Inside the Matrix
CTI Frameworks as Partial Abstractions of Complex Threats

Oosthoek, Kris; Doerr, Christian

DOI
10.1109/BigData52589.2021.9671425

Publication date
2021

Document Version
Final published version

Published in
2021 IEEE International Conference on Big Data (Big Data)

Inside the Matrix: CTI Frameworks as Partial Abstractions of Complex Threats

Kris Oosthoek
Cyber Threat Intelligence Lab
Delft University of Technology
Delft, The Netherlands
k.oosthoek@tudelft.nl

Christian Doerr
Cyber Threat Intelligence Lab
Hasso Plattner Institute
Potsdam, Germany
christian.doerr@hpi.de

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671425

***Abstract*—The Cyber Threat Intelligence (CTI) field has evolved rapidly and most of its reporting is now fairly standardized. Where the Cyber Kill Chain was its sole reference framework 5 years ago, today ATT&CK is the de facto standard for reporting adversary tactics, techniques and procedures (TTPs). CTI frameworks are effectively abstraction layers of malicious behavior and thus effective CTI dissemination hinges on their ability to accurately represent this behavior. We argue that this is an area with significant opportunity for improvement. The aforementioned models are attacker- and intrusion-centric, while much of the CTI reporting currently is artifact- and malware-centric. In other words, most analysis is performed using artifacts of adversary *tools*, while in-the-wild evidence of adversary *techniques and procedures* is limited or lacking. Applying an intrusion model to artifact-based analysis leads to information loss, affecting and potentially misleading CTI-based decision-making. Intelligence analysis naturally builds on imperfect information, but CTI frameworks should be oriented more towards this key premise. In this conceptual work we compare the intrusion-centric ATT&CK with Malware Behavior Catalog (MBC), which is malware-centric. We compare how their application affects reporting of analysis outcomes. For this we reverse a piece of APT malware, replicating how many CTI reports are produced. We find that compared to ATT&CK, the abstraction offered by MBC enhances the information density of our reporting. While currently in most industry malware reports ATT&CK is applied, our analysis shows that on these occasions using MBC, potentially in tandem with ATT&CK, improves reporting. With the daily amount of new malware samples only increasing, accurate behavior labeling is key to the success of CTI sharing and dissemination.**

***Index Terms*—malware analysis, cyber threat intelligence, Malware Behavior Catalog (MBC), ATT&CK, reverse engineering**

I. Introduction

The Cyber Threat Intelligence (CTI) field aims to build a common understanding of the tools, techniques and procedures (TTPs) used by adversaries to compromise computer systems and networks. A key factor in the success of building a common understanding of threats is having a common vocabulary. In recognizing the importance of this, the field has produced many of these *systems of systems* to understand adversary behavior. Well known examples of attacker-centric frameworks are the Cyber Kill Chain, ATT&CK, the Diamond Model, VERIS, CAPEC and TARA. The unified lexicon they provide enables quick filtering, curation and consequently more effective sharing and dissemination of CTI.

Much of the contributions of these frameworks to CTI corresponds to the *separation of concerns* provided by *abstraction layers* in software development. An example of a well-known conceptual model is the OSI model, which consists of seven abstraction layers. The layers provide a *separation of concerns*, encapsulating and addressing requirements into different representation levels of data flow. Through this, it reduces complexity in the engineering of communication solutions, as only the specific layer and its protocol need to be taken into account rather than the whole ecosystem. CTI frameworks are similar. They encapsulate intelligence about TTPs and intrusion sets, providing a representation of behavior relevant to the target audience and applying *information hiding* to details deemed unnecessary to CTI decision-making.

These frameworks source their input from various cyber security disciplines. A trail of forensic evidence of intrusion activity is preferred, as this provides rich details on tools, techniques and procedures respectively. A rigorous incident response effort will extract relevant malicious binaries, host logging but also network traces of attacker movement. Most of the time however such ground truth is not available as input to CTI analysis. In such cases, analysis is based on single artifacts like IP addresses, DNS names, malware samples, or a combination of these, effectively serving as *derivatives* of attacker activity.

Regardless of which conceptual model is used, the *integrity* of the underlying information must be maintained. If the framework of choice does not accurately cover subsets of this real-world behavior, the resulting abstraction - the CTI report - is consequently deficient. This potentially negatively affects all decision-making based on this reporting. Therefore we argue that in cases where evidence of actual attacker or intrusion activity is absent and one only has a malware binary or select Indicators of Compromise (IoCs) to work with, applying an attacker- and intrusion-centric framework like ATT&CK is sub-optimal. A framework capturing intrusion behavior is unfit to describe malware behavior, which is a small subset of a full intrusion set. Behavior of this smaller subset (the malware) is ideally described using a dedicated abstraction layer.

An example of this are malicious techniques like heap spraying, stack pivoting, which are not covered by taxonomies such as ATT&CK. Other examples are anti-behavioral and anti-static analysis techniques which are not necessarily malicious, as many developers of benign commercial software use these to protect their intellectual property. However when observed in context of malicious software these become suspicious and potentially malicious. Ideally all of the aforementioned behavior is captured by a framework used to disseminate analysis results, or it is likely omitted from the eventual reporting. This is especially important because for a the majority of the readers, the ATT&CK technique lists in CTI reports have become *single sources of truths* to consider whether follow-up for mitigation and detection is required. Weaknesses in the framework, as well its application, then make it a *single point of failure*.

