Dual-Use Research In Ransomware Attacks: 
A Discussion on Ransomware Defence Intelligence

Ziya Alper Genç\textsuperscript{1} \textsuperscript{a} and Gabriele Lenzini\textsuperscript{1} \textsuperscript{b}

\textsuperscript{1}Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
\{ziya.genc, gabriele.lenzini\}@uni.lu

Keywords:
Ransomware, Double Use Research in Cryptography, Threat Intelligence and Counter-Intelligence.

Abstract:
Previous research has shown that developers rely on public platforms and repositories to produce functional but insecure code. We looked into the matter for ransomware, enquiring whether also ransomware engineers re-use the work of others and produce insecure code. By methodically reverse-engineering 128 malware executables, we have found that, out of 21 ransomware samples, 9 contain copy-paste code from public resources. Thanks to this finding, we managed to retrieve the decryption keys with which to nullify the ransomware attacks. From this fact, we recall critical cases of code disclosure in the recent history of ransomware and, arguing that ransomware are components in cyber-weapons, reflect on the dual-use nature of this research. We further discuss benefits and limits of using cyber-intelligence and counter-intelligence strategies that could be used against this threat.

1 INTRODUCTION

In anti-ransomware research, ransomware samples are routinely analyzed. The goal is to understand how they generate or retrieve the encryption keys; how they search, sort and prioritize which files to target first; and which files they encrypt first and by using which encryption algorithm. In this quite methodical work, it is routine to reverse engineer ransomware samples and analyze their source codes. While performing this task, we found that some piece of code was not original but copy-and-pasted from well-known public repositories or developers communities. From this discovery, with some additional work, we managed to build a decryptor for those ransomware samples.

Although our discovery is not surprising—researchers have already commented on how codes from public repositories is re-used and how this impacts security (e.g., see (Fischer et al., 2017))—realizing that also ransomware’s security depends on public code has captured our attention. We started wondering whether there were other cases of copy-and-pasted code in ransomware. And we started reflecting on which consequences such re-use of code may bring into the fight against ransomware attacks. This article reports on our insights on the subject.

Although motivated by some experimental findings, our contribution is purely argumentative. But, by developing our argument rigorously, we hope to contribute to a scientific discussion on “the matter”. And being “the matter” related to dual-use of concern in ransomware research, we intend to embark on other questions as well: What famous precedents exist in the recent history of ransomware that could enlighten us on the pros and cons of dual-use research? Should ransomware be considered components of a cyber-weapon? And, as such, are there reasons to classify ransomware as having military use? Thus, would it be reasonable to resort to intelligence and counter-intelligence strategies, such as those suggesting to contain information spreading in case of an attack or to control public information, to mitigate the threat? We restrict our argument to cryptographic ransomware, those which rely on cryptography. Other kind of ransomware, e.g., those which aim to distress victims to pay up but,
like the scammers, only pretend to use encryption but do not, are excluded from the discussion.

2 PRELIMINARIES

Is copy-and-paste from public repositories a practice in ransomware engineering? To investigate the question we have first to collect and obtain the code of real-world ransomware samples and reverse engineer it. The most accurate way to accomplish this latter is to decompile the malicious binaries. The task becomes quite practical if the malware is implemented using the .NET framework. Looking into malicious .NET assemblies downloaded from “Hybrid Analysis”, an automated malware analysis platform (Hybrid Analysis, 2019). Hybrid Analysis utilize sandboxing technique to determine if an executable exhibits malicious behaviour or poses no specific threat. From it, we collected ransomware samples by searching on report database with the following settings: (i) Exact Filetype Description as Mono/.Net assembly, for MS Windows; (ii) Verdict field as Malicious; and (iii) Hashtag field as #ransomware.

On a initial set of 128 executable, we applied dnSpy (dnSpy, 2019), a tool to obtain source codes. 39 samples, obfuscated, precluded any analysis. Of the remaining 89, we manually pe-rused the source code, searching for key generation and encryption routines. 68 samples turned out to be non-cryptographic ransomware, with no such routines in their program body. The remaining 21 cryptographic ransomware samples were our final data set.

