the sample and any external source.

3.1.4 Endpoints Extraction

We analyze every processed piece of information obtained so far looking for information. This is straight-forward for text files, since we need to parse files searching for portions of text that have the typical structure of BitCoin addresses, URLs, or mails.

Images need an intermediate step: using OCR techniques we convert the content of the image in a text file, and we analyze it in a way similar to the one described before. The OCR technique preferred in this case follows a specific processing pipeline in order to extract text with minimum error. The image is first analyzed in order to identify text blocks. Every block is then selected and analyzed in order to detect text lines, found by fitting baselines. Each line is then enriched with more lines, used to follow top and bottom borders of the text, and catch possible curved lines. This is not important in case of computer images, but necessary when working on scanned images. The following step of the procedure breaks down every line in single characters, measuring an approximate distance between them and chopping in the middle. In case of overlapping characters different cuts are tried, keeping only the one that improves confidence. After these steps, we need to recognize characters: it is done by extracting features from characters, and confronting them with the features associated to each character in training phase. While it is common to enforce classification using a dictionary of common words as support, in order to avoid one-character errors and recognize correctly words belonging to a specific language, it is important for our analysis to avoid this technique: since BitCoin addresses are composed of a sequence of characters that do not form a meaningful word, this might introduce a bias in recognition and thus lead to more frequent errors. For text coming from images we use a slightly different information extraction process in order to be sure not to miss any detail, and reduce error probability: we keep both valid and not valid addresses and we save them with a label indicating if that address is valid or not, in order to introduce some tolerance, given the error probability of the OCR process. It is also useful for the user since he can see that an address might still be present, and evaluate specific cases properly.

The same technique used for text files is then applied also to text found in memory and in network packets. After the analysis all results are filtered: we validate BitCoin wallets found and remove possible false positive addresses, we keep only URLs generated by malicious activity, often pointing to unknown or uncommon domains. All results are then collected and saved in a report that is available and readable both for a human user and any other piece of software extending the pipeline.


Recycle Bin

@WanaDec...

@WanaDec...

Wana Decrypt0r 2.0

Ooops, your files have been encrypted!

English

What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Payment will be raised on
10/21/2017 07:17:51
Time Left
02:23:59:49

Your files will be lost on
10/25/2017 07:17:51
Time Left
06:23:59:49

bitcoin ACCEPTED HERE
Send $300 worth of bitcoin to this address:
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
Copy

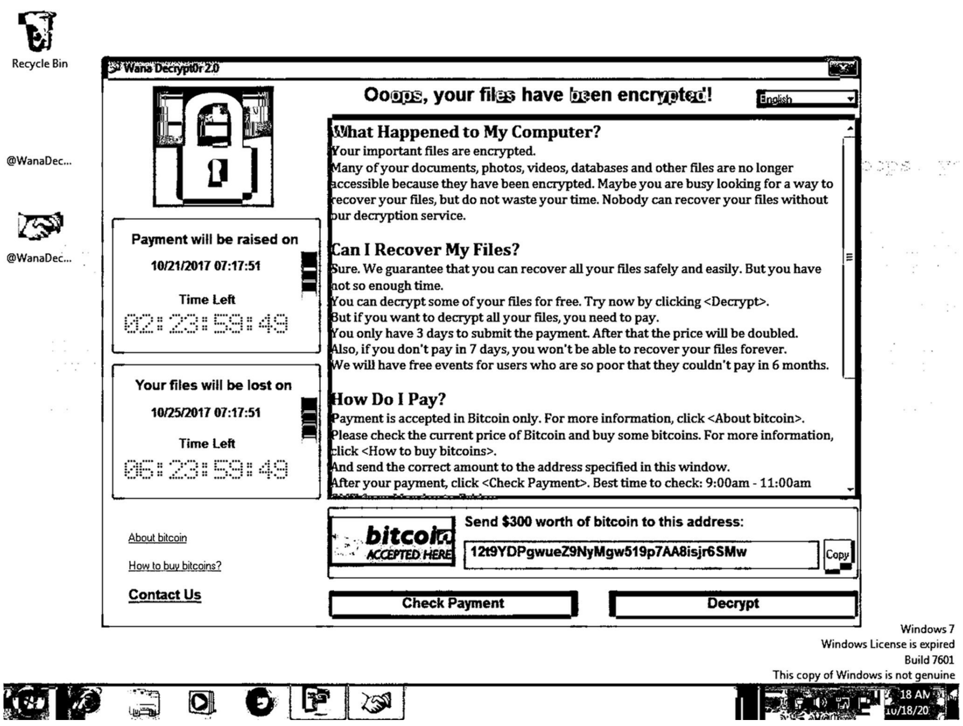
About bitcoin
How to buy bitcoins?
Contact Us

Check Payment
Decrypt

Windows 7
Windows License is expired
Build 7601
This copy of Windows is not genuine

7:18 AM
10/18/2017

(a) Original image


Recycle Bin

@WanaDec...

@WanaDec...

Wana Decrypt0r 2.0

Ooops, your files have been encrypted!