In this paper we consider the utility of ATT&CK and Malware Behavior Catalog (MBC) [1] in standardizing CTI reporting based on malware analysis. MBC aims to serve as a catalog of malware objectives and behaviors to support standardized malware analysis reporting. Where ATT&CK after its launch in 2016 effectively became the industry CTI reporting standard, MBC has been around since 2019 and is lesser-known. Google Scholar queries show that MBC has received no academic research attention until now, with its industry adoption also still relatively limited. We compare both frameworks based on analysis of a single piece of APT malware, *BBSwift*. We performed a dedicated analysis of *BBSwift* for this paper to accurately reflect how many CTI reports are realized. The 2016 intrusion in which *BBSwift* was involved is colloquially referred to as the Bangladesh Bank cyber heist. Though the event has received considerable popular media attention, from a CTI perspective the malware itself has been given limited attention and the official investigation report remains yet unpublished. Specifically *BBSwift*'s main malware binary contains capabilities which allow for a good evaluation of both frameworks. Our analysis highlights the potential for further standardization and enhancement of CTI reporting, as well suggestions for further extension of the frameworks at hand. With this paper we make the following contributions:

- We have compared ATT&CK and MBC in context of their application for dissemination of malware-based CTI.
- We show examples of significant malware behavior that currently is not captured by MBC nor ATT&CK.
- We show that both frameworks offer partial abstractions of malicious behavior and how this impacts CTI.

The remainder of this paper is structured as follows: Section 2 introduces ATT&CK and MBC and their use in CTI, Section 3 provides context around the intrusion in which the malware was used, Section 4 describes the methodology of our analysis. Section 5 then classifies our malware reverse-engineering results using ATT&CK MBC. In Section 6 we discuss malicious capabilities that could not be mapped to ATT&CK or MBC or both. Section 7 discusses related work and Section 8 summarizes our findings.

II. ATT&CK and MBC

As described in the original white paper by MITRE, the ATT&CK framework is a curated knowledge base and taxonomy of adversary behavior [2]. It reflects the phases of and actions within an adversary's attack life-cycle, which are the attacker objectives, classified as tactics. Each tactic holds techniques which describe how a tactic is achieved on a technical level, with sub-techniques providing additional granularity.

The Malware Behavior Catalog (MBC) is a collection of malware objectives and behaviors, developed by MITRE to support malware analysis. It carries similarities with Malware Attribute Enumeration and Characterization (MAEC), which is scheduled to be discontinued. Key concepts of MBC are objectives, behaviors, methods and micro-behaviors, which are abstractions of characteristics accomplished by malicious code. Objectives describe *why* malware performs a certain action. That action, *what* technique is used to implement the objective, is captured one level deeper, in behaviors. The definition of behaviors is further deepened by methods, which provide further detail around the specific implementation of the behavior. A behavior element always has a method element attached. MBC also includes micro-behavior, which using several categories describes system-level behavior that is not necessarily malicious, but is suspicious especially when triggered during malware runtime. MBC maps to ATT&CK, not vice versa, and both map to STIX 2.1.

The *malware behavior* described by MBC is what differentiates it from ATT&CK, which describes *adversary behavior*. MBC focuses on behavior and characteristics of malware binaries, whereas ATT&CK focuses on adversary behavior (potentially including application of malware). It can thus be argued that MBC focuses on *malware analysis*, where ATT&CK has a focus on *intrusion analysis*. The Diamond Model of Intrusion Analysis characterizes the relationship between four typical components of intrusion campaigns. Based on this model, Figure 1 shows how MBC and ATT&CK differ in their value proposition to CTI. The ATT&CK framework and knowledge base allow for a contextual understanding of the various features of an intrusion campaign and their interconnections, whereas MBC strictly focuses on the capability. MBC allows for a specific and deeper understanding of the malicious tools than ATT&CK, which focuses on aggregate TTPs. Table I compares key various aspects of both frameworks against each other. Given the different use cases of both frameworks, it is remarkable that most industry reports currently use ATT&CK,

TABLE I: Comparison of MBC and ATT&CK characteristics

	MBC	**ATT&CK**
Why	Objectives	Tactics
How	Behaviors	Techniques
Low-level How	Methods	Sub-Techniques
Columns	21	14
Behavior	Malware binaries	Human adversaries
Lifecycle	Malware runtime	Intrusion timeline
Primary use case	Malware analysis	Campaign analysis

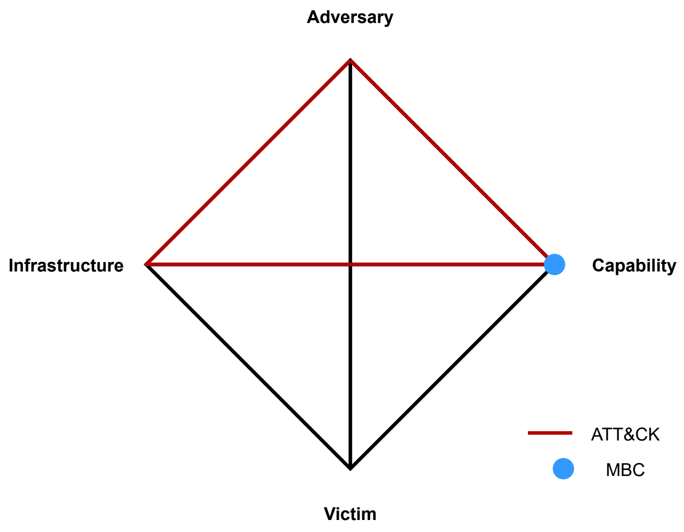


Fig. 1: Diamond Model of Intrusion Analysis with features covered by ATT&CK (red) and MBC (blue).

even when the sole subject of analysis is a malware sample. This is probably due to ATT&CK being the first mover, but based on a quick comparison of their specifications, MBC appears to be better suited to capture malware-based CTI than ATT&CK, which is appropriate when artifacts of the full intrusion are available.

