C3PO: Large-Scale Study of Covert Monitoring of C&C Servers via Over-Permissioned Protocol Infiltration

Jonathan Fuller, Ranjita Pai Kasturi, Amit Sikder, Haichuan Xu, Berat Arik
Vivek Verma, Ehsan Asdar, Brendan Saltaformaggio
Georgia Institute of Technology

ABSTRACT

Current techniques to monitor botnets towards disruption or takedown are likely to result in inaccurate data gathered about the botnet or be detected by C&C orchestrators. Seeking a covert and scalable solution, we look to an evolving pattern in modern malware that integrates standardized *over-permissioned* protocols, exposing privileged access to C&C servers. We implement techniques to detect and exploit these protocols from over-permissioned bots toward covert C&C server monitoring. Our empirical study of 200k malware captured since 2006 revealed 62,202 over-permissioned bots (nearly 1 in 3) and 443,905 C&C monitoring capabilities, with a steady increase of over-permissioned protocol use over the last 15 years. Due to their ubiquity, we conclude that even though over-permissioned protocols allow for C&C server infiltration, the efficiency and ease of use they provide continue to make them prevalent in the malware operational landscape. This paper presents C3PO, a pipeline that enables our study and empowers incident responders to automatically identify over-permissioned protocols, infiltration vectors to spoof bot-to-C&C communication, and C&C monitoring capabilities that guide covert monitoring post infiltration. Our findings suggest the over-permissioned protocol weakness provides a scalable approach to covertly monitor C&C servers, which is a fundamental enabler of botnet disruptions and takedowns.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**;

KEYWORDS

Malware; Botnets; Covert Infiltration; Symbolic Execution

ACM Reference Format:
Jonathan Fuller, Ranjita Pai Kasturi, Amit Sikder, Haichuan Xu, Berat Arik and Vivek Verma, Ehsan Asdar, Brendan Saltaformaggio. 2021. C3PO: Large-Scale Study of Covert Monitoring of C&C Servers via Over-Permissioned Protocol Infiltration. In *Proceedings of Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15–19, 2021 (CCS '21),* 14 pages.
https://doi.org/10.1145/3460120.3484537

1 INTRODUCTION

Botnet disruptions and takedowns are driven by Command and Control (C&C) server monitoring before any action is taken and after to gauge success. This means that disruption or takedown attempts are not only provably necessary, but must be targeted and effective [1]–[9]. Modern approaches can be categorized as passive or active monitoring. Passive monitoring (e.g., sensor node injection) is coarse-grained and may not give accurate insights into the botnet [10], [11], i.e., the number and location of the victims and the extent of damages incurred. It also requires a full reverse engineering effort to maintain sensor nodes making this approach not widely used [11]. Therefore, active monitoring is the preferred approach [1], [10], generally providing better insights into botnet operations. However, active monitoring techniques, including remote penetration testing [12]–[16] and domain seizure [1], [3]–[6], [17], are *noisy* making them easily detectable. Seeking a better solution, this research proposes that standard protocols, which are increasingly used by botnets, can be leveraged for *general and covert* C&C server monitoring.

In previous botnet disruption and takedown attempts, authorities first monitored the C&C server to prove malware as the catalyst for incurred damages before legal permission was granted for counteraction [18]. Yet, accurate monitoring goes beyond determining the legality of counteraction. For example, to protect the 2020 election, Microsoft took down 120 of 128 Trickbot C&C servers [19]. Accurately identifying C&C servers pre-takedown (profiling), then tracking successes post takedown (validation), required an in-depth understanding of the peers in the botnet, C&C server locations, and weaknesses to leverage for botnet disruption. Therefore, successful monitoring must result in accurate, legally-admissible information gathered during profiling and remain covert to avoid discovery by C&C orchestrators, prompting defensive evasion or hardening [11], [20], [21]. An ideal solution should provide authorities with a means to access the C&C server under the guise of normal bot operation.

As the *end-host agents* of a C&C orchestrator, bots are entrusted with C&C server access. In fact, attackers are entirely dependent on the information exfiltrated by bots to gain situational awareness in a victim's network. To enable command and control, bots use standard protocols for file transfer, data storage, and message-based communication. However, many standard protocols are over-permissioned, meaning that they provide feature-rich and unfettered access to the server beyond the subset of features implemented by a given client. A similar trend has been observed in benign software where over-permissioned client-side protocols lead to unauthorized server access [22]–[25]. This prompted our key insight: *over-permissioned protocols combined with the trust C&C servers place in their bots expose a scalable opportunity for covert monitoring of C&C servers through protocol infiltration.*

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484537

To explore this insight, a systematic study is needed to identify the evolution of over-permissioned protocol use in malware. Moreover, to conduct such a study, the analysis must be scalable, reproducible, and provide the requisite information to covertly monitor C&C servers through over-permissioned protocol infiltration. The study must expose over-permissioned protocols, how they are being used, and the associated levels of access and recoverable data on the C&C server. Finally, an automated pipeline must be made available to enable the authorities to take action on these common malware weaknesses in future botnet outbreaks.

We turned our attention to how the authorities could recover C&C server access privileges from over-permissioned bots (bots using over-permissioned protocols) allowing them to spoof bot-to-C&C communication. To this end, we designed and implemented C3PO[1], an automated memory-image-based symbolic analysis measurement pipeline. C3PO analyzes a malware memory image to identify (1) over-permissioned protocols, (2) infiltration vectors (i.e., authentication information to spoof bot-to-C&C communication), and (3) C&C monitoring capabilities (i.e., capabilities in the end-host bot that reveal the C&C server's composition and content to guide covert monitoring post infiltration).

Through our collaboration with Netskope, the leading Secure Access Service Edge (SASE) provider, which provides cloud security and networking to more than 30% of the Fortune 100, we used C3PO to study the evolution of over-permissioned protocol use in 200k malware spanning back 15 years. C3PO uncovered 62,202 over-permissioned bots (≈1 in 3). Our empirical measurement revealed several interesting findings: FTP is the most prevalent over-permissioned protocol found in over 79% of all over-permissioned bots. C3PO also identified 443,905 C&C monitoring capabilities (an average of 7 per bot), enabling victim profiling, evidence collection from spyware, and even client-side code reflection. This trend has only increased since 2006, with over 8,000 over-permissioned bots appearing per year in 2018 and 2019. Furthermore, recent bots (since 2015) implemented as many as 3 over-permissioned protocols.

Finally, we present two case studies to demonstrate covert C&C server monitoring through protocol infiltration. We were careful to follow ethical guidelines and adhere to applicable laws when conducting this study. Covert monitoring succeeded and revealed the number of files, their contents, and validation of information inferred by the C&C monitoring capabilities, which will support future botnet disruption and takedown attempts. We are working with Netskope towards the disclosure and remediation of the identified C&C servers. Lastly, we have made C3PO available to the community at: https://cyfi.ece.gatech.edu/.

2 A MOTIVATING EXAMPLE

Botnet disruptions and takedowns rely on accurate C&C server monitoring to profile the botnet beforehand and validate successes after. Consider Sanny, an APT that targets government agencies through spearfishing. After infection, Sanny hijacks Windows service components to enable persistence, deletes dropped files to cover their tracks, and conducts sensitive data exfiltration. The Sanny botnet survived takedown attempts in 2013 [26] and persists today. After botnet monitoring began to fail, an extensive investigation was conducted in 2018, revealing Sanny's C&C server update [27], but this required a tedious manual analysis.

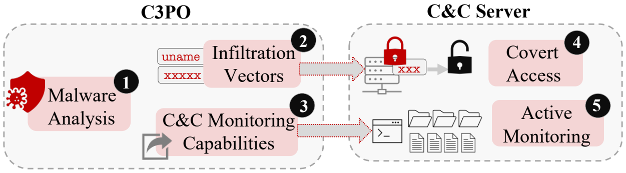


Figure 1: C3PO-enabled Covert Monitoring of Sanny.

The authorities reverse engineered dropped malicious files to investigate the new Sanny variant. At the time, authorities found never-before-seen FTP APIs and authentication credentials throughout the malware binary and configuration files on the infected system, revealing the update to the Sanny C&C server. However, since no further action was taken, they likely did not realize the leverage this provided for covert C&C server infiltration. If they did, the authorities could have also identified the malware capabilities that rely on FTP for interaction with the C&C server. This would have allowed them to reinstate monitoring of the botnet's spread by extracting victim profiles and new bot command updates, all under the covert guise of a trusted FTP connect.

Armed with our key insight, C3PO monitors the C&C server by first identifying over-permissioned protocols, FTP in this case, through their invocation points in the malware. Figure 1 illustrates the sequence of events toward covert C&C server monitoring. During malware analysis ❶, C3PO identified FTP APIs (e.g., `FTPPutFile`) in Sanny which confirmed the updated Sanny C&C server (Table 1, Row 1). C3PO then used Iterative Selective Symbolic Execution (iSSE) to extract infiltration vectors (IVs) from FTP APIs ❷, allowing C3PO to spoof bot-to-C&C communication for infiltration while masquerading as a trusted bot (Table 1, Row 2).

Had authorities realized the leverage FTP provided for botnet infiltration, they could have monitored victim profiles and new bot command updates. C3PO automatically provides this by identifying C&C monitoring capabilities ❸ revealing the C&C's composition and content that authorities can expect post infiltration. C3PO only targets those capabilities that are exploitable, i.e., they interact with the C&C server in a way that can be observed by C3PO when it connects to the C&C server using the same protocol. For example, Sanny performs victim profiling by exfiltrating victim locale information, files, and passwords (from Firefox and Microsoft Outlook) via FTP and used code reflection to execute arbitrary commands on the victim system from a file on the C&C server (Table 1, Row 3). C3PO maps these capabilities to specific files and directories to monitor on the C&C server via FTP protocol infiltration.

Table 1: C3PO's Analysis of the Sanny Malware[27].