English

What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Payment will be raised on
10/21/2017 07:17:51
Time Left
02:23:59:49

Your files will be lost on
10/25/2017 07:17:51
Time Left
06:23:59:49

bitcoin ACCEPTED HERE
Send $300 worth of bitcoin to this address:
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
Copy

About bitcoin
How to buy bitcoins?
Contact Us

Check Payment
Decrypt

Windows 7
Windows License is expired
Build 7601
This copy of Windows is not genuine

18 AM
10/18/20

(b) Binarized image

Figure 3.3: Comparison between the same screenshot of a ransomware note, before and after the binarization process

Chapter 4

RansomScan: Implementation

4.1 System Architecture

The approach described in Section 3 is implemented directly in a framework that collects all the different techniques for endpoints extraction. The framework, named RansomScan, is designed to reflect the phases of the approach and the division between the techniques using a modular structure. In this way the analysis can be conducted either using a pipeline that automatically returns the results of the full extraction, or using the single modules in order to focus on specific parts of the process. This is helpful in terms of scalability: we can conduct different analyses in parallel, and reduce considerably execution time and computation complexity in case of specialized analyses. Given the modularity of the system, we will discuss about the single modules in detail in the dedicated section.

4.2 Language and Tools

The first precaution we need to consider when treating malware is creating a safe, simulated environment in which to let the malware execute. The automatic execution of the sample and the dynamic analysis are performed via Cuckoo [26], an open-source, flexible system that can be easily integrated in the framework. The virtualization tool chosen to generate and prepare the virtual machines described in the approach is Oracle VirtualBox [27], given its high performance and the compatibility with Cuckoo. In our analysis we perform a large amount of OCR operations, and we choose to use different tools for this task, in order to achieve better results and make an evaluation of the performances. For this purpose, we adopted Google CloudVision, a commercial solution, and confronted it with Tesseract OCR and Kraken, both open-source. Processing and analyses on data are written entirely in Python. We decided to adopt it given its flexibility, its compatibility with the tools mentioned above, and the number of libraries coming with it. Along with the common libraries for specific operations (e.g. strings manipulation, regular expressions, multiprocessing management, etc.), we adopted PIL [28] for image processing, SqlAlchemy [29] and the PostgreSQL driver Psycopg2 [30] for database accesses, the Scapy package [31] for network packages manipulation, and the library Requests [32] for HTTP requests creation.

4.3 System Details

4.3.1 Preliminaries

As described in the approach section, we need to create the virtual machines used to automatize samples executions. We start from fresh virtual machines provided by Microsoft [33], with a version of Windows 7 already installed. We then proceed to install the dependencies and the agent to allow communications with Cuckoo, set a completely white desktop background to remove possible background noise, and execute different hardening algorithms. With this procedure we aim to change all the default values of a virtual machine in plausible values of a real computer. System details like BIOS or components, usually checked by malware evasion procedures, are now corresponding to existing machines and this makes it more difficult for malicious software to realize it is running in a virtual environment. In this way we have appliances that simulate a real system, reducing the probabilities of being detected by malware during execution. After all the operations we take a snapshot of the running VM (either with GUI or using VboxManage command); now Cuckoo can reset the VMs to a clean state after every sample execution, reducing the number of necessary VMs and remove completely human presence in the analysis process. VM control is managed by Cuckoo using VboxManage commands provided by VirtualBox. Another important operation is the configuration of Cuckoo parameters: we increase the sample execution timeout to ten minutes, in order to avoid programmed delays in malware deploying; we also add keywords to the set that Cuckoo uses to interact with samples, making it click also on buttons labelled “connect”, “start” or “proceed”.

4.3.2 Dynamic Analysis

The framework receives the sample (or the folder with the samples) to analyse, and submit an analysis to Cuckoo. Cuckoo is in charge of collecting the files being created and downloaded by the malware, doing a full memory dump of the machine, sniffing the network packages, and taking screenshots. Cuckoo then stores all these artefacts in its storage folder, to which we access in order to perform our analysis. In order to achieve a higher grade of modularity and solve compatibility problems between different Python versions, we decided to start Cuckoo and submit analyses using native subprocess function. In this way we don’t need to include Cuckoo code in the tool. After submission, we collect the response of the command, we extract the task ID from it (i.e. the number corresponding to the analysis issued), necessary in order to access analysis results. With this value we access Cuckoo database in order to periodically check analysis status, as shown in Code Snippet 4.1 and know when it is possible to start information extraction. We set up a maximum number of database accesses at scheduled intervals, in order to avoid conflicts between parallel analyses, and catch possible unfinished or ill-terminated analyses. In particular we wait a fixed time of 8 minutes before trying to check the analysis status, and if the status is still pending, wait for a total of thirty seconds before trying again.

4.3.3 Files Analysis

All files created or downloaded by the ransomware sample during execution are identified in the log of the process execution, using the ID corresponding to the interested task, and selected in the storage folder using the file name. These files are parsed with the help of python-magic library [34]. This allows us to know the content of the file (either text, image or others), and thus separate and analyse them differently based on the content. We specifically search for text files containing non-extended ASCII characters only. In this way we can use three different regular expressions, implemented in Code Snippet 4.2, to search in the file for BitCoin addresses, email contacts and URLs. This is possible thanks to the peculiar features of every piece of information searched. BitCoin addresses are always composed of 26 to 35 alphanumeric characters, beginning with the number 1 or 3, and not containing the uppercase letter “O”, uppercase letter “I”, lowercase letter “l”, and the number 0, for visual ambiguity prevention. URLs are identified looking for strings with dots inside, possibly starting with “http” or “www”,

```python
time.sleep(480)
while res != "reported":
    try:
        time.sleep(10)
        query = con.execute("SELECT status FROM
           tasks where id = {id}".format(id=taskid
           ))
        rows = query.fetchone()
        res = rows[0]
    except:
        time.sleep(20)
        print("Waiting for analysis to finish...")
    if errcount > 45:
        mail("RANSOMSCAN","Task " + taskid + " non
             ha mai finito di eseguire.")
        return -1
    errcount += 1
```

Code Snippet 4.1: Periodic check for analysis result

and ending with a domain from 2 to 6 characters long. Finally email addresses are found simply looking for strings containing at least a character before a "@", at least a character after, one dot and at least one character composing the domain. This does not completely remove false positives, since email format is very generic, but limit the number of useless information that could be retrieved. We parse every line of the file looking for these kind of patterns, and collect all positive matches. Image files are analysed in a different way (see 4.3.3), while non-text, non-image files are discarded. In this phase no pre-processing of the files is used.