III. The Bangladesh Bank Heist

On February 4, 2016 an adversary attempted to illicitly transfer 951 million USD from a reserve account owned by the central bank of Bangladesh. The adversary had obtained a presence in the Bangladesh Bank network long before through a phishing and potentially assisted by complicit internal actors [3]. After obtaining an initial foothold, the adversary deployed Remote Access Trojans (RATs), keyloggers and screengrabbers in order to gain an understanding of the bank's ordinary course of business. An important part of the traffic flow of every major bank is transacting with the SWIFT network, an international transaction layer between banks. The adversary was able to interact with Alliance Access, the client software required for banks to interact with the SWIFT network.

The attackers attempted to move 951 million USD from the bank's custodial account with the Federal Reserve Bank of New York. Of this amount, spread out over 30 malicious transactions, 850 million was blocked and 81 million USD is still missing [4]. This was due to two reasons. The adversary had setup multiple bank accounts in Sri Lanka and the Philippines to redirect the stolen funds to. Some of these were bank accounts were opened at a branch of the Philippines RCBC bank at Jupiter Street, called RCBC Jupiter. By coincidence, Jupiter was also the name of an Iranian shipping company and oil vessel under economic sanctions [5], which caused the Federal Reserve to block most of the transactions. Only 5 were authorized by the Federal Reserve, which amounts of 101 million USD, of which 20 million was reversed to its origin by the receiving Sri Lankan bank, who spotted a spelling mistake in the recipient name.

The attack is not only significant due to its financial impact, but also due the malware used. This was tailored to the Bangladesh Bank's procedures, not only influencing its deployment of Alliance Access, but also undermining the bank's internal manual verification process of SWIFT transactions based on print-outs.

A. BBSwift Malware

According to an FBI flash alert 4 malicious file artifacts were used in the attack on Bangladesh Bank [6]. In this paper we focus on *evtdiag.exe*, which interacted with the SWIFT client software. We selected this malicious binary because it contains all logic to interact with the SWIFT database and its code is not obfuscated by its authors, which allows for straightforward replication of our research. Deobfuscation is an activity that is normally not easily reproduced, as many paths through program execution can be taken, thus hindering replication. Metadata on evtdiag.exe is available in Table II.

B. Attribution

From a CTI perspective the incident at Bangladesh Bank was significant, as nation state actors usually do not manifest criminal intent. Several US government agencies have attributed the attack to the state-sponsored North-Korean HIDDEN COBRA group [5]. Many attacks with a financial objective have been attributed to this group, targeting banks, the SWIFT network, Initial Coin Offerings, ATMs and cryptocurrency trading platforms and apps. The attack on Bangladesh Bank was attributed to BeagleBoyz, a subset of HIDDEN COBRA activity focused on hacking banks [7].

IV. Methodology

A sample of the BBSwift malware was sourced from MalShare, based on its SHA-256 hash as listed in Table II. The sample was then loaded into an instance of IDA Professional 7.6.210427 which also included the x86 decompiler add-on. We chose IDA due to our familiarity with its appliance in our professional practice. A comparable free open-source alternative that allows for replication of our analysis is provided by the National Security Agency with Ghidra.

We analyzed the malware binary in IDA using a typical reverse engineering process as described by Votipka and Rabin [8], including the identification and location of file sections, headers strings and API calls. We then analyzed the data and control flow paths of the malware and inspected malicious and suspicious functions based on code, constants and subroutines, which we describe in Section V, using the graphical view of disassembled functions. This is also available in Ghidra.

The results of our analysis were mapped to MBC v2.1 based on the Markdown specification available on Github [1]. For ATT&CK, we built on a mapping of static binary features developed for an earlier project [9]. Techniques and subtechniques added to ATT&CK in the meantime were included based on the JSON specification of ATT&CK v9.

TABLE II: Metadata of Analyzed Malware

CARO name	TrojanSpy:Win32/BBSwift.A
File type	Win32 PE
File name	evtdiag.exe
MD5 hash	24d76abbc0a10e4c977a28b33c879248
Import hash	aab0b4b819af30b63a6352a276e87d83
SHA-256 hash	4659dadbf5b07c8c3c36ae941f71b63
	1737631bc3fded2fe2af250ceba98959a

V. Analysis of the BBSwift Malware

In this section we report our findings of our manual reversing of the malware in IDA. After an overview of the general function flow of the malware derived from IDA, we categorize the malware's capabilities using ATT&CK and MBC. Figure 2 provides a visual overview of the classification of our results in both frameworks. The second row shows the categorization of observed techniques in ATT&CK, where the third and fourth row respectively show MBC behaviors and micro-behaviors. Observed capabilities that could *not* be plotted on either of the frameworks - and which thus would normally lead to signal loss - will be discussed in the subsequent section.