Protocol	FTP
Infiltration Vectors	**Username**: cnix_21072852 **Password**: vlasimir2017 **Server**: ftp.capnix.com
C&C Monitoring Capabilities	Victim Profiling, File Exfiltration, Password Stealing, and Code Reflection
Covert Monitoring Outputs	(1)Peer disclosure as victim information is listed as `"<Victim ID>_(#report) | UserName | TimeStamp"` (2) Code Reflection to update the C&C host name

[1] C3PO: **C**overt Monitoring of **C&C** Servers via **P**rotocol Infiltrati**O**n

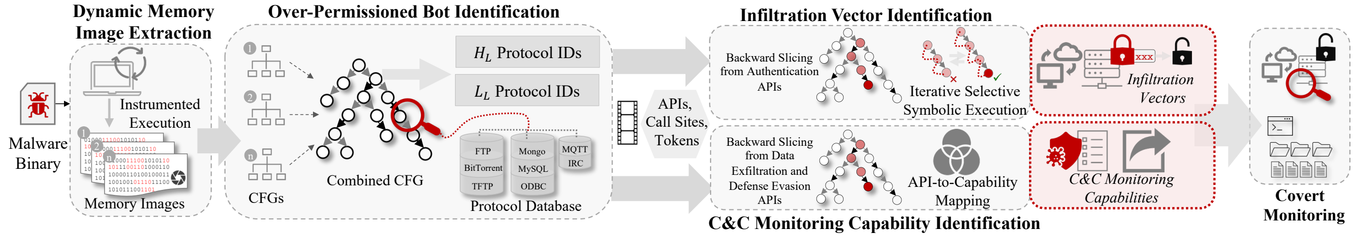

Malware Binary
Dynamic Memory Image Extraction
Instrumented Execution
Memory Images
Over-Permissioned Bot Identification
CFGs
Combined CFG
H_L Protocol IDs
L_L Protocol IDs
FTP
BitTorrent
TFTP
Mongo
MySQL
ODBC
MQTT
IRC
Protocol Database
APIs, Call Sites, Tokens
Infiltration Vector Identification
Backward Slicing from Authentication APIs
Iterative Selective Symbolic Execution
Infiltration Vectors
Backward Slicing from Data Exfiltration and Defense Evasion APIs
API-to-Capability Mapping
C&C Monitoring Capabilities
C&C Monitoring Capability Identification
Covert Monitoring

Figure 2: C3PO Measurement Pipeline.

After C3PO extracts the IVs ❷ and capabilities ❸, it actively monitors the C&C server. C3PO can use the IVs (Table 1, Row 2) to infiltrate ❹ the Sanny C&C server, via the trusted bot-to-C&C channel, and directly locate data from victims ❺ in the form of files containing infected system information and passwords resulting in peer disclosure (Table 1, Row 4) which serves as evidence of computer fraud and abuse. Furthermore, C3PO identified code reflection where the bot orchestrators issue the chip command to the bot to trigger the FTP hostname to update. The ability to monitor this transaction ensures that we maintain persistent covert monitoring irrespective of migrating servers.

In contrast to previous works, C3PO gives the ability to identify, assess, and pursue counteraction via scalable covert monitoring. Notably, C3PO does not attempt to find exploitable vulnerabilities in protocol implementations but instead, leverages the inherent capabilities of the protocol.

3 MEASUREMENT PIPELINE

In collaboration with Netskope, we designed C3PO to study the adoption of over-permissioned protocols in bots and how their use has evolved from April 2006 to June 2020. Our dataset included 200k malware with collection dates spanning back 15 years. This allows us to retroactively deploy C3PO by analyzing each malware sample and give C3PO the vantage point to observe existing trends in the progression of malware development. C3PO identified 62,202 of these as over-permissioned bots totaling 65,739 over-permissioned protocol uses detected across 8,512 malware families. Furthermore, C3PO identified that each bot contains on average 7 C&C monitoring capabilities, totaling 443,905 capabilities identified across our dataset. We hope C3PO provides an automated measurement pipeline to study the over-permissioned bot landscape in the wild and this opportunity for covert botnet monitoring.

Figure 2 shows the four phases of C3PO's automated measurement pipeline that employs a memory-image-based symbolic analysis. Taking a malware binary as input, C3PO conducts *Dynamic Memory Image Extraction* (subsection 3.1) by executing the malware under instrumentation and capturing memory images during this execution for analysis. This provides the best vantage point to bypass malware packing and obfuscation. C3PO transitions to static analysis for *Over-Permissioned Bot Identification* (subsection 3.2) by identifying invocation points for protocol APIs and protocol keywords/commands (tokens). Next, C3PO uses Iterative Selective Symbolic Execution (iSSE) for *Infiltration Vector Identification* to allow the authorities to spoof bot-to-C&C communication for infiltration (subsection 3.3). C3PO then conducts *C&C Monitoring Capabilities Identification* to reveal the composition and content that authorities can expect from the C&C server during infiltration (subsection 3.4). Finally, infiltration vectors can be used for *Covert Monitoring* of the C&C servers to pinpoint data inferred by C&C monitoring capabilities enabling botnet monitoring.

3.1 Dynamic Memory Image Extraction

Malware often employs sophisticated packing and obfuscation techniques that constrain analysis and also inhibit large-scale measurements [28], [29]. Although there are numerous unpacking tools available, modern packing techniques employ robust anti-analysis methods rendering existing solutions mute [28]. While sandboxes or software emulation are viable approaches, they require careful configuration per malware sample/family which is likely to prevent scaling to analyze a large dataset and may accidentally result in introduced errors through incomplete configurations. As a pipeline designed for large-scale measurement, C3PO aims to provide a scalable means of malware analysis through dynamic unpacking and memory image extraction, i.e., taking a snapshot of the malware during normal execution. Ideally, creating a memory image during dynamic execution allows the malware to unpack and deobfuscate itself, leaving C3PO with unpacked and deobfuscated code and execution data to analyze.

Inspired by prior works [28], [30], C3PO extracts multiple memory images during the malware execution by *hooking* Internet and Network (I/N) APIs[2]. This technique is based on two observations: (1) Irrespective of the packing scheme, after unpacking, the malware must invoke I/N APIs to interact with its C&C server. (2) Since recent research has shown that most modern packers have at least two layers of packing [28], if malware memory image extraction is untimely, or at the wrong layer, it will still be packed. Therefore, C3PO extracts multiple memory images by hooking all I/N APIs, as their DLLs are loaded, using a trampoline to replace instructions in the hooked API with a call to our custom code that writes the memory image to a file and returns to the trampoline. Each memory image contains the execution context (i.e., register values, stack, program counter, etc., at the time of memory image extraction) which ensures that malware analysis begins from a valid execution point in the malware.

After extracting malware memory images, C3PO proceeds to the memory-image-based analysis to measure the prevalence of over-permissioned protocol use and the leverage they provide to covertly monitor C&C servers.

[2] I/N APIs allow the malware to interact with FTP and HTTP protocols to access Internet resources.

3.2 Over-Permissioned Bot Identification

Over-Permissioned bots use over-permissioned protocols that authorities can leverage to covertly monitor C&C servers. We construct a *protocol database* that C3PO can reference as it confirms the invocation of protocol identifiers (APIs and protocol keywords or commands, i.e., tokens) validating protocol use. If the bot is over-permissioned, C3PO outputs the protocol APIs, tokens, and call sites for later analysis.

3.2.1 Protocol Implementations. Protocols are implemented using low-level functions or high-level, built-in library functions to achieve the same overall functionality. We, therefore, categorize protocol implementations as low-level (L_L) or high-level (H_L) for our measurement study.

$\mathbf{H_L}$ Implementations. Protocol-specific APIs are used for H_L protocol implementations (e.g., `SQLConnect`), which reduce flexibility in modifying or adding to the protocol but make communications easy and efficient given the built-in APIs.

$\mathbf{L_L}$ Implementations. Malware authors often hide the use of well-known protocols and prevent an investigator's immediate understanding of the C&C communication routines. L_L implementations use raw-socket (non protocol specific) APIs (e.g., `send`) in conjunction with official protocol tokens (e.g., `NICK` for the IRC protocol).

Notably, all protocols have L_L implementations, but only some also have a H_L implementation. Although custom protocol implementations are feasible, their uniqueness supports signature development making them easier to filter with firewall rules. Thus, C3PO identifies H_L and L_L implementations, and could be easily extended to other protocols when deemed necessary for an investigation.

3.2.2 Protocol Database. Standard protocols are often used for: (1) file transfer, (2) data storage, and (3) message-based communication. However, their ubiquitous integration into benign software has prompted research into inherent vulnerabilities which has led to unauthorized server access [22]–[25]. Noticing a similar trend in malware, we select common over-permissioned protocols discovered in preliminary research, reports from industry experts [31], [32], and related work [33] for our study, as shown in Table 2.

Based on the protocols, we constructed a database of all protocol identifiers for C3PO to reference during protocol identification (subsubsection 3.2.3). To construct this database, we developed a web-crawler and targeted it to the respective protocol documentation [34]–[38] or manually extracted protocol details to populate the database. However, as other over-permissioned protocols become widely adopted by malware, they can be easily integrated by adding their identifiers to the protocol database. Based on the protocol implementations and the database as a reference, C3PO conducts protocol identification to pinpoint protocol use.

3.2.3 Protocol Identification. To establish the execution context for malware analysis, C3PO parses the memory images and extracts code pages enabling import address and export directory tables (IAT and EDT) reconstruction. For each memory image, C3PO identifies the code regions to construct a CFG starting at the point the memory image was taken to all reachable code. This results in one CFG per memory image, rooted at the instruction pointer from the memory image. C3PO then creates a Combined CFG (C2FG) by matching overlapping blocks in all CFGs, ensuring no duplication. It then traverses this C2FG to identify all function call sites and compares it against the reconstructed IAT and EDT for a matching API. Although a common challenge in static analysis is resolving indirect function calls, the initial dynamic execution to generate memory images populates concrete function pointers in memory before image extraction, which aids in indirect call resolution. A data dependence graph, built from the C2FG, also resolves additional indirect calls.

Table 2: Over-Permissioned Protocols

Category	Over-Permissioned Protocol	Implementation(s)
File Transfer	File Transfer Protocol (FTP/TFTP)	L_L, H_L
	Web Distributed Authoring & Versioning (WebDAV)	L_L, H_L
	BitTorrent/Micro Transport Protocol (μTP)	L_L, H_L
Data Storage	Mongo Database	L_L, H_L
	MySQL	L_L, H_L
	PostgreSQL	L_L, H_L
	Object DB Connectivity (ODBC)	L_L, H_L
Message-based Communication	Internet Relay Chat (IRC)	L_lL
	Message Queuing Telemetry Transport (MQTT)	L_L H_L

$\mathbf{H_L}$ Identification. To identify H_L implementations, C3PO traverses the C2FG and resolves call targets. If it encounters an API that is in the protocol database, C3PO stores the call site and the called API. From our example in section 2, C3PO detected `FTPPutFile` in Sanny, classifying it as a over-permissioned bot because it uses FTP.

$\mathbf{L_L}$ Identification. L_L implementations use raw-socket APIs with a protocol token. When C3PO traverses the C2FG and encounters a call to a raw-socket API, it extracts API arguments to deduce tokens (as described in subsection 3.3). If the token is in the protocol database, C3PO stores the call site and the called API/token combination.

C3PO identified 62,202 over-permissioned bots (≈30%) in 200k malware. After protocol identification, C3PO continues the analysis to identify information that can be used to spoof bot-to-C&C communication toward infiltration.