4.3.4 Images Analysis

We take all the images created by the ransomware sample and found in the previous step, as well as screenshot generated by Cuckoo, we try and open them using PIL library, encode them to base 64 and send them in a HTTP request (with the help of Requests library for Python) to CloudVision tool, that returns the first interpretation of the text contained in the image. Given CloudVision rates limit it is important to send requests with intervals between two consecutive images, we set it to 5 seconds for a total of twenty attempts, in order to try and get results as fast as possible without

```
btc_address = re.compile(u'(?<![a-km-zA-HJ-NP-Z1-9])[13][a
    -km-zA-HJ-NP-Z1-9]{26,33}(?![a-km-zA-HJ-NP-Z1-9])')
url = re.compile('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@
    .&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
mailaddr = re.compile('[A-Za-z0-9\.\+_-]+@[A-Za-z0-9\._
    -]+\.[a-zA-Z]+')
```

Code Snippet 4.2: Regular Expression Used

incurring in crowded servers. After that we use a custom binarization algorithm to convert the images in a version that contains only black and white. The algorithm, implemented in Code Snippet 4.3, operates in this way: it increases the image dimensions in order to lose the least amount of information possible during manipulations. It then applies a mode filter, used to reduce noise, that for every box of 2x2 pixels converts all the values of the pixels to the most frequent pixel value inside the box; after this the image is converted in greyscale. From the greyscale image we create two thresholds: pixel too dark will be set directly to black, and pixel too bright will be set directly to white. We then create a pixel window wide enough to always contain areas with different shades, and inside this window we assume that the pixel value that appear the most will be part of the background, while the rest will be part of text, border, or other elements. During our work we found that an optimal value for the window is 32x24 pixels, but this can change for different image formats.

With this binarization we aim to obtain an image with a full white background, with text and elements in black, in order to help text recognition by the OCR software. We feed the open-source OCR tools the image processed and collect the interpretations. In order to use the CloudVision API we need to send a request containing the type of analysis required and the encoded image. The specific request needed is "DOCUMENT_TEXT_DETECTION", and the image needs to be encoded in base 64 format. In order to confront the different OCR tools, we send the original image to CloudVision, with no binarization. The original image is also the input for Kraken OCR, and we let it work almost autonomously. We just set the steps it needs to perform, and so we indicate that we need to "binarize" the image, automatically "segment" it, and then perform "ocr". As we said before, we don't change the default parameters since we want to compare the performance of OCR tools out of the box, with the performance of Tesseract OCR with custom binarization and specific parameters. We feed Tesseract the already binarized image, and we set no language dictionary: it is important to reduce the use of dictionaries, since we are often looking for sequences of character with no meaning, and that bias could bring in some errors. We also indicate that the tool should perform page segmentation with orientation and script detection. This is necessary since ransomware messages are often composed of multiple text blocks around the interface, and we don't know how these blocks are placed and oriented, so we need to allow the maximum flexibility possible in order to correctly detect all of them. We parse each of the three different versions of the textual content looking for information, in a fashion similar to the one used for text files, using the same regular expressions.

4.3.5 Memory Dump

We take the full memory dump produced during execution and we detect every string of characters that could represent useful information. To do so, we mimic the Linux built-in function "Strings" [35], that find all printable character sequences that are at least 4 characters long. We created ad-hoc functions, shown in Code Snippet 4.4, that reads memory content and checks if it's composed of printable characters. In this way we extract strings contained in memory, in which we go through with the regular expressions already mentioned. In this phase we don't collect the results directly, we instead reduce the amount of unrelated data comparing the strings extracted from the examined memory dump with the ones extracted from a dump generated during the execution of a clean, not-infected VM of the same kind. The results left are then collected and added to the intelligence information collection,ready to be shown.

4.3.6 Network Analysis

Starting from the Pcap file generated after network analysis, we examine every single package, looking for content of text/plain type; this is performed checking the HTTP headers. After packets sorting, we can parse the payload of the selected packets with the regular expressions created for the different kinds of desired information.