An adversary needs to gain an initial foothold on a system in order to deploy and execute attacker-controlled files. In the Bangladesh Bank intrusion, the malware was deployed on a server running Alliance Access, which is a messaging interface required to interact with the SWIFT network. Alliance Access depends on a native, embedded Oracle database to store incoming and outgoing messages for audit purposes. The delivery mechanism of the malware, as well how it was initially executed (e.g. user execution after opening a malicious email attachment) can logically not be derived from a malware binary. While it is speculated that in this case the malware was delivered via phishing emails with malicious document attachments, speculations of insider involvement also exist. Thus definitive *intelligence* about this does not exist and our analysis below starts with the malware attempting to maintain persistence. In this section we describe the general function flow of the application, a detailed description of *BBSwift*'s capabilities using ATT&CK and MBC is available in Table III.

- The **main()** function (0x409DB0) usually invokes the execution of portable executables, which in this case takes the command line arguments *resume, pause, on, off* and *queue* to control print operations which will be described later (0x402580). After loading command and control information from an RC4-encrypted configuration file, it establishes itself a service using *StartServiceCtrlDispatcherA()*. Using *sprintf(Dir, %c:\\Users\\%s\\AppData\\Local\\%s", Buffer[0], aAdministrator, aAllians);* it creates a stealth directory to store log files and a configuration file (0x40F0A4). It circumvents a validation step by the victim application Alliance Access, by enumerating running processes to find *liboradb.dll*, which is used by Alliance Access to validate incoming and outgoing SWIFT messages. If found, it will then perform a small NOP slide of two instructions, which deceives Alliance Access to regard messages as authentic. The code segment through which this is implemented is shown in Listing 1.
- After access to the Alliance Access application, which handles SWIFT transactions, is obtained the malware performs command-line SQL SELECT queries to monitor for new records with the code *BBHOBDDHA* and *Log%%*, which signal SWIFT authentication activity by the victim bank (0x40F9B4).
- Based on the type of authentication observed, the malware is configured to send HTTP/1.1 GET requests —*N*, —*C* and —*O* to an attacker-controlled remote server. Likely after this manual attacker intervention takes place via a command and control mechanism not included in the malware binary.
- Strings hard-coded in the malware are used to perform SQL SELECT statements (0x40F220) on the command line, the contents of which are exfiltrated to temporary files in the attacker-controlled directory created earlier. After this, the legitimate victim-issued SWIFT transactions are deleted from the local SWIFT database, issuing DELETE commands in SQL (0x4027C0). Potentially using the Command Control channel, the attacker can use this information to compose temporary SQL files with malicious transactions. The malware then has the ability to perform SQL queries using statements from a temporary file as input.
- As an audit step in daily operations, the victim Bangladesh Bank had Alliance Access configured to print confirmations of issued outgoing transactions. To avoid detection of its activities via this route, the adversary obscures its malicious transactions by modifying the print controller service and spoofing the printouts (0x409460).

In the next section we will focus on malicious capabilities that we could not map to either or both the frameworks.

Listing 1: C-Pseudocode fragment of function at *0x402580*

```
  if ( !VirtualProtectEx(hProcess, v7, 2u,
         PAGE_EXECUTE_READWRITE, (PDWORD)&hProcess)
    || !ReadProcessMemory(v3, v4, &Buffer, 2u, &
         NumberOfBytesRead) )
  {
    return -1;
  }
  if ( a3 )
  {
    if ( (_WORD)Buffer != NOPNOP[0] )
      return -1;
    v5 = WriteProcessMemory(v3, v4, JNZ6, 2u, &
         NumberOfBytesWritten);
  }
  else
  {
    if ( (_WORD)Buffer != JNZ6[0] )
      return -1;
    v5 = WriteProcessMemory(v3, v4, NOPNOP, 2u, &
         NumberOfBytesWritten);
  }
  if ( !v5 )
    return -1;
  VirtualProtectEx(v3, v4, 2u, (DWORD)hProcess, &
       flOldProtect);
  return 0;
}
```