3.3 Infiltration Vector (IV) Identification

Infiltration vectors (IVs) are the credentials used by the bot to connect to the C&C server. To spoof bot-to-C&C communication, C3PO identifies IVs using a combination of *backward slicing* and *iterative selective symbolic execution*.

Backward Slicing. C3PO uses the previously identified APIs, call sites, and tokens to first locate the authentication APIs (e.g., `SQLConnect` for H_L or `send` and a protocol token for L_L). C3PO performs backward slicing (of the C2FG) from these API arguments to identify a path to them through the malware. A challenge faced during backward slicing is that API arguments only point to the first byte of the data buffer (e.g., `lpszPassword` for `InternetConnect`) resulting in an incomplete slice. To address this, C3PO generates target instructions by identifying all instructions that were last to write to all bytes of the data buffer.

Iterative Selective Symbolic Execution (iSSE). C3PO symbolically executes along each of the backward slices to the authentication API. Since C3PO is constrained by the slice, symbolic execution is *selective* precluding path explosion while maintaining accuracy.

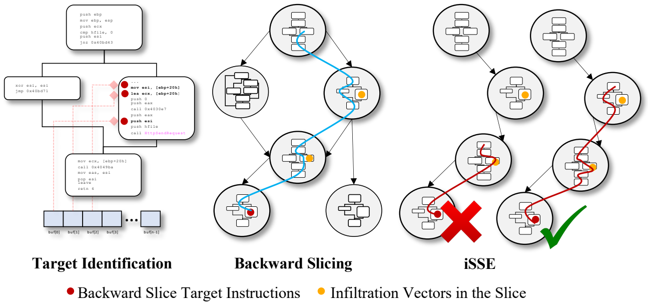

Target Identification
Backward Slicing
iSSE
● Backward Slice Target Instructions ● Infiltration Vectors in the Slice

Figure 3: C3PO's Infiltration Vector Identification of Sanny.

When iSSE reaches the authentication API, it halts to extract API arguments by dereferencing data buffer pointers. If the arguments are concrete, they are decoded to strings and iSSE analysis ends, as the IVs have been found. If they are symbolic, it means the API arguments were passed as parameters from the preceding (calling) function. C3PO, guided by the path, *incrementally* expands the exploration region by starting in the preceding function before re-initiating iSSE. This iterative process continues until the IVs are found. We discuss instances where concretization is not possible in section 8. Although execution can begin at the entry point, C3PO is more likely to encounter symbolic loops which can cause resource exhaustion if specific functions in the malware are computationally complex. Therefore, C3PO starts small (within the function), then incrementally expands to increase the likelihood of argument extraction. Loop handling is still necessary and C3PO employs a loop limiter to exit symbolic loops. However, loop avoidance is still preferred.

Figure 3 illustrates C3PO's IV Identification steps for the Sanny malware. C3PO performs backward slicing from the authentication API `InternetConnect`. For each of the authentication API arguments (e.g., `lpszServerName`, `nServerPort`, `lpszUserName`, `lpszPassword`, etc.), C3PO calculates the memory addresses for all bytes of the data buffer using a shadow memory that was populated during data dependency graph generation, a prerequisite for backward slicing. C3PO finds each instruction that was the last to write to each byte of the data buffer (*Target Identification* in Figure 3). Using these target instructions, C3PO conducts a backward slice to identify all influencing operations of the data buffer (the blue line through four of Sanny's functions in Figure 3). C3PO now traverses each slice using iSSE (*iSSE* in Figure 3) to extract IVs for all arguments. For Sanny, C3PO extracted the server hostname, username, and password (e.g., Table 1, Row 2) after covering only 3 of the 4 functions in the backward slice (the red iSSE line in Figure 3). Thus, C3PO can spoof bot-to-C&C communication and masquerade as a trusted bot.

3.4 C&C Monitoring Capabilities

Bots execute capabilities on the infected systems, some of which can be leveraged to provide covert monitoring. These C&C monitoring capabilities either (1) exfiltrate victim data or (2) allow bot orchestrators to execute arbitrary commands. These capabilities are valuable because the former alerts the authorities about the types and format of data stored on the C&C server, and the latter triggers commands on peer systems for botnet disruption upon infiltration.

To identify these capabilities, C3PO constructs a backward slice from all data exfiltration and code reflection targets in the malware. It then performs *API-to-capability mapping* to derive the C&C monitoring capabilities.

Backward Slicing. C3PO uses the previously identified APIs and call sites to locate data exfiltration (e.g., `HttpSendRequest`) and code reflection (e.g., `ShellExecute`) APIs. With each of these APIs as data sinks, C3PO performs backward slicing. For data exfiltration APIs, it backward slices from the API argument corresponding to the data exfiltration buffer (e.g., `lpOptional` for `HttpSendRequest`). For code reflection APIs, it backward slices from the operation arguments that reveal the C&C command triggers (e.g., `lpOPeration` for `ShellExecute`).

API-to-Capability Mapping. C3PO locates all API calls along each of the backward slices, similar to the technique used in subsubsection 3.2.3. This gives C3PO API sequences that influence the contents of the data exfiltration buffer or operation argument. These sequences of APIs are then compared against the capability models to identify the C&C monitoring capabilities. The capability models are derived by manually reverse engineering known malware and by using the insights from industry reports [32], [39]. In our study, we considered 6 categories of 16 C&C monitoring capabilities, as shown in Table 3.

To illustrate, C3PO identifies the victim profiling capability in the Sanny malware (section 2). C3PO performs backward slicing from the data sink `HttpSendRequest`. It calculates the memory addresses for all bytes of the sink buffer by referencing the shadow memory that was populated during data dependency graph generation (subsection 3.3). C3PO then finds each instruction that was the last to write to each byte of the buffer. Using these target instructions, C3PO conducts a backward slice to identify all influencing operations of the sink buffer. It identifies `GetUserDefaultLCID` and `GetLocaleInfoW` APIs leading up to `HttpSendRequest` API. This API sequence conforms with the capability model for Victim Locale Information, and hence the Sanny malware is classified as having a Victim Profiling Capability.

Note that this capability can be used for covert monitoring because it describes the type of data and format stored on the C&C server which results in immediate victim identification. It also reveals the scope of infection and potential damages incurred (victim

Table 3: C&C Monitoring Capabilities

Category		C&C Monitoring Capabilities
Browser Password Stealing	(1)	Mozilla Stealer Chrome Stealer Internet Explorer Stealer
Service Password Stealing	(2)	WiFi Stealer Kerberos Stealer Windows System Stealer
Victim Profiling	(3)	Registry-stored System Details Live System Operating State System OS Details Victim Locale Information
Spying, Live Monitoring	(4)	Keylogger Screen Capture Audio Capture
File Exfiltration	(5)	High-level Protocols Raw Socket Transfer
Code Reflection	(6)	Code Reflection

credentials provide access to sensitive accounts) providing legally admissible evidence to confirm computer fraud and abuse.

To identify code reflection, the same process holds. However, instead of identifying all APIs along the backward slice, C3PO locates the closest API to the sink that reads incoming information (e.g., `recv`). Once found, C3PO extracts the argument from the buffer to reveal the C&C command that triggered code reflection. This allows the authorities with C&C access to issue the commands to peers in the botnet to trigger arbitrary code execution. This capability goes beyond C&C server monitoring, and instead supports botnet disruption and takedown.

4 VALIDATING OUR TECHNIQUES

C3PO is implemented in C++ and Python, totaling 11k lines of code leveraging Detours [40] for memory image extraction and angr [41] to support binary analysis with specific applications to protocol identification, backward slicing, and iSSE. We also used the recently released AVClass2 [42], the current state-of-the-art in malware labeling tools, whose predecessor, AVClass [43], has long been relied upon in top-tier research [44]–[47].

Before deploying C3PO on the full data set, we validate its accuracy in identifying protocols and leverageable malware capabilities which enable covert and targeted C&C server monitoring. We leave the efficacy of infiltration vector analysis for our case studies (section 6) which demonstrate our ability to covertly infiltrate C&C servers. We evaluated C3PO using a ground truth dataset of 35 manually reverse engineered Windows malware from 13 different families, covering all protocols in Table 2.

4.1 Protocol Identification Evaluation

Table 4 presents C3PO's protocol identification evaluation. Columns 1-2 list the malware families (categorized by protocols found in each) and the number of malware variants (Var). Columns *Low-level* and *High-level Identifiers* present the ground truth (GT) findings, C3PO's analysis results of protocol identifiers found, and the true positive (TP), false positive (FP), and false negative (FN) metrics for each, respectively. C3PO correctly (TP) identified 290 (121 L_L +169 H_L) protocol identifiers. Our GT analysis confirmed 304 (135 L_L + 169 H_L) of them, revealing 13 FPs, 14 FNs, and an overall accuracy of over 94%.

We then dug into the detection of protocols among all variants. As an example, we identified 4 of the 13 malware families use FTP employing both L_L and H_L identifiers. C3PO's analysis of the Softcnapp, Ragebot, Blackhole, and Rbot malware reported no FTP FPs and FNs.

Table 4: Validating Protocol Identification. Var represents malware variants. GT represents the ground truth compared with C3PO's results to identify the TP, FP, and FN metrics.

Malware (by protocols)	Var	Low-level Identifiers					High-level Identifiers				
		GT	C3PO	TP	FP	FN	GT	C3PO	TP	FP	FN
FTP/TFTP											
Softcnapp	5	0	0	0	0	0	15	15	15	0	0
Ragebot	2	2	2	2	0	0	0	0	0	0	0
Blackhole	3	3	3	3	0	0	0	0	0	0	0
Rbot	2	2	2	2	0	0	0	0	0	0	0
Subtotal	12	7	7	7	0	0	15	15	15	0	0
WebDAV											
Equationdrug	2	54	42	42	0	12	0	0	0	0	0
Subtotal	3	54	42	42	0	12	0	0	0	0	0
BitTorrent/μTP											
Sathurbot	2	18	24	18	6	0	0	0	0	0	0
Icloader	1	0	0	0	0	0	21	21	21	0	0
Subtotal	3	18	24	18	6	0	21	21	21	0	0
MySQL											
Delf	4	0	0	0	0	0	24	24	24	0	0
Subtotal	4	0	0	0	0	0	24	24	24	0	0
MongoDB											
Cstealer	1	2	4	2	2	0	5	5	5	0	0
Subtotal	1	2	4	2	2	0	5	5	5	0	0
ODBC											
Zbot	4	0	0	0	0	0	60	60	60	0	0
Subtotal	4	0	0	0	0	0	60	60	60	0	0
PostgreSQL											
Alma	2	0	0	0	0	0	5	5	5	0	0
Subtotal	2	0	0	0	0	0	5	5	5	0	0
IRC											
Softcnapp	5	15	15	15	0	0	0	0	0	0	0
Ragebot	1	6	8	6	2	0	0	0	0	0	0
Rbot	2	9	7	7	0	2	0	0	0	0	0
Slackbot	4	12	12	12	0	0	0	0	0	0	0
Delf	4	12	15	12	3	0	0	0	0	0	0
Subtotal	16	54	57	52	5	2	0	0	0	0	0
MQTT											
Expiro	3	0	0	0	0	0	39	39	39	0	0
Subtotal	3	0	0	0	0	0	39	39	39	0	0
Total	35	135	134	121	13	14	169	169	169	0	0

Upon close inspection, we found that FPs occur when C3PO incorrectly identifies the use of a token (protocol command or keyword). C3PO reported 2 extra IRC tokens in Ragebot due to custom C&C commands which also used the PASS keyword (also an IRC command). Similarly, C3PO reported FPs in L_L implementations of MongoDB (2 false tokens), IRC (5 false tokens), and BitTorrent/uTP (6 false tokens) due to tokens appearing as substrings in other C&C communication. Although adding missed tokens to the protocol database reduces FNs, this is a case-by-case basis. Also, there is a tradeoff between FPs and FNs - allowing and ignoring substrings would increase FPs and FNs, respectively (e.g., some IRC bots use multiple tokens in one message, while in general, FTP bots do not).