Using the found crypto-related code lines (e.g., key derivation, encryption) as keywords, we searched for those lines in public developer platforms. When analyzing the hits, we compared the semantics of code snippets, naming of constants and variables, function signatures, strings, and error messages. From this searching and matching we discovered that some code was a verbatim copy-paste. Other code resulted, at least apparently, a plagiarism of some public available code.

Were we witnessing code-reuse (i.e., dual-use) in ransomware? Before claiming code-reuse, we had to verify whether the code had been published before the first appearance of it in the malware. There should also be a reasonable time frame between the two events. The date of the first appearance of a ransomware, checked by using VirusTotal (VirusTotal Threat Intelligence, 2019), has been compared with the date on which the knowledge was first shared on online. A double-checked on the integrity of the pieces of information available on the executable was also performed. According to our findings, at least 9 out of 21 ransomware samples resulted to contain snippets bearing a marked resemblance to codes at online resources, this leading us to conclude that they are in fact a copy-paste.

In the following, we can comment on an excerpt from the ransomware samples (see Table 1) that we have found being a copy-paste from (i) a public repository of fully functional ransomware prototypes; (ii) tutorials and posts at developer communities. We also elaborate, where possible, about where the original code comes from, and about its cryptographic qualities.

Ransomware from repositories of fully functional prototypes. Tiggre, see Table 1, is a sample of cryptographic ransomware that uses a key generation function that is copy of a piece of public code known as HiddenTear (Șen, 2015b) (see Fig. 2 and Fig. 1). From it, Tiggre inherits a weakness: the password is generated using the outputs of a cryptographically weak algorithm. In fact, the same author of HiddenTear had developed a decryptor by using this weakness (Şen, 2015a). We tell the full story later, but what counts for now is that the open-source ransomware HiddenTear is a very famous ransomware code, which was posted publicly in 2015 allegedly for educational purpose. Since then, cyber-criminals have been using it as a source of inspiration for their ransomware variants (van der Wiel, 2016). This was also the case for Tiggre.

The original HiddenTear works as follows: it generates a password by calling CreatePassword which is shown in Fig. 1. The password, from which the encryption keys are derived, is sent to Command and Control (C&C) server. Next, before notifying the user, the ransomware attempts to encrypt all the files in test folder under the user’s Desktop directory.

Ransomware authors that copy from HiddenTear had to implement their own back-ends before having a working ransomware, but HiddenTear remains their point of reference. We have found that basic functionalities such as password generation and encryption blocks have been replicated from HiddenTear: for each file, the encryption key is derived from the same master secret, the password; this latter is generated using System.Random, a class
that provides (cryptographically weak) pseudo random numbers.

From a cryptographic point of view, the outputs of System.Random is reproducible when using the same seed and its secrets are vulnerable to a forensics analysis. But other variants of HiddenTear eliminate this weakness: the weak key generation method is not seen in those samples.

## Ransomware from community platform.

Confidentiality of data is a highly demanded and legitimate need in the digital world. While cryptographic techniques can be used to protect the secrecy of data, developing a security application is an error-prone process. Therefore, developers who recently entered in the field of cybersecurity might need to use the help of online tutorials. For example, Fig. 3 shows a post on CodeProject which the key is derived. The variant uses an embedded password to generate a password from the password (see Table 1). The post, available at (den Bosch, 2005), explains how to encrypt files with a user-supplied password in VB.NET programming language. Many portions of the code is reused by the ransomware samples, bar the part which takes input (i.e., the password) from the user. Alternatively, the said code portion is not used/referenced by the program.

In another case, we observed that an online tutorial published in 2005 inspired two ransomware samples: Perseus and CloudSword (see Table 1). The post, available at (den Bosch, 2005), explains how to encrypt files with a user-supplied password in VB.NET programming language. Many portions of the code is reused by the ransomware samples, bar the part which takes input (i.e., the password) from the user. Alternatively, the said code portion is not used/referenced by the program.

**Discussion.** Our findings, we can conclude that certain ransomware engineers do copy-and-paste code from public sites. Surely, this conclusion cannot be representative of how all ransomware variants are coded. We do not even know whether who took advantage the public resources are professionals or amateurs, and it may be inherently hard to investigate for an answer
on this matter due to the difficulty to reach out ransomware developers. However, we speculate, ransomware engineers are likely not in a different position than security developers. In (Acar et al., 2017), it is reported that in a population of three hundreds developers among which also professionals, only a quarter relied on the official documentation, while the rest consulted “the Internet”, inevitably relaying in their code errors na"ıvities “out there”, cause them to introduce security vulnerabilities in their code.