TABLE III: Observed capabilities in *evtdiag.exe* and mappings to ATT&CK and MBC

	Capability	**Implementation**	**Function Address**	**ATT&CK**	**MBC**
1	Starting application as service using command line argument -svc	StartServiceCtrlDispatcherA()	0x409D7C	Execution::Executio Guardrails	Anti-Behavioral Analysis::Conditional Execution::Runs as Service
2	Establish service based on issued command-line argument	CreateProcessA()	0x401C91	Persistence::Create or Modify System Process::Windows Service	
3	Request memory allocation for service	VirtualProtectEx(), PAGE_EXECUTE_READWRITE	0x4025B8, 0x4025A0		Memory::Allocate Memory
4	Get SYSTEM access	SeDebugPrivilege	0x4023F2	Privilege Escalation::Access Token Manipulation	
5	Create attacker-controlled directory	\Users\Administrator\AppData\Local\A	0x401127		File System::Create Directory
6	Place malicious payload in attacker-controlled directory	Copy gpca.dat to attacker-controlled directory	0x401AE7		File System::Copy File
7	Check if files are placed correctly	GetFileAttributes()	0x401462		File System::Get File Attributes
8	Decrypt (RC4) malicious configuration file gpca.dat	Decryption key 4e381fa77f08ccaa0d56edeff9ed08ef	0x40F020		Cryptography::Encryption Key::RC4 KSA; Cryptography::Decrypt Data::RC4
9	Read malware configuration from gpca.dat	ReadFile() gpca.dat	0x40F020		
10	Create log file	WriteFile() recas.dat	0x40BBA4		File System::Writes File
11	Encode log file with XOR	Tight loop	0x40BB42		Data::Encode Data::XOR
12	List running system processes	CreateToolhelp32Snapshot()	0x4023B0	Discovery::Process Discovery	Process:Enumerate Threads
13	Hijack liboradb.dll	Patch memory offset (NOP sled) to force validation	0x402580		
14	Monitor SWIFT database for login and logout activity	WHERE JRNL_DISPLAY_TEXT LIKE ’%%LT BBHOBDDHA: Log%%’ ORDERBY JRNL_DATE_TIME DESC) A	0x408EA0		
15	Monitoring SWIFT database for account manipulation	SELECT MESG_S_UMID FROM SAAOWNER.MESG_%sWHERE ...	0x402900		
16	Parse obtained SWIFT transactions in temporary file	WriteFile()	0x408550		File System::Writes File
17	Delete benign transaction records from database	DELETE FROM SAAOWNER.TEXT_%s WHERE TEXT_S_UMID = ’,27h,’%s’,27h,’ DELETE FROM SAAOWNER.MESG_%s WHERE MESG_S_UMID = ’,27h,’%s’,27h,’	0x4027C0		
18	Insert malicious transactions in temporary file	WriteFile()	0x406D80		
19	Command line execution of transactions	cmd.exe /c echo exit — ”%s” -S / as sysdba @%s >”%s”’	0x40F8EC		Execution::Command and Scripting Interpreter
20	Manipulate account balance	SELECT MESG_S_UMID FROM SAAOWNER.MESG_%s WHERE MESG_SENDER_SWIFT_ADDRESS LIKE ’,27h,’%%%s%%’,27h,’ AND MESG_FIN_CCY_AMOUNT LIKE %%%s%%	0x40F484	Impact::Data Manipulation::Stored Data Manipulation	Impact::Compromise Data Integrity
21	Hijacking printer control	Replace print controller with nroff.exe and append .bak	0x40FCFC		Process::Create Thread
22	Create temporary print files	Create PRT files with spoofed transaction confirmation	0x40F6B8		
23	Print spoofed transaction confirmation statements	Obfuscated PCL	0x4095B0		Anti-Static Analysis::Disassembler Evasion::Argument Obfuscation
24	Command and Control over HTTP	Setup traffic to 196[.]202[.]103[.]174	0x40F030	Command and Control::Application Layer Protocol::Web Protocols	Command and Control::Send Data; Command and Control::Receive Data; HTTP Communication::Create Request; HTTP Communication::Send Request; HTTP Communication::Connect to Server; HTTP Communication::Read Header

VI. Non-mappable Capabilities

In our analysis we have observed behavior and techniques that fall outside either or both of MBC's and ATT&CK's matrices of techniques. As these are highly relevant examples of adversary TTPs, we discuss in this section why this indicates that ATT&CK and MBC currently deliver partial abstractions of adversary behavior. The malware analyzed carries capabilities with a significant and documented real-world impact, which currently is not accounted for by MBC and ATT&CK. We deem this is a clear opportunity for improvement for both frameworks, hence why we consider how it *could* be mapped to these frameworks. The *BBSwift* malware was custom developed to be deployed as part of a targeted intrusion with a specific goal. This goal was to interact with the SWIFT network to transfer funds away from various custodial bank accounts to attacker-controlled accounts. On the transaction level this was achieved by manipulating the SWIFT client database to execute fraudulent transactions. Printouts of transaction confirmations used for manual verification were forged to further obfuscate the malicious SWIFT transactions.

A. Buffer overflow using NOP sled

The NOP sled, also called NOP slide, is one of the most common methods of performing a buffer overflow. As shown in Figure 1, it provides a critical mechanism for the *BBSwift* to gain full control over the Alliance Access application. While ATT&CK has an Execution technique called Exploitation for Client Execution, we believe this is too broad to capture all means of client-side exploitation. Both ATT&CK and MBC framework could benefit from adding sub-techniques such as Buffer Overflow, Heap Spray and User After Free.

B. Command-line SQL Execution

The adversary managed to interact with the SQL database containing current and historical SWIFT transactions. As discussed earlier, the first step is placing the NOP instruction to bypass the security verification. After this, the malware is able to perform hard-coded SQL SELECT and DELETE queries to extract, execute and remove SWIFT transactions from the database. Using the access obtained, the adversary also monitored for benign SWIFT authentication traffic by the victim bank. The SQL SELECT statements performed by the malware would classify in ATT&CK and MBC as a Collection technique, but currently the frameworks do not have techniques to describe data collection from a database. ATT&CK includes a technique named *Data from Information Repositories*, with sub-techniques covering collection from Confluence and SharePoint instances. This could argue for adding sub-techniques, but more importantly should question the practicality of including concrete frameworks in a meta framework. The DELETE queries could potentially be classified as Impact or Defense Evasion techniques.

C. Printer Spoofing

As part of a manual validation process, the victim bank had its SWIFT server sent transaction confirmations to a printer. While these are only printed after execution of the transaction, these should allow for timely intervention during regular office hours. To let malicious transactions go unnoticed to bank staff reviewing the printouts, the authors of the *BBSwift* malware implemented functionality to spoof the printouts. Technically this is implemented by spoofing the PRT files which are used by the print spooler, which is in this case *nroff.exe*. The malware binary can also take command line arguments to control the status of a printer, as shown in Listing 2. The impact of this technique is that malicious transactions are not observed as they take place and thus it could be added to the frameworks as a Defense Evasion or Impact technique.