4.3.7 Endpoint Extraction

All results derived from the above mentioned phases, are collected in a single data structure. We proceed with checking the presence of duplicates, and with BitCoin address validation. In functions reported in Code Snippet 4.5 we exploit the fact that we know from specifics that the last four bytes of the address a checksum check, and that they are the first four bytes of a double SHA-256 digest of the previous 21 bytes, so we can easily verify that the extracted address is a valid address or a meaningless string. This allows us to have no false positives when it comes to identify BitCoin wallets. Finally, every piece of information, along with execution time of the analysis (taken from the end of Cuckoo execution to this collection phase), sample name, and execution information, are formatted into a dictionary that is immediatly saved in a JSON file. This produces a report clear, easy to understand, and easy to access to allow further analyses by both human operators or software.

```python
def image_processing(image):

    width, height = image.size
    newwidth,newheight = (int(width*2),int(height*2))
    # Enlarge image
    image = image.resize((newwidth,newheight), Image.BICUBIC)
    image = image.filter(ImageFilter.ModeFilter(2))
    #Convert to gray scale
    image = image.convert(mode="L")
    for i in range(0,newwidth,PIXEL_WIN_W):
        for j in range(0,newheight,PIXEL_WIN_H):
            box = (i,j,i+PIXEL_WIN_W,j+PIXEL_WIN_H)
            # Select window
            crop = image.crop(box)
            pixel_values = list(crop.getdata())
            # Pixel too bright or too dark approximated
            pixel_values = [255 if x > 180 else x for x in pixel_values]
            pixel_values = [0 if x < 100 else x for x in pixel_values]
            # Count pixel frequency
            freq = dict((x,pixel_values.count(x)) for x in set(pixel_values))
            # Find most common colour in the window
            kmax = max(freq.items(), key=operator.itemgetter(1))[0]
            # Set most common colour (assumed background) to white, black otherwise
            pixel_values = [255 if(x==kmax) else 0 for x in pixel_values]
            tmp = Image.new("L",(PIXEL_WIN_W,PIXEL_WIN_H))
            tmp.putdata(pixel_values)
            # Overwrite new window on original image
            image.paste(tmp,box)
    # Transform operations to clean binarization results
    image = image.filter(ImageFilter.BoxBlur(1))
    image = image.filter(ImageFilter.SHARPEN())

    return image
```

Code Snippet 4.3: Custom Binarization Process

```python
def strings(fname, n=4):
    with open(fname, errors="ignore") as f:
        result = ""
        for c in f.read():
            if c in string.printable:
                result += c
                continue
            if len(result) >= n:
                yield result
            result = ""
        if len(result) >= n:  # catch result at EOF
            yield result
```

Code Snippet 4.4: Custom "Strings" Function

```python
def decode_base58(bc, length):
    n = 0
    for char in bc:
        n = n * 58 + digits58.index(char)
    return n.to_bytes(length, 'big')

def check_bc(bc):
    bcbytes = decode_base58(bc, 25)
    return bcbytes[-4:] == sha256(sha256(bcbytes[:-4]).
        digest()).digest()[:4]
```

Code Snippet 4.5: BitCoin Address Validation

Chapter 5

Experimental Setup and Validation

5.1 Goals

We performed four different kinds of experiments on RansomScan in order to evaluate our approach and the implemented framework. First, we showed that our system is able to correctly extract information from samples, without losing important traces and reducing the amount of false positives. To do so, we tested it on real samples coming from known collections and on fresh samples found in the wild. Also, our framework showed to be able to generalize and extract artifacts from samples belonging to different families. We tested it on ransomware samples from different families in order to verify the generalization capabilieties in presence of new samples. In order to choose the best OCR tool for this kind of analysis, we took some of the examined samples and manually checked if the OCR tools managed to successfully extract the text contained in the image, comparing performances in terms of accuracy. Finally, we tested RansomScan performance in terms of space required on disk, and time consumption, in order to give users an estimation about requirements to use the developed system.

5.2 Dataset

The dataset we used includes ransomware samples coming from different sources. The first batch of samples comes from the ShieldFS dataset [9], and it is composed by 382 samples, representing the test set used for ShieldFS study. The second batch, composed of 150 samples, is collected from Virus-

Total using the available Intelligence API [36]. In this way, we could obtain fresh and active samples. During analyses we found that a great portion of the samples did not produce malicious activity: given the age of several samples of the set, we discovered that these samples could not contact their server in order to start the encryption, so we could not use them since they are no more functioning and thus not showing data. We could not retrieve information from the connection attempts since analyzed samples tried to contact different addresses in a range, and since we did not get a response, we were not able to detect the exact server address. Because of this we excluded 205 samples of the original set. Another group of samples did not show activity in any case: we checked manually 30 samples that did not show signs of malicious activity, and we verified that during the analysis the sample did not create encrypted files, nor showed ransom notes, and we did not monitor unusual network activity. We suspect this is mostly due to evasion techniques carried out by malware in order to avoid information leakage, or a delay in activation too high to be detected by our analysis. Because of these reasons we had to exclude another part composed by 108 samples. From the union of the sources mentioned above, and removal of the inactive samples we obtained the group of samples that we can consider as the meaningful dataset we used for evaluation, obtaining a total of 219 working samples.

5.3 Experimental Setup

We conducted experiments on a headless 64-bit machine with two cpu Intel Xeon ES-2680 v2 2.80 GHz, and 377 GB of RAM. The virtual machines used for the controlled environment are instead 32-bit machines working with one cpu, and have 40 GB reserved for storage, and 1 GB of RAM. We report here the parameters for both host machine and virtual machines in order to describe the study setup, they are not mandatory in order to let the framework operate. In fact we used this setup in order to have the necessary power to run multiple analyses at the same time, but for a single execution of RansomScan even smaller machines would suffice.