Of the 135 manually-verified L_L identifiers, C3PO produced 14 FNs because of undocumented tokens resulting in 121 of 135 TPs. The only FNs occurred during the IRC and WebDAV protocols identification via their L_L implementation in the Rbot and Equationdrug samples, respectively. Whenever we encountered undocumented tokens, we subsequently added them to the protocol database. However, we retained our accuracy metrics results pre-modification as it represents a more accurate depiction of C3PO's protocol identification capability given the possibility of future undocumented tokens. Overall, C3PO was 94% accurate in identifying protocols making it robust enough to be applied to the large-scale study.

4.2 C&C Monitoring Capabilities Evaluation

Table 5 presents our evaluation of C3PO's ability to identify C&C monitoring capabilities. Columns 3-8 present the capabilities we consider in our study, but C3PO can be extended to support other capabilities. Their sub-columns are divided into two categories: GT and C3PO represents the number of ground truth capabilities identified and automatically identified, respectively. We found that C3PO correctly identified 100 (TP) C&C monitoring capabilities. Our GT analysis confirmed 104 capabilities, revealing 7 FPs, 4 FNs, and an overall accuracy of over 94%.

Table 5: Validating C&C Monitoring Capabilities Identification. Var represents malware variants. GT and C3PO represent the number of manually verified and automated capability identifiers per category, respectively.

Malware	Var	Browser Password Stealer		Service Password Stealer		Victim Profiling		File Exfiltration		Spying, Live Monitoring		Code Reflection		Accuracy Metrics		
		GT	C3PO	GT	C3PO	GT	C3PO	GT	C3PO	GT	C3PO	GT	C3PO	TP	FP	FN
Softcnapp	5	5	5	0	0	5	5	5	5	5	5	0	0	20	0	0
Cstealer	1	1	1	0	0	1	0	0	0	0	0	0	0	1	0	1
Ragebot	2	0	0	0	0	2	2	2	2	0	2	2	2	6	2	0
Expiro	3	0	0	0	0	3	3	3	3	3	3	0	0	9	0	0
Sathurbot	2	2	0	0	0	2	2	2	2	2	2	0	0	6	0	2
Icloader	1	0	0	0	0	1	1	1	1	1	1	0	0	3	0	0
Alma	2	0	0	0	0	2	2	2	2	2	2	0	0	6	0	0
Zbot	4	0	0	0	0	0	1	0	0	4	4	0	0	4	1	0
Rbot	2	1	1	0	0	2	2	2	2	2	2	2	2	9	0	0
Slackbot	4	0	0	0	0	4	4	4	4	0	0	0	0	8	0	0
Delf	4	0	0	0	0	4	4	0	4	4	4	0	0	8	4	0
Blackhole	3	2	1	2	2	3	3	3	3	3	3	0	0	12	0	1
Equationdrug	2	0	0	0	0	2	2	2	2	2	2	2	2	8	0	0
Total	35	11	8	2	2	31	31	26	30	28	30	6	6	100	7	4

Table 5 shows that Victim Profiling ranks highest among the capabilities, accounting for 29% (31 of the 104) of the capabilities. Next are Live Monitoring and File Exfiltration, with 28 (27%) and 26 (25%) capabilities, respectively. Toward covert monitoring, this shows that the authorities can expect to locate victim information on the C&C server including system information, personal files, and legally admissible evidence of spying.

Among 35 variants, 3 of them (Ragebot, Zbot, and Delf) had 7 FPs from the Victim Profiling, File Exfiltration, and Live Monitoring identification. Next, 3 of the FNs occurred in the Browser Password Stealer while 1 occurred in the Victim Profiling distributed among Cstealer, Sathurbot, and Blackhole. Further investigation revealed that both the FPs and the FNs are attributed to issues experienced using angr either due to unresolved symbolic constraints during CFG generation or temporary variable reuse causing spurious dependencies in the backward slice. However, our investigation confirmed these are rare occurrences. Given the low number of FPs and FNs, and over 94% accuracy, C3PO provides the means to effectively identify C&C monitoring capabilities.

5 LARGE-SCALE DEPLOYMENT

We deployed C3PO to measure over-permissioned protocols and C&C monitoring capabilities. We demonstrate that our automated measurement pipeline provides a scalable means for over-permissioned bot analysis.

5.1 Post Deployment Dataset Highlights

Deploying C3PO on our dataset exposed a growing trend of over-permissioned protocol use in malware. C3PO revealed that 62,202 (over 30%) of malware use one or more over-permissioned protocols. Figure 4 illustrates the adoption of over-permissioned protocols per bot from April 2006 to June 2020. We found that over-permissioned protocol use peaked in years 2015-2019, which also accounted for 80% of all over-permissioned protocols C3PO identified in our study. Interestingly, Figure 4 shows that not only has the use of over-permissioned protocols increased, but also the number of protocols used per malware. While a single bot using multiple over-permissioned protocols was uncommon, this practice is more prevalent now than ever before with over 4,000 bots using multiple protocols. In fact, since 2019 alone, C3PO found over 1,500 malware that used more than one over-permissioned protocol.

C3PO found the remainder of the malware (i.e., 70%) in our dataset used only HTTP-based communication for command and control. The prevalence of HTTP-based communication in our dataset is inline with observations by Peredisci et. al. [48], who reported that 75% of malware exhibit network activity via HTTP-based communication. This prevalence led many prior works [49]–[51] to target HTTP-based malware exclusively. HTTP-based malware send and receive data in HTTP packets using non-standard message protocols. Unlike the protocols considered in this paper, these HTTP-based messages do not readily provide authorities access to the C&C server. As such, these HTTP-only malware do not function as over-permissioned bots.

5.2 Over-Permissioned Bot Landscape

Table 6 presents interesting insights into over-permissioned protocol use. Columns 1 shows the protocols we studied; Column 2, the number of protocol uses; and Columns 3-4, the total number of H_L and L_L identifiers found. Columns 5-6 show their distribution with temporal protocol changes of each sample displayed in Column 8. The total number of malware families observed using specific protocols, as well as the first and last time the malware was seen in 2006-2020 are presented in the remaining columns.

C3PO detected 65,739 uses of over-permissioned protocols. Confirming our hypothesis that protocol efficiency supports continued prevalence, FTP is predominant among all protocols occurring 53,687 accounting for 81% of protocols identified (Column 2, Row 1). Besides, FTP has been consistently used across 88% of the 8,512 malware families over 15 years. This confirms our suspicion that malware authors either do not realize the inherent vulnerabilities from using over-permissioned protocols or simply do not expect them to be used as IVs. We expect the latter to be the case given the

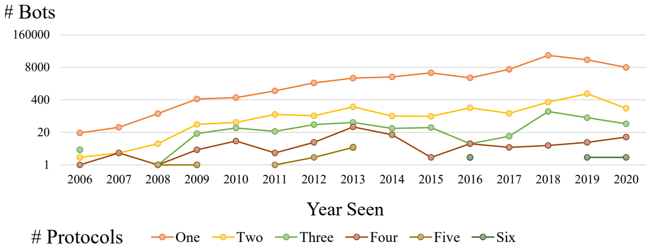
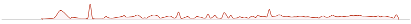
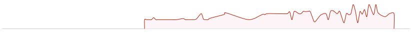
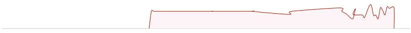
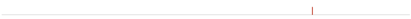
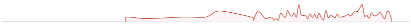
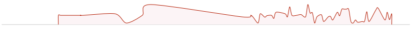
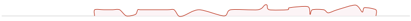
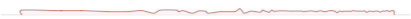


Figure 4: #Over-Permissioned Protocols Per Bot Since 2006.

Table 6: Distribution of Over-Permissioned Bots Identified During the Large-Scale Study.

Over-Permissioned Protocol	#Used	#Protocol ID		Identifiers/Protocol			Temporal Changes 2006 - 2020	#Families	First Seen	Last Seen
		H_L	L_L	Min	Avg	Max				
FTP/TFTP	55,494	32,531	22,963	2	4	9		8,163	2007-07	2020-06
BitTorrent/μTP	953	892	61	1	7	21		56	2011-04	2020-06
WebDAV	2,963	2846	117	8	16	21		135	2012-07	2020-06
MongoDB	1	0	5	5	5	5		1	2019-11	2019-11
ODBC	670	670	0	1	3	12		126	2010-08	2020-05
MySQL	262	262	53	1	5	11		53	2008-12	2020-05
PostgreSQL	117	117	25	2	4	7		25	2009-06	2019-10
MQTT	24	23	18	1	2	3		1	2014-04	2015-12
IRC	10,458	0	10,458	1	3	6		400	2006-10	2020-06
Total	65,739[1]	34,495	33,607	2	3	7		8,512	[1] 2006-10	2020-06

1: This is not the sum of this column, but the total number of protocols uses or malware families (see column 9) detected.

known FTP insecurities. Thus, bot orchestrators are unknowingly leaving the front door wide open, a trend our study sheds light on.