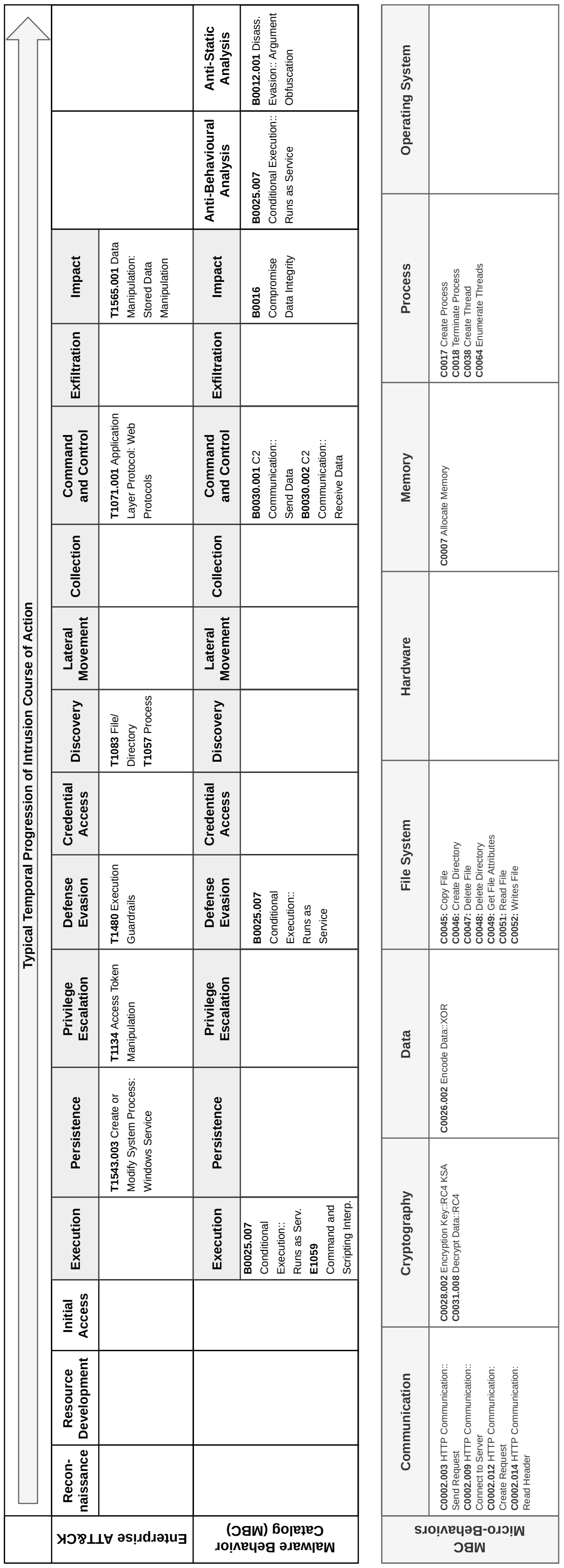
D. Theft of funds

The final objective of the attacker with this malware, was the diversion of funds to attacker-controlled accounts. While this would also classify as behavior or technique serving the Impact tactic, this direct financial impact is currently not recognized in the abstractions currently provided by both frameworks. We believe it should be, as this motive is not exclusive to this attack, but also applied by cryptocurrency-stealing malware which empties software cryptocurrency wallets on end-user systems.

The techniques above are not covered in the current versions of MBC and ATT&CK and could be potential additions to the frameworks. Another observation for improvement of both frameworks is their documentation of techniques appearing in multiple tactics (ATT&CK) or objectives (MBC). MBC has behaviors serving different objectives, like ATT&CK has multi-tactic techniques. While one technique or behavior can clearly serve multiple adversary objectives, the documentation is not clear whether this behavior should then be attached to all three or a single objective. On one hand the classification of techniques into single categories complicates end-user application of the frameworks, as it forces inferring attacker motivation for multi-tactic techniques. On the other hand adding it to all tactics misleads consumers of analysis output, as sub-techniques of Process Injection like Process Hollowing clearly serve as Defense Evasion, not Privilege Escalation.

Listing 2: Obfuscated string *HP LaserJet 400 M401 PCL* manually constructed on the stack at *0x4033E0*

```
mov     [esp+5B0h+var_59F], 48h ; 'H'
mov     [esp+5B0h+var_59D], cl
mov     [esp+5B0h+var_59C], 4Ch ; 'L'
mov     [esp+5B0h+var_59B], 61h ; 'a'
mov     [esp+5B0h+var_59A], 73h ; 's'
mov     [esp+5B0h+var_599], 65h ; 'e'
mov     [esp+5B0h+var_598], 72h ; 'r'
mov     [esp+5B0h+var_597], 4Ah ; 'J'
mov     [esp+5B0h+var_596], 65h ; 'e'
mov     [esp+5B0h+var_595], 74h ; 't'
mov     [esp+5B0h+var_594], cl
mov     [esp+5B0h+var_592], 30h ; '0'
mov     [esp+5B0h+var_591], 30h ; '0'
mov     [esp+5B0h+var_590], cl
mov     [esp+5B0h+var_58F], 4Dh ; 'M'
mov     [esp+5B0h+var_58D], 30h ; '0'
mov     [esp+5B0h+var_58C], bl
mov     [esp+5B0h+var_58B], cl
mov     [esp+5B0h+var_589], 43h ; 'C'
mov     [esp+5B0h+var_588], 4Ch ; 'L'
```