5.4 Experiments

5.4.1 Information Extraction

RansomScan successfully extracted intelligence information from all the 219 working samples. This means that when ransomware altered the system in the analyzed samples, we were able to find and retrieve some kind of artifact. We noticed that these artifacts are mostly composed of cryptocurrency wallets or URLs. Specifically, we found 117 distinct valid BitCoin addresses, and more than 500 suspicious URLs. We noticed that it is very rare that different samples share the same BitCoin address. Specifically, we found two duplicated address: "1nNzekuHGGzBYRzyjfjFEfeisNvxkn4RT", found in three different samples, and "12NbRAjAG5U3LLWETSF7fSTcdaz32Mu5CN", found in two of the samples. Regarding URLs, it is difficult to find a way to autonomously declare an address as malicious or not, so we got different false positives. We still can spot Onion addresses inside the set, and we can be sure that those are left by the malware. We collected 181 addresses that are Onion URLs, but the number of possibly malicious links is higher, considering the whole unidentified URLs set. The situation is different for BitCoin addresses: since they have a specific structure and a deterministic way to determine if a supposed address is valid or not, we can be sure to retrieve only actual wallets, obtaining a high grade of accuracy. In Code Snippet 5.1 we present an example of report generated by RansomScan, based on a real analysis. We included analysis information like the sample name, a flag indicating the effective malware activation, the execution duration. After this section we listed the results, starting from image analysis results, divided by tool. Every result here is followed by a flag that represents if that address has been validated or not, so if it is correct. Then we have BitCoin addresses coming from other sources (text files, memory, network packets), and finally the lists of URLs and email addresses. We had to show in the example some of the most significative results for layout purposes, but the number of URLs and mails retrieved is higher in the real case. We divided the data collected according to the type of source file (text files, images, memory, network traffic), and we discovered that only a small part of the wallet addresses in images or shown on the screen is detected by OCR. Between all messages we detected only 5 BitCoin addresses. Most of them, together with URLs and mails were retrieved from text files generated or in memory. We had no significant results coming from network packets: this is expectable since communication between C&C server and the sample is usually encrypted. When packets are decrypted, if artifacts are present, they are then detected by memory analysis. In Figure 5.1 we show how the results are distributed between the different typologies of files.

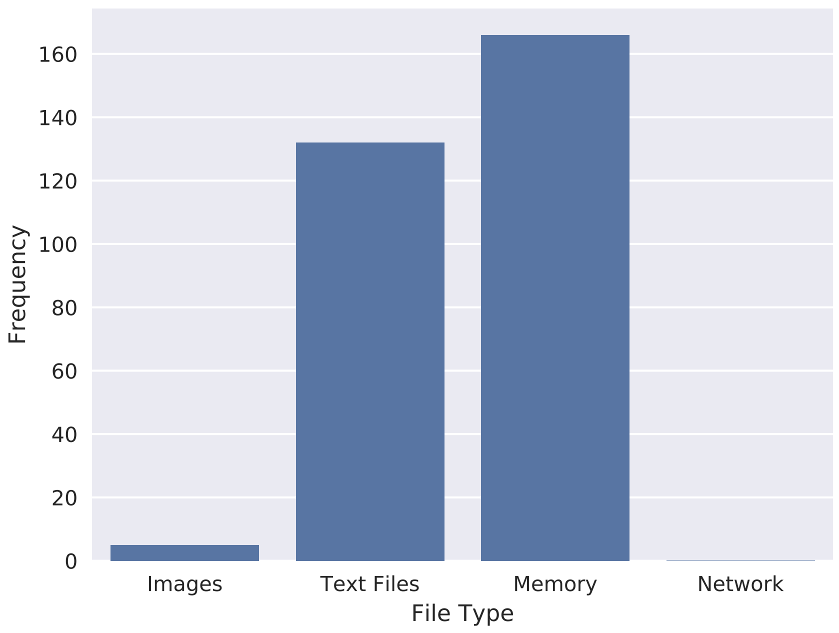


Figure 5.1: Sources analyzed, with the number of succesful extraction per type

5.4.2 Generalization Capability

Of the 219 working samples tested, we verified that intelligence information extracted in Section 5.3.1 comes from samples of 10 distinct ransomware types. This proves that we are able to extract useful information from samples belonging to different families, achieving the generalization we need in order to face future samples. In Figure 5.2, we report the different families analyzed with their distribution inside our working dataset.

5.4.3 OCR Comparison

For this comparison we selected 10 samples from different families that showed the BitCoin wallet address in the ransomware note on the screen, since not every sample gives you the payment coordinates directly, and evaluated the performance of OCR tools: we noticed that after the customized binarization of the images, Tesseract was able to recognize addresses in five cases, while CloudVision could extract address from the original image only in two cases. Kraken was never able to recognize a BitCoin address. The important thing to notice is that while being able to recognize the fact that there was an address, neither Tesseract nor CloudVision managed to read

```
{
"sample_name": ["d1a5d938880f8c8a8fac9da1e69d46224ef34da5c470fdaa6412c-f5832fc2694"],
"executed": ["1"],
"elapsed_time": ["556.7990386486053"],
"cloudvision_addresses": [],
"kraken_addresses": [],
"tesseract_addresses": [{
        "address": "144o6waEixJanwaLKGr97T4tjyepng",
        "valid": 0}],
"other_addresses": ["144o6wDdEixJJnzwMfLKGr97T4tjyepdnV"],
"extracted_urls": [
      "http://l.twimg.com/i/hpkp_report",
      "http://ocsp.infonotary.com/responder.cgi0V",
      "http://th.symcb.com/th.crt0",
      "http://www.signatur.rtr.at/de/directory/cps.html0",
      "http://pesquisa.sapo.pt/livesapo?q=",
      "http://www.e-szigno.hu/SZSZ/0",
      "http://34r6hq26q2h4jkzj.onion.cab",
      "https://34r6hq26q2h4jkzj.tor2web.org",
      "http://34r6hq26q2h4jkzj.onion/",
      "https://crbug.com/401439",
      "http://www.sk.ee/juur/crl/0",
      "http://pesquisa.sapo.pt/?q=",
      "https://www.thawte.com/repository0W",
      "http://torproject.org",
      "http://34r6hq26q2h4jkzj.onion/FILE0"],
"mail_addresses": [
      "personal-premium@thawte.com",
      "server-certs@thawte.com",
      "personal-basic@thawte.com",
      "info@globaltrust.info",
      "(dan.blanchard@gmail.com)",
      "info@a-cert.at",
      "polly.disused@fake.chromium.org",
      "ancert@ancert.com",
      "polly.deletable@fake.chromium.org",
      "info@izenpe.com",
      "gold-certs@saunalahti.fi",
      "jmctester@fake.chromium.org"],
}
```