From Table 6, Column 8 illustrates the number of protocol identifiers used in malware since 2006. The number of identifiers per sample has generally fluctuated, except for the MQTT and IRC protocols. Similar to the findings in section 4, IRC implementations generally use 3 tokens to communicate with the C&C server as illustrated in the average identifiers per use (Column 6). The total distribution of protocol identifiers found (min=2, avg=3, max=7) indicates that many protocols use multiple APIs or tokens giving authorities multiple monitoring vantage points. Notably, the MongoDB protocol was used in the Cstealer malware then spontaneously disappeared from use resulting in a single spike in Column 8. This is likely due to its rapid discovery and public reporting [52], which immediately revealed a weakness resulting from the use of the over-permissioned MongoDB protocol. We expect to see a resurgence of the MongoDB protocol as some malware authors continue to prefer efficiency and ease of use over security. Furthermore, the temporal changes in levels of protocol implementations (i.e., identifiers used) gives us insights into the type of protocol capabilities enabled; e.g., if C3PO identifies `FTPGetFile` or `FTPPutFile` in the malware, the C&C supports at least FTP read/write. However, as our study shows, it is safe to assume full protocol implementation since malware operators adopt over-permissioned protocols for ease of use and scalability.

Table 6, Row 8 (*MQTT*) shows a similar trend as the MongoDB protocol. However, the use of the MQTT protocol is observed over a longer period. The Expiro malware is the only malware family detected using the MQTT protocol and disappeared from detection in 2015. Interestingly, industry experts observed and reported on a resurgence of Expiro in 2017 [53] adding clarity to our observation, given that we did not detect it between 2015 and 2020. Industry experts also reported improvements in Expiro, which we believe correlates with its lack of presence in recent years, likely stemming from its exclusion of the MQTT protocol.

From Table 6, we also observe that L_L identifiers are the majority with 33,636 detected versus 33,486 H_L identifiers, although, only by a slight margin. However, the majority of L_L implementations resulted from FTP and IRC protocols. As discussed, the IRC protocol has no H_L implementation. Although many IRC bots are no longer active because of the centralized architecture, which has proven limitations, new IRC malware have been detected in 2020. We now turn our attention to the 8,512 malware families identified. This is important as it illustrates the wide-scale applicability of C3PO across multiple malware families and variants.

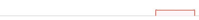
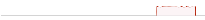
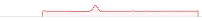
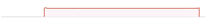
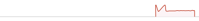
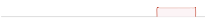
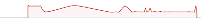
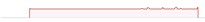
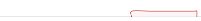
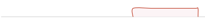
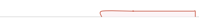
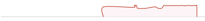
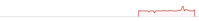
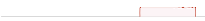
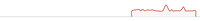
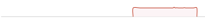
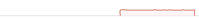
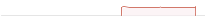
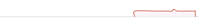
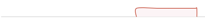
5.3 C&C Monitoring Capabilities at Scale

C&C monitoring capabilities guide covert monitoring after C&C server infiltration. As shown in Table 7, C3PO identified 443,905 C&C monitoring capabilities in 62,202 over-permissioned bots revealing an average of 7 capabilities per bot. Notably, Victim Profiling

Table 7: C3PO Identification of C&C Monitoring Capabilities Mapped to Over-Permissioned Protocols.

C&C Monitoring Capabilities	FTP/TFTP	BitTorrent/μTP	WebDAV	MySQL	MongoDB	ODBC	PostgreSQL	IRC	MQTT	Total
Service Password Stealing										
WiFi Stealer	1	0	0	0	0	0	0	0	0	1
Kerberos Stealer	3	0	0	0	0	0	0	1	0	4
Windows Sys. Stealer	7	0	0	0	0	1	0	1	0	9
Subtotal	11	0	0	0	0	1	0	2	0	**14**
Code Reflection										
Code Reflection	202	0	140	1	0	6	2	77	0	428
Subtotal	202	0	140	1	0	6	2	77	0	**428**
Browser Password Stealing										
Internet Exp. Stealer	1,611	0	6	0	0	1	0	11	0	1,629
Chrome Stealer	812	0	0	1	1	5	0	5	0	824
Mozilla Stealer	2,103	0	496	3	0	2	0	24	0	2,628
Subtotal	4,526	0	502	4	1	8	0	40	0	**5,081**
File Exfiltration										
High-level Protocols	6,891	6	1188	0	0	65	2	60	0	8,212
Raw Socket Transfer	52,223	110	1168	210	0	510	110	6,374	24	60,729
Subtotal	59,214	116	2,356	210	0	575	112	6,434	24	**69,041**
Spying/Live Monitoring										
Audio Capture	10,788	1	62	30	0	15	0	86	0	10,982
Keylogger	39,551	84	2,296	185	0	348	113	4,256	24	46,857
Screen Capture	52,458	109	2,524	220	0	537	110	6,469	24	62,451
Subtotal	102,797	194	4,882	435	0	900	223	10,811	48	**120,290**
Victim Profiling										
Victim Locale Info.	51,924	99	2,462	217	0	512	98	6,441	24	61,777
System OS Details	52,354	110	2,518	222	1	530	110	6,469	24	62,338
Registry-stored Info.	52,354	110	2,512	225	1	528	98	6,471	24	62,323
Live OS State	52,564	110	2,510	226	1	534	117	6,477	24	62,563
Subtotal	209,196	420	10,002	890	3	2,104	423	25,858	92	**249,051**
Total	375,946	730	17,882	1,540	4	3594	760	43,222	164	**443,905**

Table 8: Evolution of the Top 10 Families of Over-Permissioned Bots Detected in our Dataset.

Malware Family	#Over-Permissioned Bots	Over-Permissioned Protocols	Protocol Use			Evolution of Protocol Use 2006 - 2020	C&C Monitoring Capabilities[1]	# C&C Monitoring Capabilities			Evolution of C&C Monitoring Capabilities 2006 - 2020
			Min	Avg	Max			Min	Avg	Max	
Dinwod	9,713	FTP	1	1	1		BPS, VP, FE, LM	3	3	4	
Autoit	5,763	FTP, IRC	1	1	2		BPS, VP, FE, LM	3	3	4	
Softcnapp	4,382	FTP, IRC, ODBC	1	1	2		BPS, VP, FE, LM	3	3	4	
Delf	4,331	FTP, IRC, MySQL, TFTP, ODBC	1	1	3		BPS, VP, FE, LM, CE	3	3	5	
Wabot	2,388	IRC	1	1	1		VP, FE, LM	3	3	3	
Fareit	1,479	FTP, IRC, ODBC	1	1	2		BPS, VP, FE, LM	2	3	4	
Sivis	1,167	FTP, IRC, ODBC, MySQL, Bittorrent	1	1	3		BPS, VP, FE, LM, CE	3	3	5	
Lamer	1,019	FTP, IRC, ODBC	1	1	2		BPS, VP, FE, LM	3	3	4	
Virut	998	FTP, IRC, ODBC, MySQL	1	1	3		BPS, VP, FE, LM	3	3	4	
Snojan	897	FTP, IRC	1	1	2		BPS, VP, FE, LM	3	3	4	

1: BPS = Broswer Pwd Stealing, VP = Victim ID, FE = File Exfiltration, LM = Live Monitoring, CE = Code Execution (see Table 7)

and Live Monitoring account for the majority of capabilities, at 56% and 27% with 249,051 and 120,290 identifications, respectively. It follows that the majority of over-permissioned bots use techniques that can be applied more broadly to information stealing, which Victim Profiling and Live Monitoring provide. File Exfiltration is the next commonly used (i.e., 69,041 capabilities), 15% of all capabilities identified.

Of all 16 capabilities, 375,946 or 88% and 43,222 or 10% of them occur in FTP and IRC protocols, respectively. C3PO's ability to extract IVs for these protocols alone allows the authorities to covertly monitor over 85% of over-permissioned bots in our dataset, which we believe is representative of the larger malware landscape. Another observation is that although password stealers capture sensitive victim information, their tactics are tailored for a limited number of applications or services, reducing the scale of their impact explaining the low numbers in the password stealing categories at 5,095 or 0.01%. Overall, C3PO reveals the composition and content of C&C servers through C&C monitoring capabilities identification allowing it to provide targeted monitoring post infiltration.

5.4 Ranking Over-Permissioned Bot Families

Table 8 presents the protocols and the C&C monitoring capabilities identified in the top 10 malware families of our study. The Dinwod malware family ranks the highest with 9,713 over-permissioned bots. Dinwod only uses FTP and has remained consistent, even in the analysis of capabilities that include Browser Password Stealing (BPS), Victim Profiling (VP), File Exfiltration (FE), and Live Monitoring (LM), averaging 3 capabilities per sample.

Another observation from Table 8: FTP is used in 9 of the top 10 malware families as expected since it is the most prominent over-permissioned protocol in our dataset. About half of the families in Table 8 maintain a generally consistent number of protocols used, with the exception of Delf, Sivis, and Virut, with 3 maximum protocols used each, attributing to the spikes in Column *Evolution of Protocol Use.*

Lastly, the top capabilities — File Exfiltration, Live Monitoring, and Victim Profiling — appear in all 10 families. However, we did not expect Browser Password Stealing in 9 families since it accounts for 1.26% of all C&C monitoring capabilities (ref. Table 7). From this study, we can infer that while the majority of over-permissioned bots can be considered Information Stealers, many of the top malware families are Password Stealers.

5.5 Packed Malware

C3PO uses a hybrid approach to analyze packed malware (subsection 3.1). We present the most common packers encountered in our study in Table 9 and use the packer taxonomy proposed by Ugarte-Pedrero et al. [29]. Column 1 lists these packer types, and Column 3 shows the number of packed malware that use the packers[3] presented in Column 2. The packer types range from Type-I to Type-VI, which also represent their order of complexity [29].

For clarity, Type-I packers are the easiest to unpack only using a single unpacking routine before transferring control to the malware payload. Type-II packers use multiple layers of packing, only transferring control after the last layer is complete. Like Type-II packers, Type-III is multi-layered but does not unpack in a top-down manner and instead uses complex layer organization. While Type-IV packing can use single or multi-layers, the unpacking routine is interwoven with the malware payload switching control back and forth. Type-V and Type-VI are quite similar to Type-IV, except more and more malware payload code is interwoven increasing the complexity of the unpacking routine.

From Table 9, we see that C3PO can unpack and analyze samples packed with Armadillo (Row 3), i.e., it can handle the most complex category of packers. Of the 62,202 over-permissioned bots, C3PO unpacked 10,237 malware. The remainder of the samples were not packed. In our dataset, C3PO did not encounter any malware packed with Type-II, Type-IV, and Type V packers. But given its ability to handle Type-VI packers, we believe that C3PO is robust enough to enable a large-scale study of modern malware.

6 C3PO APPLIED

We present two over-permissioned bot case studies to illustrate the efficacy of our techniques. We focus on the cases that use the most prevalent FTP over-permissioned protocol. We redact the C&C server information because the servers are still active as of this

[3] Packers are identified using PackerID [54]

Table 9: Packers Encountered in our Dataset.

Packer Type [29]	Packer	# Malware
Type-I	UPX	9,372
	BobSoft Mini Delphi	86
Type-III	ASPack	48
	ASProtect	22
Type-VI	Armadillo	709
	Total	10,237

writing, but present the monitoring outputs we extracted adhering to ethical practices, which we describe next.