Typical Temporal Progression of Intrusion Course of Action

Enterprise ATT&CK	**Recon-naissance**	**Resource Development**	**Initial Access**	**Execution**	**Persistence**	**Privilege Escalation**	**Defense Evasion**	**Credential Access**	**Discovery**	**Lateral Movement**	**Collection**	**Command and Control**	**Exfiltration**	**Impact**		
					T1543.003 Create or Modify System Process: Windows Service	**T1134** Access Token Manipulation	**T1480** Execution Guardrails		**T1083** File/ Directory **T1057** Process			**T1071.001** Application Layer Protocol: Web Protocols		**T1565.001** Data Manipulation: Stored Data Manipulation		
Malware Behavior Catalog (MBC)				**Execution**	**Persistence**	**Privilege Escalation**	**Defense Evasion**	**Credential Access**	**Discovery**	**Lateral Movement**	**Collection**	**Command and Control**	**Exfiltration**	**Impact**	**Anti-Behavioural Analysis**	**Anti-Static Analysis**
				B0025.007 Conditional Execution:: Runs as Serv. **E1059** Command and Scripting Interp.			**B0025.007** Conditional Execution:: Runs as Service					**B0030.001** C2 Communication:: Send Data **B0030.002** C2 Communication:: Receive Data		**B0016** Compromise Data Integrity	**B0025.007** Conditional Execution:: Runs as Service	**B0012.001** Disass. Evasion:: Argument Obfuscation

MBC Micro-Behaviors	**Communication**	**Cryptography**	**Data**	**File System**	**Hardware**	**Memory**	**Process**	**Operating System**
	C0002.003 HTTP Communication:: Send Request **C0002.009** HTTP Communication:: Connect to Server **C0002.012** HTTP Communication: Create Request **C0002.014** HTTP Communication: Read Header	**C0028.002** Encryption Key::RC4 KSA **C0031.008** Decrypt Data::RC4	**C0026.002** Encode Data::XOR	**C0045:** Copy File **C0046:** Create Directory **C0047:** Delete File **C0048:** Delete Directory **C0049:** Get File Attributes **C0051:** Read File **C0052:** Writes File		**C0007** Allocate Memory	**C0017** Create Process **C0018** Terminate Process **C0038** Create Thread **C0064** Enumerate Threads	

Fig. 2: Visual comparison of MBC and ATT&CK mappings of analysis results.

VII. Evaluation

We performed an analysis of *BBSwift*, a piece of malicious software used by a threat group referred to as HIDDEN COBRA in a campaign against the Bangladesh Bank. The behavioral characteristics of the malware binary allowed for a detailed comparison of the abstraction layers of adversary action provided by the CTI frameworks ATT&CK and MBC. In general it can be argued that MBC lends itself better to the description of malware capabilities, observed in static and dynamic analysis of malware. ATT&CK is better suited to describe attacker behavior observed over the course of an attack. This may include malware behavior, but also manual attacker intervention and reconnaissance activities.

BBSwift's capabilities for buffer overflow, SQL database permutation, printout forging and its direct financial impact are characteristics of critical importance to incident responders, intelligence analysts and other stakeholders which currently are not recognized by ATT&CK and MBC. As we could not categorize these capabilities into ATT&CK, we found that ATT&CK is not fine-grained enough to encapsulate all analysis details from a piece of advanced malware like *BBSwift*. There might be different reasons why these capabilities are not recognized in current versions of the framework. While it can be argued that including NOP slides as a sub-technique for client-side exploitation is too specific for a meta intrusion framework like ATT&CK, for other techniques it already includes highly specific sub-techniques. The direct financial impact of the diversion of funds might be considered too uncommon for inclusion in an enterprise intrusion framework, on the other hand it is significant to be excluded. Any significant technique not included potentially leads to loss of signal, which is undesirable in intelligence contexts where factual information is inherently scarce. MBC provides a more detailed level of abstraction, with its micro-behaviors capturing suspicious and malicious behavior that ATT&CK does not recognize, but still leaving some to be desired. A potential drawback of MBC's micro-behaviors, which are currently in beta, is that the added granularity adds noise to the signal, capturing obvious details that any benign binary file exhibits, without adding intelligence value.

In practice, many reports and vendor solutions apply a very broad interpretation in mapping analysis to ATT&CK techniques. This can provide either very rich or thin reporting, depending on how "liberal" one assigns techniques and behaviors to analysis outcomes. MITRE does not provide methodology or documentation on how, when - and more importantly when not - to classify certain techniques. Some of ATT&CK's and MBC's specifications for techniques and behaviors are quite broad, which in practice leads to different outcomes depending on who performs the plotting. In the spirit of these frameworks as an abstraction layer of real-world adversary behavior, one should maintain precision and clarity, or analysis integrity is negatively affected. Many industry reports currently do not provide methodological justification of how techniques are assigned to analysis outcomes.

For many industry reports and commercial detection tools, ATT&CK is currently the CTI framework of choice. Even when the underlying analysis mechanism is purely malware-based. This is not so much a problem of ATT&CK or MBC, it is a symptom of product development and framework adoption by industry players. As Figure I clearly shows, they use a framework that covers the entire intrusion chain, where their tooling only really supports one of the vertices (malware). This is important, as the abstractions offered by frameworks as ATT&CK and MBC are often key inputs to lower-level tactical CTI decision-making. Due to its richer description of binary behavior, using MBC allows for a more accurate and effective mitigation of endpoint threats than ATT&CK does.