Code Snippet 5.1: Example of report. It is possible to see execution parameters and intelligence information extracted divided by type

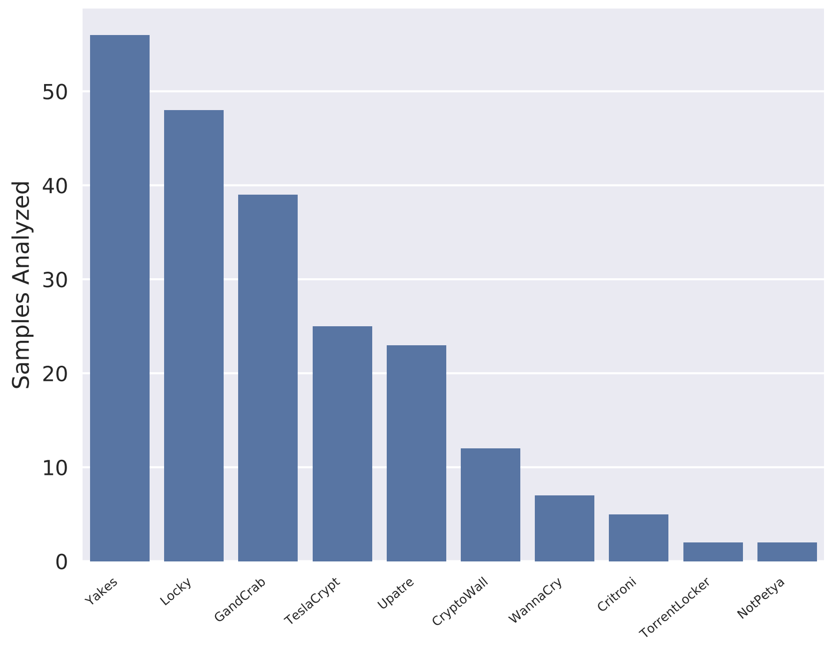


Figure 5.2: Different families analyzed, with relative number of samples inside the working dataset

the address in the correct way, mistakenly recognizing only one character in most of the cases (four out of five for Tesseract, and both cases for Cloud-Vision), or missing characters in the remaining cases. In two cases where no tool recognized the address, we noticed that they missed it only because they misdetected spaces inside the address so we could not recognize the characteristic pattern of BitCoin addresses. This happens because the random sequence of characters that compose the BitCoin address is not comparable with words of a natural language, making OCR optimization on dictionaries useless. We also verified that our preprocessing steps are necessary for Tesseract to perform better: if we apply a standard binarization function, for example the one that Tesseract uses by default, we are not guaranteed to transform the image in the proper way. Tesseract, in fact, needs black content on white background. We tested the standard Tesseract binarization function for the 10 cases mentioned above, but due to the different colors used in the images we obtained white content on black background in 7 cases out of 10. In the remaining three cases the conversion was not good enough to allow correct character recognition, and thus it completely missed the address contained in the image. From this series of tests we concluded that in this case Tesseract, helped by a custom preprocessing pipeline, performs better than any agnostic commercial tool, like CloudVision.

5.4.4 Performance Evaluation

We measured the performance of our framework distinguishing the case when the sample shows malicious activity from the one where the sample does nothing. For both of the cases we exclude the time spent for dynamic analysis, that is fixed at ten minutes plus a few seconds for auxiliary files generation and memory dump. Given this assumption, we focus on the time for framework execution only, from the moment the dynamic analysis is terminated, to the time RansomScan produces an analysis report. For working samples RansomScan required an average execution time of about 21 minutes, with a minimum of 3 minutes and a maximum of 57 minutes. This difference depends on the number of artifacts analyzed, and specifically images: this is the most time-consuming operation, due to image processing and CloudVision requests limit. In Figure 5.3 we report the results of the performance test in terms of analysis duration for working samples. Regarding disk occupation, the space requested is the one used for dynamic analysis results, composed mainly of memory dump and screenshots. With our setting we have an average of 9.7 GB for each folder containing the full dynamic analysis result. We decided to keep these files in order to allow possible reanalysis, but in low space situations we could remove them after the framework execution, making the occupation only temporary. The framework needs little to no space, since most of the process is executed in memory and it saves only the final report, with a maximum weight of 150 KB. We can see that our approach is not impacting on disk, and does not include resource-consuming operations. The execution time is sample-dependant, but overall it is acceptable.