This seems to remain valid in our case: the security of some ransomware depends, at least in part, on the security reliability of the unofficial sources. A question remains open. Has the code-use helped ransomware criminals? The question is intertwined with the practice of dual-use of research in the field and, for this reason, we looked into the recent history of ransomware attacks in search for episodes of code re-use.

3 DUAL-USE & RANSOMWARE

Article 2 of Council Regulation (EC) No 428/2009 defines ‘dual-use items’ as items which can be used for both civil and military purposes. The article includes “Computers” and “Telecommunications and Information security” as categories to be screened for potential dual-use.

When it comes to cryptography, dual-use is a serious matter. In response to the US Munitions List, Category XIII, Materials and Miscellaneous Articles, which mentions “cryptographic devices, software and components”, in a T-shirt shown at the DEFCON conference it was reported provocatively a piece of (encryption) code with the comment “this [code] can also be a munition” (Herr and Rosenzweig, 2015).

Within the cryptography community there is awareness that dual-use comes with a moral burden. Rogaway wrote that “cryptography is an inherently political tool, and it confers on the field an intrinsically moral dimension” (Rogaway, 2016). Rogaway’s argument is scoped in the contention between privacy on one side and mass surveillance on the other, but the message on that DEFCON T-shirt extends, even reverses, the matter. It raises the stake by pointing out that cryptographic code can be misused as a weapon. This is still the vision in certain countries, for instance the US, where non-military cryptography exports are if not forbidden at least controlled.

Being the subject of this paper ‘ransomware’, the matter must be contextualized: what about dual-use for cryptographic ransomware? And are ransomware and their cryptographic components weapons? To answer this question we look into cases of dual-use in ransomware. The most controversial is that of HiddenTear and its clones.

HiddenTear and its Clones. In 2015, a Turkish programmer Utku Şen published the first fully-fledged, open-source ransomware HiddenTear. This is the sample we commented in the previous section and whose code to generate a password is shown in Fig. 1.

From the early days, the release of HiddenTear prototype received criticisms from the security community (Kovacs, 2016). The main concern of the researchers is that even novice programmers can also make use of the published ransomware code while developing new variants. Time showed that they were right. A McAfee researcher stated that “in June (2017) almost 30% of the ‘new’ ransomware species we discovered was based on the...
private void EncryptFile(string inputFile, string outputFile)
{
    try
    {
        string password = @"myKey123";
        UnicodeEncoding UE = new UnicodeEncoding();
        byte[] key = UE.GetBytes(password);
        string cryptFile = outputFile;
        FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);
        RijndaelManaged RMCrypto = new RijndaelManaged();
        CryptoStream cs = new CryptoStream(fsCrypt, RMCrypto.CreateEncryptor(key, key), CryptoStreamMode.Write);
        FileStream fsIn = new FileStream(inputFile, FileMode.Open);
        int data;
        while ((data = fsIn.ReadByte()) != -1)
            cs.WriteByte((byte)data);
        fsIn.Close();
        cs.Close();
        fsCrypt.Close();
    }
    catch
    {
        MessageBox.Show("Encryption failed!", "Error");
    }
}

private static string GetEncKey()
{
    try
    {
        using (WebClient webClient = new WebClient())
    }
    catch
    {
        return "myke123!";
    }
}

private static void EncryptFile(string inputFile, string outputFile, string password)
{
    try
    {
        byte[] bytes = new UnicodeEncoding().GetBytes(password);
        FileStream fileStream1 = new FileStream(outputFile, FileMode.Create);
        RijndaelManaged rijndaelManaged = new RijndaelManaged();
        CryptoStream cs = new CryptoStream(fileStream1, rijndaelManaged.CreateEncryptor(key, key), CryptoStreamMode.Write);
        FileStream fileStream2 = new FileStream(inputFile, FileMode.Open);
        int data;
        while ((data = fileStream2.ReadByte()) != -1)
            cs.WriteByte((byte)data);
        fileStream2.Close();
        cs.Close();
        fileStream1.Close();
    }
    catch
    {
        Console.WriteLine("Error: Encryption failed!");
    }
}

Figure 3: A simple function to encrypt files with a password, published at CodeProject. Contrary to the common practices, e.g., PBKDF2 (Kaliski, 2000), encryption key is derived directly using UTF-16 character encoding. In addition, instead of generating a unique value, encryption key is used as IV.