Ethical Considerations. We follow the precedence established in previous works [16], [25], [55], [56] while exposing the weaknesses that make C&C servers vulnerable to infiltration. Besides, Burnstein [57] provides legal and ethical conduct for cybersecurity research, arguing that injecting traffic into C&C servers can be considered consent when using the communication channel the bot orchestrators provided to the enslaved systems. Similarly, we use the bot-to-C&C channel and the authentication details provided to us through the malware. Moreover, after verifying access permissions we (1) only retrieve the metadata (e.g. file quantity, table schema, etc.) of the service being investigated (FTP, MongoDB, etc.) and (2) perform no write operations. We emphasize that we *do not* exploit, disrupt, or attempt takedown of C&C servers, avoiding any claim of *tortious interference* as described in Mouton vs. VC3 [58].

6.1 Case Study 1: Steam

The Steam malware is a Remote Access Trojan (RAT) [59] first discovered in 2016 and persists today. C3PO identified FTP in Steam and extracted IVs and C&C monitoring capabilities (Table 10, Rows 1-3). Leveraging the IVs, C3PO covertly monitored the Steam C&C server resulting in the identification of approximately 50 MB of data (522 files in 5 directories). Of the files, 27% of them have "game"-related names like *matchroom* and *tournament* confirming that our sample is indeed tailored for the Steam platform.

C3PO identified Victim Profiling and File Exfiltration, so we expected to find a large number of files on the C&C server containing stolen victim information. Since this malware is relatively new, it is not surprising that we only found less than 20% of these files, but we expect it to grow as the malware spreads. However, C3PO identified two *data* files whose filenames began with *ssfn*. The authorization files for the Steam online gaming platform also begin with *ssfn*. These files are likely encrypted since their entropy values are 7.90 and 7.92 (on a scale of 0.0 - 8.0), respectively. These authorization files could either be stolen files for authentication to the Steam platform (as suggested by C3PO's File Exfiltration C&C monitoring capabilities), or they belong to the bot orchestrator. For the latter case, an incident responder can use these files to pursue attribution since it provides access to the bot orchestrator's account.

C3PO also revealed filenames that piqued our interest. Specifically, several files are named in the Russian language, the C&C server's likely country of origin. Furthermore, C3PO discovered a JavaScript file containing code that looked for cross-site scripting (XSS) vulnerabilities. This suggests further malicious intent to perpetrate additional cybercrimes. Our findings are further confirmed by a Steam analysis report [59] validating C3PO's effectiveness in covert monitoring and extracting valuable insights.

Table 10: C3PO's Steam Malware Analysis Results.

Protocol	FTP

Infiltration Vectors	**Username**: j91{***}
	Password: Dom{***}
	Server: {***}.beget.tech
	Port: 21

C&C Monitoring Capabilities	Victim Profiling and File Exfiltration
Covert Monitoring Outputs	(1) Country of origin, (2) Steam Authentication Files (3) XSS injection script

6.2 Case Study 2: Detplock

The Detplock malware is another RAT first seen in 2016 and is still active today. This malware allows the bot orchestrator to execute commands on the infected machines. Table 11 summarizes C3PO's covert monitoring results by analyzing the DeptLock malware. C3PO extracted IVs such as the username, password, server address, and port, as shown in Table 11, Row 1. Based on the sever address suffix `.ko.cr`, the C&C server is likely located in South Korea. This C&C server responds to FTP queries, which we used to only catalog file metadata, enumerating directories, keeping count of the number of directories and files as well as file extensions and file sizes. Overall, we identified approximately 640MB of data including over 2,500 files across 47 directories. Of the 31 file extensions found, the most common extensions were PNG (44%), HTML (34%), TXT (8%), and EXE (6%).

C3PO also identified Victim Profiling, Live Monitoring, and File Exfiltration capabilities (Table 11, Row 2). From covert monitoring, C3PO discovered many PNG files, which was expected since its analysis showed that Detplock performed Live Monitoring by taking PNG screenshots. This confirms the effectiveness of C3PO's C&C monitoring capabilities towards covert monitoring. C3PO also located the `userData` directory which is used to store victim information, corresponding to the Victim Profiling malware capability (Table 11, Row 3). While this directory was empty upon infiltration, convert monitoring allows us to regularly monitor for added infected systems to understand the scope of infection and enable peer disclosure.

Lastly, C3PO found malicious files on the C&C server's `download` directory, confirming that Detplock spreads other payloads. Specifically, 7 of the 158 Windows EXE and 2 BIN files contained suspicious metadata. Their signatures revealed ASPack v2.12 packing and their hash search on VirusTotal [60] confirmed maliciousness. Although the C&C monitoring capabilities did not infer additional payloads on the C&C server, our ability to covertly infiltrate and leverage over-permissioned FTP functionality to quickly query the server revealed at least additional 9 malicious files.

Table 11: C3PO's Detplock Malware Analysis Results.

Protocol	FTP

Infiltration Vectors	**Username**: eg{***}
	Password: vrg{***}
	Server: {***}.co.kr
	Port: 21

C&C Monitoring Capabilities	Victim Profiling, Live Monitoring, and File Exfiltration
Covert Monitoring Outputs	(1) PNG files confirming the live monitoring capability (2) 9 malicious executables and binaries

7 RELATED WORK

C&C Infiltration and Monitoring. C&C monitoring research has focused on P2P botnets [11], [61]–[64]. These works are transparent to botnet architectures and inspired our monitoring techniques. Dispatcher [65] analyzes botnet protocols to enable infiltration, but it does not consider the plethora of commodity protocols used in malware. Conversely, C3PO detects and exploits weaknesses in several widely used protocols, with room for growth, and enables a scalable approach to covert C&C monitoring. Similar works have also achieved infiltration of botnets that led to their disruption [66] validating the necessity of not only monitoring, but the immediate benefits of covert infiltration. Although successful, this attempt targeted weaknesses specific to the MegaD botnet and also used "Google hacking" for information gathering. Lastly, domain seizure approaches provide another relevant avenue to monitoring by taking over the botnet [1], [3]–[6], [17] but these approaches required detailed reverse engineering efforts to understand domain generating algorithms. Instead, C3PO analyzes the malware alone extracting hidden information to enable infiltration towards covert monitoring.

Protocol Identification. Several works infer protocol formats based on network traces [67]–[70] or after first understanding how the binary processes network messages [65], [71]–[73]. However, these approaches rely on capturing all protocol functionality, which can limit the overall effectiveness when full network traces are not captured. Conversely, protocol identification is only one of the multiple enabling techniques employed by C3PO towards our goal of providing authorities with leverage over C&C servers through covert monitoring. Furthermore, C3PO uses binary analysis to identify protocols based on their APIs, commands, or keywords that are invoked in a binary, which is more scalable when full network traces cannot be collected (e.g., with defensive malware). Different from prior works, solely identifying protocols is orthogonal to enabling authorities to take action against malicious campaigns, enabled by C3PO.

Selective Symbolic Execution. Symbolic Execution is used to find software bugs [74]–[76], generate test cases [77]–[79], and improve the execution of dynamic analysis [80]–[82]. Specifically, MalMax [82], X-Force[80], and J-Force [81] used forced execution for exploration. While MalMax uses backtracking to enable path exploration, J-Force mutates satisfiable branch predicates to explore un-visited paths. Similarly, Smartgen[83] proposed Selective Symbolic Execution for the Android framework using solved path constraints to guide execution. C3PO's iSSE traversable paths are not determined by derived inputs, but instead by the backward slice of the execution target, which constrains feasible paths reducing the exploration area to identify IVs.

Malware Capability Analysis. Several works use behavior analysis [84]–[86], behavior modeling [87]–[89], and network traffic observation [90], [91] to identify malware. Those that detect malware capabilities [84], [87], [88], [92] are either specific to the Android framework or use dynamic analysis to identify just enough capabilities for malware detection. On the contrary, C3PO introduces a scalable approach to identify 16 malware capabilities and offers the option of extending support for others using capability modeling.

8 DISCUSSION AND LIMITATIONS

Domain Generating Algorithms. DGA-based malware allows bot orchestrators to move from centralized architectures to more robust architectures using automatically generated pseudo-random C&C domain names [93]. This technique allows over-permissioned bot orchestrators to subvert persistent infiltrations through C3PO since the C&C domain names are dynamically generated. Other malware adopts a similar approach, using cloud-based services to retrieve C&C domain names [94]. These categories of malware pose significant challenges for C3PO. However, they are not insurmountable, as C3PO can be used to complement existing techniques to identify DGA future candidate domains, as demonstrated by Le et al. [6].

Subverting Dynamic Memory Image Extraction. C3PO's primary technique for memory image extraction is API hooking. As an automated pipeline, C3PO is limited in its ability to spoof specific environments for malware but could be combined with techniques such as forced execution to overcome this [80], [81]. However, sandboxes can also be used to augment C3PO in-lieu of memory image extraction. For example, S2E [75] enables symbolic execution within a sandbox to explore thousands of system paths. Toward unpacking, there are three evasion types to thwart API hooking: stolen code, child process, and process hollowing, often seen in the Themida, PEP, and Pespin packers [28]. Although C3PO can handle Type-I, III, and IV packers, it cannot analyze malware that uses virtualization packed techniques. These packers convert programs into bytecode increasing complexity and eludes C3PO memory image extraction. However, virtualization packers account for less than 2% of packed malware, while Type-I packers (e.g., UPX) account for over 55% [29].

Custom Low-level Protocol Implementations. Some malware prefer custom protocol implementations to make their analysis more difficult, but the uniqueness of custom protocols supports signature development increasing their chances of IDS detection. So, C3PO focuses on protocol implementations that adhere to official protocol specifications. However, since C3PO relies on official APIs and tokens, custom tokens evade C3PO's identification. Even if we consider well-known (but not official) tokens, since C3PO analyzes the client-side binary alone, it cannot match a custom keyword to a protocol without knowing how the server parses it. While malware authors can use official protocol commands to trick analysts in misidentifying the protocol used, we have not observed this practice during our large-scale study. In order to support the identification of over-permissioned custom protocols, the integration of tools such as Prospex [73] can be used to automatically reverse engineer custom protocols revealing identifiers that can be *exploited* for covert C&C server monitoring. Although extracting relevant information from the protocol, then adding them to C3PO's protocol database requires some effort upfront, maintenance is all that is required after allowing seamless integration into C3PO.