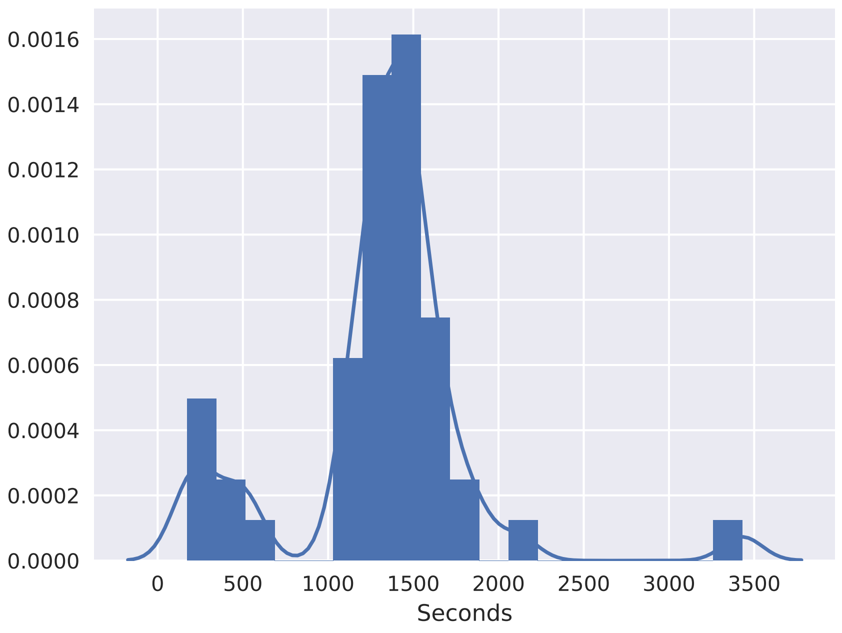


Figure 5.3: Probability distribution for execution times

Chapter 6

Limitations

During the course of this study, we found that our approach falls short in his aim in specific cases determined by external causes that we could not manage. The first limitation we have to acknowledge is that we could not analyze every ransomware sample obtained, due to the ensemble of ransomware evasion techniques: even if we adopted the already mentioned countermeasures in order to overcome at least in part this problem, we still faced samples that were able to detect the virtual environment and stop the execution. This could be solved using a bare-metal virtualization system, like the frameworks BareBox and BareCloud proposed in [37, 38]. These frameworks consider using bare-metal systems based on fast, rebootless, system restore technique, and without introducing in-guest monitoring components. On top of this kind of approach, it was possible to develop advanced techniques aimed to harden the system and make it stealthier, in order to avoid detection by malware [39]. In our case we could not apply those approaches since those would request a different kind of hardware resources, and would reduce the flexibility given by a host virtualization hypervisor in controlling the virtual machines. This was not the only factor causing a reduction in the number of analyzed families: many samples were harmless because deactivated after a given time, voluntarily or for external causes, so we could observe only a small part of the original dataset. This deprived us of their information and made it difficult for us to make considerations about the comparison between results obtained from old samples and the ones coming from fresh samples. Finally, given the abrupt changes in ransomware ecosystem, the last limitation is given by the impossibility of training the OCR software in order to increment the performances: we noticed that, due to the combination of a broad range in images quality, differences in ransomware graphical elements and differences in calibration of the OCR software, it is very difficult to correctly identify sequences of characters with no linguistic meaning, such as a cryptocurrency wallet address. Since we rely on external OCR tools we cannot train them specifically to recognize the information we need. Moreover, since we have too many ransomware families with different graphic interfaces, training a specific OCR just to recognize wallet addresses would result in a loss of generalization: there is no way in fact to predict how new ransomware would display their messages and thus we would incur into an overall narrowing of the recognized interfaces. We could probably achieve an increasing in accuracy on known families but we would fail to read information from all the new ones. This would totally be in contradiction with the aim of this work that is to develop a tool as comprehensive as possible. Even if we would try and overcome this obstacle anyway, the training of the new tool would still be difficult and would require a great number of samples in order to maintain a good grade of generalization.

Chapter 7

Future Work

While it is clear that the main priority emerging from this study is to overcome the limitations presented in section 6, we can think about long term development of this topic and identify possible suggestions in order to contribute to this work.

Analyzing the framework in a broader sense, we can notice that it relies on several external components in order to accomplish its tasks. It is natural that with an improvement in the performances of these tools, whether they are fine-tuned, or ad-hoc ones are created, comes also an improvement in the performances of our system.

Another important underlying factor is the state of the art in malware anti-evasion techniques: most recent samples have multiple complex criteria to decide if they are under examination or not, and creating a non detectable simulated environment is becoming more and more difficult. From an improvement of the anti-evasion techniques, we would benefit in terms of number of available samples; an important parameter that helps having insights on new families, and thus keeping the level of generality of the framework high.

We can highlight also a more specific suggestion related to our work: given the fact that ransomware is constantly evolving, our framework could become obsolete. It is then important to keep it updated to the last ransomware trends: implement recognition of different formats of cryptocurrency wallets, modify image processing heuristics in order to obtain a better result following new trends in messages showed on screen by ransomware.

Chapter 8

Conclusions

We proposed RansomScan, a framework that combines different techinques for data extraction in order to offer an automated analysis of ransomware samples. The tool exploits all traces left from the malware on the victim's machine, with the purpose of extracting endpoints useful for ransomware ecosystem understanding. Every different type of artifact is analyzed in a dedicated way in order to make the most of it. Differently from parallel studies in the same field, our system is designed to combine a high level of precision and performance with a most general and comprehensive approach, not limited to specific ransomware families or traits. In this way we try to cover all possible cases, making the tool robust to ransomware evolution and mutability. Furthermore, given the modularity of our framework, it is suitable for different use cases: it can be used as a black box if we are interested in a full automation of the analysis process, but can be separated in its components for specific usage. This allows reanalysis of samples without repeating the dynamic part, with great time saving, but also makes the tool compatible with all kind of artifacts coming from different means of analysis.