Figure 4: File encryption function of the Crypren sample. If C&C server is not reachable, which is shut down at the time of this writing, the embedded password is used to derive keys. The resemblances between hard-coded passwords, key derivation methods and error messages are remarkable.

HiddenTear code" (bee, ).

Three months after the first release, Şen claimed that he wished (i) to provide an example of ransomware for beginners (ii) to build a honeypot for script kiddies (Şen, 2015a). It was partly true that the first variants of HiddenTear contained the same critical bugs that enabled the recovery of files (van der Wiel, 2016). However, one real thing in the malware history is evolution. The bugs in the original HiddenTear was fixed, and HiddenTear variant replaced the cryptographically insecure key generation method with a new one (Trend Micro Blog, 2017) which evades the state-of-the-art key-oriented anti-ransomware defenses. Later, Şen admitted that his experiment was a total failure.

Another criticism to publishing the full source codes of a ransomware regards the principle of responsible disclosure. Prior to sharing the sources, Şen did not inform the anti-virus vendors. It should be noted that, when HiddenTear was released, on August 2015, only a few anti-ransomware systems existed: signature-based detection was the main technique to stop ransomware, just as the other malware types. Since HiddenTear and its variants were previously unseen, they were not recognized by AVs and therefore could run undetected for a while. The only precaution Şen took was putting a warning message in HiddenTear source code, which cyber-criminals could easily ignore.
Further Public Prototypes. Şen is not the only person that published a full ransomware prototype. There are several ransomware projects in different programming languages, publicly available on the Internet. For instance, Are- scrypt is another open source ransomware implemented in C# (Fox, 2017). GonnaCry is a Linux ransomware, implemented in both C and Python (Marinho, 2017). Aiming at web servers, a ransomware script written in PHP is also available at (Sincek, 2019). There is even an “academic” ransomware prototype implemented in Go language (de Souza Nunes, 2016). All these projects are publicly available at GitHub, a well-known platform among software developers. Moreover, although Şen abandoned the HiddenTear project, there are still several clones of the original repository and even some improved versions of HiddenTear on GitHub website, for example (Rosa, 2017).

Zaitsev followed a different strategy when publishing CryptoTrooper (Zaitsev, 2016). He shared the core part of the prototype as a closed source binary. The encryption algorithm, whose code was not shared, contained a cryptographic flaw which enabled the recovery. Being closed source, the flaw in the encryption module of CryptoTrooper could not be fixed by the script-kiddies. Still, the community was divided: some found the idea useful, others did not (Cimpanu, 2016). In the end, Zaitsev removed the project from GitHub but, as in the case of HiddenTear, CryptoTrooper was forked by other developers. It is still accessible via various repositories.

All the developers of the publicly available ransomware prototypes states that their main motivation was educational. However, a well documented ransomware code would also help to-be-cyber-criminals to enter the ransomware business. Since ransomware prototypes remain available on the Internet, the ethical question here is whether security researchers need to publish and share full ransomware codes without feeling accountable of the consequences, a recognized ethical issue.

4 RANSOMWARE INTELLIGENCE

Herr and Rosenzweig suggest that a piece of code is cyber-weapon when it combines “propagation, exploitation, and payload [i.e., damaging] capabilities” (Herr and Rosenzweig, 2015). Each components, despite innocuous in separa-

Ransomware Positive Intelligence For ransomware threat, positive intelligence could consist in gathering information about modalities of working. It should be about how the ransomware propagates, exploits vulnerabilities, and executes it payload. In the Open Source Intelligence (OSINT), several initiatives exist aiming to collect and analyse information gathered from public or open sources. An example is the NoMoreRansom project. It aims to inform the

1https://www.nomoreransom.org/en/
public and to collect incidents reports, including
to gather the information from public platforms
that can be potentially utilized by ransomware
authors. Other platforms, although not specifi-
cally dedicated to ransomware, such as the Mal-
ware Information Sharing Platform (MISP) 2—a free and open source software helping infor-
mation sharing of threat intelligence, including
cyber-security indicators—can offer tools that en-
able intelligence analysis. Such platforms can be
employed to control the information flow during
an attack, spreading alerts following a Warning
and Coordination action, and to help potential
victims “raise their shields” as soon as possible.