9 CONCLUSION

This paper presented C3PO, a measurement pipeline that studied the evolution of over-permissioned protocols in 200k malware spanning back 15 years and how they can be leveraged to provide covert C&C server monitoring. C3PO identified 62,202 over-permissioned bots across 8,512 families identifying infiltration vectors that allow C3PO to spoof bot-to-C&C communication. C3PO also identified 443,905 C&C monitoring capabilities which reveal the composition and contents of the C&C server to guide monitoring post infiltration. We deployed C3PO on two bots with live C&C servers validating its ability to identify over-permissioned protocols, infiltrate C&C servers, and leverage C&C monitoring capabilities to achieve covert monitoring. Furthermore, C3PO identified over 2500 files, some of which contain victim information, additional malicious payloads, exploitation scripts, and stolen credentials providing legally admissible evidence to engender attempts of botnet disruptions and takedowns. We offer C3PO to aid the authorities as they seek to contain and eradicate botnets through covert C&C server monitoring: https://cyfi.ece.gatech.edu/.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive comments and feedback. We thank our collaborators at Netskope for their support, insights, and suggestions throughout this research. This work was inspired by our collaboration with Sandia National Laboratories; we particularly thank Tim Loffredo and Shelley Leger for their insights and guidance. This work was supported, in part, by ONR under grants N00014-19-1-2179 and N00014-18-1-2662 and NSF under Awards 1755721 and 1916550. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of our sponsors or collaborators.

REFERENCES

[1] C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C. J. Dietrich, and H. Bos, "SoK: P2pwned-modeling and Evaluating the Resilience of Peer-to-peer Botnets," in *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2013, pp. 97–111.

[2] B. Krebs, *U.s. cyber command behind trickbot tricks*, [Accessed: 2020-08-22]. [Online]. Available: https://krebsonsecurity.com/2020/10/report-u-s-cyber-command-behind-trickbot-tricks/.

[3] Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee, "Beheading Hydras: Performing Effective Botnet Takedowns," in *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, Berlin, Germany, Oct. 2013, pp. 121–132.

[4] R. Wainwright and F. J. Cilluffo, *Responding to cybercrime at scale: Operation avalanche — a case study*. [Online]. Available: http://www.jstor.org/stable/resrep20752.

[5] *New action to disrupt world's largest online criminal network*, https://blogs.microsoft.com/on-the-issues/2020/03/10/necurs-botnet-cyber-crime-disrupt/, [Accessed: 2020-03-12].

[6] V. Le Pochat, S. Maroofi, T. Van Goethem, D. Preuveneers, A. Duda, W. Joosen, M. Korczyński, *et al.*, "A Practical Approach for Taking Down Avalanche Botnets Under Real-World Constraints," in *Proceedings of the 2020 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2020.

[7] *Avast and French police take over malware botnet and disinfect 850,000 computers*, https://www.zdnet.com/article/avast-and-french-police-take-over-malware-botnet-and-disinfect-850000-computers/, [Accessed: 2020-03-29].

[8] W. Sebastian and C. Rossow, "MALPITY: Automatic Identification and Exploitation of Tarpit Vulnerabilities in Malware," in *Proceedings of the 4th European Symposium on Security and Privacy (EuroS&P)*, Stockholm, Sweden, Jun. 2019, pp. 590–605.

[9] *Zeus-p2p monitoring and analysis*, [Accessed: 2020-12-10]. [Online]. Available: https://www.cert.pl/wp-content/uploads/2015/12/2013-06-p2p-rap_en.pdf.

[10] B. B. Kang, E. Chan-Tin, C. P. Lee, J. Tyra, H. J. Kang, C. Nunnery, Z. Wadler, G. Sinclair, N. Hopper, D. Dagon, *et al.*, "Towards Complete Node Enumeration in a Peer-to-peer Botnet," in *Proceedings of the 4th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Sydney, Australia, Mar. 2009, pp. 23–34.

[11] D. Andriesse, C. Rossow, and H. Bos, "Reliable Recon in Adversarial Peer-to-peer Botnets," in *Proceedings of the Internet Measurement Conference (IMC)*, Tokyo, Japan, Oct. 2015, pp. 129–140.

[12] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active Botnet Probing to Identify Obscure Command and Control Channels," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2009, pp. 241–253.

[13] C. Zuo, Q. Zhao, and Z. Lin, "Authscope: Towards Automatic Discovery of Vulnerable Authorizations in Online Services," in *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*, Dallas, TX, Oct. 2017, pp. 799–813.

[14] A. Nappa, Z. Xu, M. Z. Rafique, J. Caballero, and G. Gu, "Cyberprobe: Towards Internet-scale Active Detection of Malicious Servers," in *Proceedings of the 2014 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2014, pp. 1–15.

[15] Z. Xu, A. Nappa, R. Baykov, G. Yang, J. Caballero, and G. Gu, "Autoprobe: Towards Automatic Active Malicious Server Probing Using Dynamic Binary Analysis," in *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*, Scottsdale, AZ, Nov. 2014, pp. 179–190.

[16] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide Scanning and its Security Applications," in *Proceedings of the 22th USENIX Security Symposium (Security)*, Washington, DC, Aug. 2013, pp. 605–620.

[17] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your Botnet is my Botnet: Analysis of a Botnet Takeover," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, Chicago, Illinois, Nov. 2009, pp. 635–647.

[18] *Trickbot botnet survives takedown attempt, but microsoft sets new legal precedent*, [Accessed: 2020-12-10]. [Online]. Available: https://www.zdnet.com/article/trickbot-botnet-survives-takedown-attempt-but-microsoft-sets-new-legal-precedent/.

[19] *An update on disruption of trickbot*, [Accessed: 2020-12-10]. [Online]. Available: https://blogs.microsoft.com/on-the-issues/2020/10/20/trickbot-ransomware-disruption-update.

[20] *Kelihos/hlux botnet returns with new techniques*, [Accessed: 2020-12-10]. [Online]. Available: https://securelist.com/kelihoshlux-botnet-returns-with-new-techniques/32021/.

[21] I. Arghire, *Trickbot botnet survives takedown attempt*, [Accessed: 2020-12-10]. [Online]. Available: https://www.securityweek.com/trickbot-botnet-survives-takedown-attempt.

[22] S. Banescu and A. Pretschner, "A Tutorial on Software Obfuscation," vol. 108, Elsevier, 2018, pp. 283–353.

[23] *Carbanak APT: The Great Bank Robbery*, https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064518/Carbanak_APT_eng.pdf, [Accessed: 2020-04-16].

[24] A. Mandal, *Thick Client Application Security*, http://www.infosecwriters.com/Papers/AMandal_Thick_Client_Application_Security.pdf, [Accessed: 2020-04-18].

[25] O. Alrawi, C. Zuo, R. Duan, R. P. Kasturi, Z. Lin, and B. Saltaformaggio, "The Betrayal at Cloud City: An Empirical Analysis of Cloud-based Mobile Backends," in *Proceedings of the 28th USENIX Security Symposium (Security)*, Santa Clara, CA, Aug. 2019, pp. 551–566.

[26] *Command and Control Used in Sanny APT Attacks Shut Down*, [Accessed: 2021-01-09]. [Online]. Available: https://threatpost.com/command-and-control-used-sanny-apt-attacks-shut-down-032213/77658/.

[27] *Sanny Malware Updates Delivery Method*, [Accessed: 2021-01-09]. [Online]. Available: https://threatpost.com/sanny-malware-updates-delivery-method/130803/.

[28] B. Cheng, J. Ming, J. Fu, G. Peng, T. Chen, X. Zhang, and J.-Y. Marion, "Towards Paving the Way for Large-scale Windows Malware Analysis: Generic Binary Unpacking with Orders-of-magnitude Performance Boost," in *Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS)*, Toronto, ON, Canada, Oct. 2018, pp. 395–411.

[29] X. Ugarte-Pedrero, D. Balzarotti, I. Santos, and P. G. Bringas, "SoK: Deep Packer Inspection: A Longitudinal Study of the Complexity of Run-time Packers," in *Proceedings of the 36th IEEE Symposium on Security and Privacy (S&P)*, San Jose, CA, May 2015, pp. 659–673.

[30] O. Alrawi, M. Ike, M. Pruett, R. P. Kasturi, S. Barua, T. Hirani, B. Hill, and B. Saltaformaggio, "Forecasting Malware Capabilities From Cyber Attack Memory Images," in *Proceedings of the 30th USENIX Security Symposium (Security)*, Virtual Conference, Aug. 2021.

[31] Mandiant, *APT1: Exposing One of China's Cyber Espionage Units*, https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf, [Accessed: 2020-05-23].

[32] *Attack matrix for enterprise*, https://attack.mitre.org/, [Accessed: 2020-06-09].

[33] M. D. Brown and S. Pande, "CARVE: Practical Security-focused Software Debloating using Simple Feature Set Mappings," in *Proceedings of the 3rd ACM Workshop on Forming an Ecosystem Around Software Transformation (FEAST)*, London, United Kingdom, 2019, pp. 1–7.

[34] *Microsoft Documentation*, [Accessed: 2021-01-09]. [Online]. Available: https://docs.microsoft.com/en-us/.

[35] C. Kalt, "Internet relay chat: Client protocol," RFC Editor, RFC 2812, Apr. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2812.

[36] *MRFC 1350 - The TFTP Protocol*, [Accessed: 2021-01-09]. [Online]. Available: https://tools.ietf.org/html/rfc1350.

[37] *MySQL Documentation*, [Accessed: 2021-01-09]. [Online]. Available: https://dev.mysql.com/doc/.

[38] *MongoDB C Driver*, [Accessed: 2021-01-09]. [Online]. Available: https://docs.mongodb.com/drivers/c.

[39] *Malpedia: Free and Open Malware Reverse Engineering Resource offered by Fraunhofer FKIE*, https://malpedia.caad.fkie.fraunhofer.de, [Accessed: 2020-05-29].

[40] G. Hunt and D. Brubacher, "Detours: Binary Interception of Win32 Functions," in *Proceedings of the 3rd USENIX Windows NT Symposium*, Seattle, WA, Jul. 1999.

[41] Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel, and G. Vigna, "SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis," in *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P)*, San Jose, CA, May 2016, pp. 138–157.

[42] S. Sebastián and J. Caballero, "AVclass2: Massive Malware Tag Extraction from AV Labels," in *Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC)*, Virtual Conference, Dec. 2020, pp. 42–53.

[43] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A Tool for Massive Malware Labeling," in *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Evry, France, Sep. 2016, pp. 230–253.

[44] C. Lever, P. Kotzias, D. Balzarotti, J. Caballero, and M. Antonakakis, "A Lustrum of Malware Network Communication: Evolution and Insights," in *Proceedings of the 38th IEEE Symposium on Security and Privacy (S&P)*, San Jose, CA, May 2017, pp. 788–804.

[45] P. Kotzias, L. Bilge, and J. Caballero, "Measuring PUP Prevalence and PUP Distribution through Pay-Per-Install Services," in *Proceedings of the 25th USENIX Security Symposium (Security)*, Austin, TX, Aug. 2016, pp. 739–756.