We evaluated RansomScan performance on a dataset of 532 samples coming from 32 different ransomware families and of different age. We managed to extract 117 addresses, and more than 500 potentially malicious URLs. The discrepancy between the number of the samples and the data extracted is given by the fact that many samples were dead or aware of the simulated environment. However, results show that RansomScan has no false positives when it comes to BitCoin addresses, and almost no false negative, also due to the intrinsic nature of said addresses. It is more difficult to evaluate URLs results, since even after controls and whitelisting, it is hard to automatically distinguish between malicious URLs and harmless ones. Based on a subset of 20 addresses, we manually verified that 10 of them are actually harmless links, giving us a 0.5 probability of finding a false positive result. This changes if we consider only Onion URLs. In that case they are all malicious, because the only way they are generated is from malware. We retrieved 181 Onion URLs in the ransomware samples analysed. We also focused on OCR software evaluation: on a small set of samples analyzed manually, we could observe that relevant information in the image is detected only in 5 cases on 10 for Tesseract-OCR. In that case Tesseract performs better than Google CloudVision and Kraken-OCR, and in any case the information is extracted with errors. This shows that in che OCR field there is still room for improvement.

We designed RansomScan with a particular care for performances: we filled the process with controls that help reducing the amount of useless operations, reaching durations that are around 21 minutes per single analysis. The framework is also exploiting analysis parallelism, in order to reduce times when it comes to submissions composed of multiple samples.

This work has been conducted with the hope of contributing to the study of ransomware ecosystem, perhaps leading to a restraint in ransomware growth and damage. In any case, we hope we are able with our contribution to deepen knowledge regarding the still little known background behind this phenomenon.

Bibliography

[1] Jim Bates. Trojan horse: Aids information introductory diskette version 2.0. *Virus Bulletin*, pages 3–6, 1990.

[2] Proofpoint.com. Proofpoint quarterly threat report 2017, aug 2017.

[3] F-Security. Cyber security report 2017, 2017.

[4] Fedor Sinitsyn Santiago Pontiroli Anton Ivanov, David Emm. Kaspersky security bulletin 2016. the ransomware revolution, dec 2016.

[5] Osterman Research. Understanding the depth of the global ransomware problem, aug 2016.

[6] Tor project. https://www.torproject.org/.

[7] Bitcoin. https://bitcoin.org/en/.

[8] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.

[9] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 336–347. ACM, 2016.

[10] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. Paybreak: defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 599–611. ACM, 2017.

[11] Matilda Rhode, Pete Burnap, and Kevin Jones. Early-stage malware prediction using recurrent neural networks. *Computers & Security*, 77:578–594, 2018.

[12] Hon Lau Kevin Savage, Peter Coogan. The evolution of ransomware, 2015.

[13] Kaspersky Lab. Kaspersky security bullettin: Story of the year 2017, 2017.

[14] Franklin S Cooper, Jane H Gaitenby, and Patrick W Nye. *Evolution of Reading Machines for the Blind: Haskins Laboratories' Research as a Case History.* Haskins Laboratories, 1983.

[15] Faisal Mohammad, Jyoti Anarase, Milan Shingote, and Pratik Ghanwat. Optical character recognition implementation using pattern matching. *International Journal of Computer Science and Information Technologies*, 5(2):2088–2090, 2014.

[16] Nallasamy Mani and Bala Srinivasan. Application of artificial neural network model for optical character recognition. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 3, pages 2517–2520. IEEE, 1997.

[17] Google. Google cloudvision. https://cloud.google.com/vision/.

[18] Tesseract ocr. https://github.com/tesseract-ocr.

[19] Kraken ocr. http://kraken.re/.

[20] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. 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.

[21] Amin Kharraz, Sajjad Arshad, Collin Mulliner, William K Robertson, and Engin Kirda. Unveil: A large-scale, automated approach to detecting ransomware. In *USENIX Security Symposium*, pages 757–772, 2016.

[22] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. Cryptolock (and drop it): stopping ransomware attacks on user data.

In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 303–312. IEEE, 2016.

[23] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[24] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security*, pages 457–468. Springer, 2014.

[25] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 618–631. IEEE, 2018.

[26] Cuckoo sanbox, automated malware analysis. https://cuckoosandbox.org/.

[27] Oracle vm virtualbox. https://www.virtualbox.org/.

[28] Python imaging library (pil). http://www.pythonware.com/products/pil/.

[29] Sqlalchemy: the database toolkit for python. https://www.sqlalchemy.org/.

[30] Psicopg: Postgresql + python. http://initd.org/psycopg/.

[31] Scapy. https://scapy.net/.

[32] Requests: Http for humans. http://docs.python-requests.org/en/master/.

[33] Microsoft free virtual machines for ie8-msedge development. https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/.

[34] Python-magic: a python wrapper for libmagic. https://github.com/ahupp/python-magic.

[35] Linux manual page - strings. https://linux.die.net/man/1/strings.

[36] Virustotal - free online virus, malware, and url scanner. https://www.virustotal.com/it/.

[37] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barebox: efficient malware analysis on bare-metal. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 403–412. ACM, 2011.

[38] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barecloud: Bare-metal analysis-based evasive malware detection. In *USENIX Security Symposium*, pages 287–301, 2014.

[39] Mario Polino, Andrea Continella, Sebastiano Mariani, Stefano D'Alessio, Lorenzo Fontana, Fabio Gritti, and Stefano Zanero. Measuring and defeating anti-instrumentation-equipped malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 73–96. Springer, 2017.