MBC could benefit from the same community adoption and consequent contributions that ATT&CK has enjoyed. At the time of writing this article, in October 2021, its micro-behaviors are still in beta and have a very short description. The full description of the Operating System micro-behavior Console reads *"Malware modifies the console"*. As we could not map behaviors such as *BBSwift*'s interaction with a printer, the actual micro-behaviors could be further extended as well. MBC would further benefit from using references to academic literature and industry or intelligence reports.

The widespread industry adoption of ATT&CK compared to MBC can be explained as first mover advantage. Our analysis however shows that for malware-based CTI reporting and solutions, MBC provides more granularity to reporting than ATT&CK. More importantly we like to argue that practitioners should decide about adequate framework usage on a per-case basis, rather than blindly picking the industry standard.

VIII. Related Work

Various authors have considered the use of taxonomies for CTI. Many different taxonomies exist, but stated by Mavroeidis et al. many of these lack conceptual links and fail to use ontology axioms and constructs [10]. While taxonomies such as ATT&CK, STIX, and models such as the Diamond Model and the Pyramid of Pain have contributed to the field, they also indicate a need for further standardization of the CTI field [11]. ATT&CK was introduced in a 2016 whitepaper by Strom et al. [2]. MBC currently does not have a white paper nor a dedicated web page. Expressions of MBC in Markdown and STIX 2.1 JSON are maintained on a Github repository [1]. A recent conference talk providing more background information on its conceptual relation with ATT&CK is also available [12].

With regards to reporting on the attack on the Bangladesh Bank, the official investigation report has not been released as of yet, five years after the fact. A concise analysis of the malware based on samples leaked online was performed by defense contractor BAE Systems [13].

IX. Conclusion

In this paper we have mapped the results of our analysis of the *BBSwift* malware to the CTI frameworks ATT&CK and MBC to show how using a framework implicates analysis outcomes. We have shown that while MBC provides more granularity than ATT&CK to reporting malware analysis results, both frameworks still have blind spots. ATT&CK technique listings in CTI reports, APT group overlays in ATT&CK Navigator and efforts to benchmark commercial detection capabilities using ATT&CK have contributed to the standardization of CTI to a large extent. For a majority of CTI consumers, they provide an analysis output serving as a *single source of truth* to decide on follow-up for mitigation and detection. We hope our analysis contributes to an understanding that practitioners should adopt more critical stance toward the selection of a CTI framework on a per-case basis, to avoid these becoming *single points of (intelligence) failure*.

This paper serves as a conceptual work which provides insight into how CTI frameworks can limit analysis. While they are useful and vital standardization efforts, they could also enforce analyst tunnel vision if used their applicance does not align with the target of analysis. Incorrect use might lead to omission of important analysis facts and thus alternative mitigation scenarios. Future extension of this work would need to further focus on differences in abstraction between both frameworks, perhaps based on a large-scale analysis of a corpus of malware samples.

Acknowledgment

The authors would like to thank Dr. Desiree Beck for her feedback which greatly contributed to the analysis.

References

[1] Mitre, "Malware Behavior Catalog." [Online]. Available: https://github.com/MBCProject
[2] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK™: Design and Philosophy," 2018.
[3] Wall Street Journal, "FBI Suspects Insider Involvement in $81 Million Bangladesh Bank Heist - WSJ," 2016. [Online]. Available: https://www.wsj.com/articles/fbi-suspects-insider-involvement-in-81-million-bangladesh-bank-heist-1462861549
[4] Reuters, "Special Report: How the New York Fed fumbled over the Bangladesh Bank heist — Reuters," 2016. [Online]. Available: https://www.reuters.com/article/us-cyber-heist-federal-special-report-idUSKCN1011AJ
[5] U.S. Department of the Treasury, "United States Exposes Iranian Shipping Scheme," 2013. [Online]. Available: https://www.treasury.gov/press-center/press-releases/pages/jl1879.aspx
[6] Federal Bureau of Investigation, "FBI Flash A-000073-MW," Tech. Rep., 2016.
[7] CISA, Treasury, FBI, and USCYBERCOM, "Joint Cybersecurity Advisory FASTCash 2.0: North Korea's BeagleBoyz Robbing Banks," 2020.
[8] D. Votipka, S. Rabin, K. Micinski, J. S. Foster, and M. L. Mazurek, *An Observational Investigation of Reverse Engineers' Processes*, 2020.
[9] K. Oosthoek and C. Doerr, "SoK: ATT&CK Techniques and Trends in Windows Malware," *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST*, vol. 304 LNICST, pp. 406–425, oct 2019.
[10] V. Mavroeidis and S. Bromander, "Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence," *Proc. - 2017 Eur. Intell. Secur. Informatics Conf. EISIC 2017*, vol. 2017-January, pp. 91–98, dec 2017.
[11] K. Oosthoek and C. Doerr, "Cyber Threat Intelligence: A Product Without a Process?" *Int. J. Intell. CounterIntelligence*, vol. 34, no. 2, pp. 1–16, 2020.
[12] Mitre, "Standardized reporting with the Malware Behavior Catalog - VB2021 localhost," 2021.
[13] BAE Systems, "BAE Systems Threat Research Blog: Two bytes to $951m," 2016. [Online]. Available: https://baesystemsai.blogspot.com/2016/04/two-bytes-to-951m.html