Ransomware Counter-Intelligence According to (Coleman, 2009), Counter Cyber Intelli-
gence (CCI) is the ensemble of “all efforts made
by one intelligence organization to prevent adver-
saries, enemy intelligence organizations or crim-
inal organizations from gathering and collect-
sing sensitive digital information or intelligence
about them via computers, networks and associ-
ated equipment”. It can be implemented using
strategies that, according to Panda Security,
a cyber-security company, either consists of
“leaving doors open” (i.e., left access points un-
protected on purpose), “inject fake information”
(i.e., fake confidential information), and “keep-
ing them busy while stealing” (i.e., watching and
obtaining information about the attacker).

Looking into the internet and searching for
“counter-intelligence for ransomware”, we have
found that the majority of the initiatives to pro-
tect from ransomware attacks focuses on rais-
ing awareness. For instance, the US National
Counter-intelligence and Security Center (NCSC)
has launched in January 2019 a campaign “Know
the Risk, Raise Your Shield”. The Cybersecu-
ritiy and Infrastructure Security Agency (CISA)
addresses ransomware specifically, but it is all
about knowing the threat and apply general secu-
ritiy best practices such as backing-up data. We
have found, within the scope of cryptographic ransomware and limitedly to this on-going work, nothing about “leaving the door open”, “inject
fake information”, or “keep them busy”.

The second measure (i.e., “inject fake infor-
mation”) may not be fully applicable at least if
that means to avoid to spread knowledge about
how to build the ransomware weapon: the in-
struments of cryptography are nowadays already
known and public. However, at the light of what
we have discussed in the previous sections, it may
be a strategy to post the code of variants whose
decryptors already exist.

For what concerns the “keep them busy”
paradigm, as discussed in (Genç et al., 2019),
it may be possible the use decoy files to deflect
a ransomware attack against irrelevant (for the
victim) files, so gaining that amount of time re-
quired to stop the attack’s development. Using
devo files could be paired with strategies that
downgrade the efficiency of encryption for appli-
cations that are not trustworthy or whitelisted.
We have not investigated in this direction, but
this option seems preferable to that of running
untrusted application in sandbox. This can be
less effective, since certain ransomware sample
recognize the presence of a virtual environment
and remain dormant. A few articles suggest the
use of Artificial Intelligence (e.g., (Huang et al.,
2018)), but we did not look into this direction.

5 CONCLUSION

Ransomware are emerging as cyber-weapons.
They have been used in attacks that resemble ac-
tions of cyber-war, and are far more dangerous
and disruptive than traditional malware. Conse-
quently, the research community should reflect on
coordinated actions to address the threat under
an appropriate code of ethical conduct.

Having discovered that a few ransomware con-
tain a copy-paste from cryptographic code avail-
able in public sources, we debated the matter of
dual-use in cryptographic research and recalled
(in)famous antecedents in the recent ransomware
history. Since we managed to build decryptors for
those ransomware, the dual use turned out to be
a double-edge for the criminals, but generally it
is not. After having build a case for ransomware
as cyber-weapon, we briefly reviewed intelligence
and counter-intelligent strategies that could be
used in the fight against ransomware.

We did not backed our speculations with field
studies or interviews. Ours is an educated argu-
mentation, but its purpose is to invite the anti-
ransomware community to be more proactive in
the cyberwar against ransomware. Even the ex-
cellent NoMoreRansom project, which offers
decryptors when they are available (as did in June
2019, with the latest version of Gandcrab3), at
the end of the day praises for keeping back-up,
within a “Better Safe Than Sorry” advice.

2https://www.misp-project.org/
3https://www.malwarebytes.com/gandcrab/
REFERENCES