[46] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. Alrwais, L. Sun, and Y. Liu, "Resident Evil: Understanding Residential IP Proxy as a Dark Service," in *Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2019, pp. 1185–1201.

[47] D. Kim, B. J. Kwon, K. Kozák, C. Gates, and T. Dumitraș, "The Broken Shield: Measuring Revocation Effectiveness in the Windows Code-Signing PKI," in *Proceedings of the 27th USENIX Security Symposium (Security)*, Baltimore, MD, Aug. 2018, pp. 851–868.

[48] R. Perdisci, W. Lee, and N. Feamster, "Behavioral Clustering of HTTP-based Malware and Signature Generation using Malicious Network Traces," in *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, Apr. 2010.

[49] R. F. M. Dollah, M. Faizal, F. Arif, M. Z. Mas'ud, and L. K. Xin, "Machine Learning for HTTP Botnet Detection Using Classifier Algorithms," vol. 10, Universiti Teknikal Malaysia Melaka, 2018, pp. 27–30.

[50] T. Nelms, R. Perdisci, and M. Ahamad, "Execscent: Mining for new C&C Domains in Live Networks with Adaptive Control Protocol Templates," in *Proceedings of the 22th USENIX Security Symposium (Security)*, Washington, DC, Aug. 2013.

[51] A. Oprea, Z. Li, R. Norris, and K. Bowers, "Made: Security Analytics for Enterprise Threat Detection," in *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC)*, 2018.

[52] *New Chrome Password Stealer Sends Stolen Data to a MongoDB Database*, https://www.bleepingcomputer.com/news/security/new-chrome-password-stealer-sends-stolen-data-to-a-mongodb-database/, [Accessed: 2020-02-06].

[53] X. Lin, *Expiro malware is back and even harder to remove*, https://www.mcafee.com/blogs/other-blogs/mcafee-labs/expiro-infects-encrypts-files-to-complicate-repair/, [Accessed: 2020-08-14].

[54] *Packerid*, [Accessed: 2021-04-03]. [Online]. Available: https://www.aldeid.com/wiki/Packerid.

[55] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, Chicago, Illinois, Nov. 2009, pp. 199–212.

[56] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson, "You've got Vulnerability: Exploring Effective Vulnerability Notifications," in *Proceedings of the 25th USENIX Security Symposium (Security)*, Austin, TX, Aug. 2016, pp. 1033–1050.

[57] A. J. Burstein, "Conducting Cybersecurity Research Legally and Ethically," in *Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, vol. 8, San Francisco, CA, Apr. 2008, pp. 1–8.

[58] *Moulton vs. VC3]*, http://www.internetlibrary.com/pdf/Moulton-VC3.pdf/, [Accessed: 2020-08-14].

[59] *Latest steam malware shows signs of rat activity*, https://blog.malwarebytes.com/cybercrime/2016/03/latest-steam-malware-shows-sign-of-rat-activity/, [Accessed: 2020-08-20].

[60] *Virustotal*, [Accessed: 2021-01-11]. [Online]. Available: https://www.virustotal.com/.

[61] L. Böck, E. Vasilomanolakis, M. Mühlhäuser, and S. Karuppayah, "Next Generation P2P Botnets: Monitoring Under Adverse Conditions," in *Proceedings of the 21th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Crete, Greece, Sep. 2018, pp. 511–531.

[62] S. Karuppayah, L. Böck, T. Grube, S. Manickam, M. Mühlhäuser, and M. Fischer, "Sensorbuster: On Identifying Sensor Nodes in P2P Botnets," in *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES)*, Reggio Calabria, Italy, Oct. 2017, pp. 1–6.

[63] P. M. Pondkule and B. Padmavathi, "BotShark—Detection and Prevention of Peer-to-peer Botnets by Tracking Conversation using CART," in *Proceedings of the International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, IEEE, vol. 1, 2017, pp. 291–295.

[64] S. Karuppayah, M. Fischer, C. Rossow, and M. Mühlhäuser, "On Advanced Monitoring in Resilient and Unstructured P2P Botnets," IEEE, 2014, pp. 871–877.

[65] J. Caballero, P. Poosankam, C. Kreibich, and D. Song, "Dispatcher: Enabling Active Botnet Infiltration Using Automatic Protocol Reverse-engineering," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, Chicago, Illinois, Nov. 2009, pp. 621–634.

[66] C. Y. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song, "Insights from the Inside: A View of Botnet Management from Infiltration," in *Proceedings of the 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, vol. 10, San Jose, CA, Apr. 2010, pp. 1–1.

[67] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M, "Unexpected Means of Protocol Inference," in *Proceedings of the Internet Measurement Conference (IMC)*, Rio de Janeiro, Brazil, Oct. 2006, pp. 313–326.

[68] W. Cui, J. Kannan, and H. J. Wang, "Discoverer: Automatic Protocol Reverse Engineering from Network Traces," in *Proceedings of the 26th USENIX Security Symposium (Security)*, Vancouver, BC, Canada, Aug. 2017, pp. 1–14.

[69] *The protocol informatics project.* [Online]. Available: http://www.baselineresearch.net/PI/.

[70] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer, "Dynamic Application-layer Protocol Analysis for Network Intrusion Detection," in *Proceedings of the 15th USENIX Security Symposium (Security)*, Vancouver, Canada, Jul. 2006, pp. 257–272.

[71] J. Caballero, H. Yin, Z. Liang, and D. Song, "Polyglot: Automatic Extraction of Protocol Message Format Using Dynamic Binary Analysis," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VA, Nov. 2007, pp. 317–329.

[72] Z. Lin, X. Jiang, D. Xu, and X. Zhang, "Automatic Protocol Format Reverse Engineering through Context-Aware Monitored Execution," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS)*, vol. 8, San Diego, CA, Feb. 2008, pp. 1–15.

[73] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda, "Prospex: Protocol Specification Extraction," in *Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P)*, Oakland, CA, May 2009, pp. 110–125.

[74] C. Cadar and D. Engler, "Execution Generated Test Cases: How to Make Systems Code Crash Itself," in *Proceedings of the International SPIN Workshop on Model Checking of Software*, Springer, San Francisco, CA, USA, Aug. 2005, pp. 2–23.

[75] V. Chipounov, V. Georgescu, C. Zamfir, and G. Candea, "Selective Symbolic Execution," in *Proceedings of the 5th Workshop on Hot Topics in System Dependability (HotDep)*, Estoril, Portugal, Jun. 2009.

[76] S. K. Cha, T. Avgerinos, A. Rebert, and D. Brumley, "Unleashing Mayhem on Binary Code," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2012, pp. 380–394.

[77] J. C. King, "Symbolic Execution and Program Testing," 7, vol. 19, ACM, 1976, pp. 385–394.

[78] R. S. Boyer, B. Elspas, and K. N. Levitt, "SELECT — A Formal System for Testing and Debugging Programs by Symbolic Execution," 6, vol. 10, ACM, 1975, pp. 234–245.

[79] L. A. Clarke, "A System to Generate Test Data and Symbolically Execute Programs," 3, IEEE, 1976, pp. 215–222.

[80] F. Peng, Z. Deng, X. Zhang, D. Xu, Z. Lin, and Z. Su, "X-force: Force-executing Binary Programs for Security Applications," in *Proceedings of the 23rd USENIX Security Symposium (Security)*, San Diego, CA, Aug. 2014, pp. 829–844.

[81] K. Kim, I. L. Kim, C. H. Kim, Y. Kwon, Y. Zheng, X. Zhang, and D. Xu, "J-force: Forced Execution on Javascript," in *Proceedings of the 26th International World Wide Web Conference (WWW)*, Perth, Australia, 2017, pp. 897–906.

[82] A. Naderi-Afooshteh, Y. Kwon, A. Nguyen-Tuong, A. Razmjoo-Qalaei, M.-R. Zamiri-Gourabi, and J. W. Davidson, "MalMax: Multi-Aspect Execution for Automated Dynamic Web Server Malware Analysis," in *Proceedings of the 26th ACM Conference on Computer and Communications Security (CCS)*, London, UK, Nov. 2011, pp. 1849–1866.

[83] C. Zuo and Z. Lin, "Smartgen: Exposing Server Urls of Mobile Apps with Selective Symbolic Execution," in *Proceedings of the 26th International World Wide Web Conference (WWW)*, Perth, Australia, 2017, pp. 867–876.

[84] L. Martignoni, E. Stinson, M. Fredrikson, S. Jha, and J. C. Mitchell, "A Layered Architecture for Detecting Malicious Behaviors," in *Proceedings of the 11th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Cambridge, Massachusetts, Sep. 2008, pp. 78–97.

[85] E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R. Kemmerer, "Behavior-based Spyware Detection," in *Proceedings of the 15th USENIX Security Symposium (Security)*, Vancouver, Canada, Jul. 2006, p. 694.

[86] E. Stinson and J. C. Mitchell, "Characterizing Bots' Remote Control Behavior," in *Proceedings of the Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, Lucerne, CH, Jul. 2007, pp. 89–108.

[87] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X.-y. Zhou, and X. Wang, "Effective and Efficient Malware Detection at the End Host," in *Proceedings of the 18th USENIX Security Symposium (Security)*, vol. 4, Montreal, Canada, Aug. 2009, pp. 351–366.

[88] Y. Aafer, W. Du, and H. Yin, "Droidapiminer: Mining Api-level Features for Robust Malware Detection in Android," in *International Conference on Security and Privacy in Communication Systems*, Springer, 2013, pp. 86–103.

[89] G. J. Széles and A. Coleşa, "Malware Clustering Based on Called API During Runtime," in *Proceedings of the International Workshop on Information and Operational Technology and Security (IOSec)*, Crete, GR, Sep. 2018, pp. 110–121.

[90] X. Deng and J. Mirkovic, "Malware Analysis Through High-level Behavior," in *Proceedings of the 11th USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, Baltimore, MD, Aug. 2018.

[91] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel, "A View on Current Malware Behaviors," in *Proceedings of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, Boston, MA, Apr. 2009.

[92] C. Kolbitsch, T. Holz, C. Kruegel, and E. Kirda, "Inspector Gadget: Automated Extraction of Proprietary Gadgets from Malware Binaries," in *Proceedings of the 31th IEEE Symposium on Security and Privacy (S&P)*, Oakland, CA, May 2010, pp. 29–44.

[93] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-away Traffic to Bots: Detecting the Rise of DGA-based Malware," in *Proceedings of the 21st USENIX Security Symposium (Security)*, Bellevue, WA, Aug. 2012, pp. 491–506.

[94] *Donot team leverages new modular malware framework in south asia*, https://www.netscout.com/blog/asert/donot-team-leverages-new-modular-malware-framework-south-asia, [Accessed: 2020-08-22].