Schrödinger's RAT: Profiling the Stakeholders in the Remote Access Trojan Ecosystem

Mohammad Rezaeirad,[†] *Brown Farinholt,*[◇] *Hitesh Dharmdasani,*[‡] *Paul Pearce*[††]
Kirill Levchenko,[◇] *Damon McCoy*[⋆]
[†]*GMU,* [◇]*UC San Diego,* [‡]*Informant Networks,* [††]*UC Berkeley,* [⋆]*NYU*

Abstract

Remote Access Trojans (RATs) are a class of malware that give an attacker direct, interactive access to a victim's personal computer, allowing the attacker to steal private data from the computer, spy on the victim in real-time using the camera and microphone, and interact directly with the victim via a dialog box. RATs are used for surveillance, information theft, and extortion of victims.

In this work, we report on the attackers and victims for two popular RATs, njRAT and DarkComet. Using the malware repository VirusTotal, we find all instances of these RATs and identify the domain names of their controllers. We then register those domains that have expired and direct them to our measurement infrastructure, allowing us to determine the victims of these campaigns. We investigate several techniques for excluding network scanners and sandbox executions of malware samples in order to filter apparent infections that are not real victims of the campaign. Our results show that over 99% of the 828,137 IP addresses that connected to our sinkhole are likely not real victims. We report on the number of victims, how long RAT campaigns remain active, and the geographic relationship between victims and attackers.

1 Introduction

Remote Access Trojans (RATs) are an emerging class of manually operated malware designed to give human operators direct interactive access to a victim's computer. Unlike automated malware (bots), RATs are predicated on the unique value of each infection, allowing an attacker to extort a human victim or otherwise benefit from access to a victim's private data. RAT software are sold and traded in underground communities as tools for voyeurism and blackmailing [11, 18].[1]

The unique danger posed by this new class of malware has received considerable attention in the security community and in popular media. Nevertheless, the *relationship* between the RAT operator and victim remains poorly understood. In this work, we bring to light the behavior of operators and victims of two popular RAT families, njRAT and DarkComet. Our primary aim is to determine *who* is attacking *whom*, the size of the victim and attacker population, and how long victims remain vulnerable after a campaign ends.

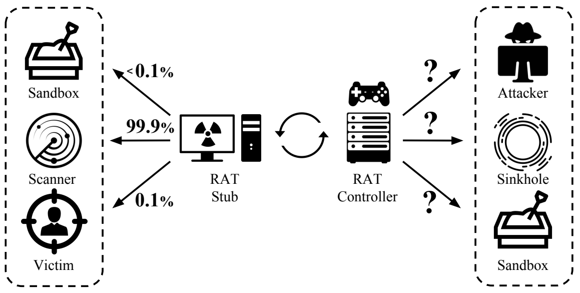


Figure 1: *Intelligence pollution* obfuscates the stakeholders in the RAT ecosystem.

One of the challenges of studying both victims and attackers remains the difficulty of accurately determining their population. Network hosts behaving as victims may be security researchers scanning for command-and-control servers [15, 17], and potential command-and-control servers may be vigilantes operating sinkholes [12, 39, 49]. The challenge of conducting a study of the RAT ecosystem is determining who is really a victim or operator, and who is pretending to be one.

In this paper, we design, implement, and deploy improved methods for measuring victims that connect to our sinkhole, called RAT-Hole, and for identifying RAT controllers using our scanner, called RAT-Scan. The task of identifying victims at scale is made difficult by the volume of traffic sinkholes receive from high-fidelity scanners and sandboxes. Similarly, differentiating between real controllers and sinkholes is complicated by high-fidelity sinkholes that look like real controllers. The in-

[1] While unskilled attackers appear to be their primary users, RATs have also been implicated in state-sponsored espionage and surveillance of journalists [46], dissidents [30], and corporations [27].

creasing fidelity in RAT scanners that emulate a real victim's behavior and sinkholes that emulate a real RAT controller's protocol has created an arms-race between completing threat intelligence operations, leading to inaccurate measurements and wasted notification efforts. We call this phenomenon *intelligence pollution*, illustrated in Figure 1.

Many operators of commodity-grade RATs rely on free dynamic DNS (DDNS) services for their command-and-control hostname [17]. While DDNS services give operators a free third-level domain name that they can associate with any IP address at any time, DDNS domains are not owned by the operator, and many DDNS services return the third-level name to the pool of available names after 30 days. We exploit this fact to carry out a study of victims, claiming the names of command-and-control servers as soon as they expire. This technique, called *sinkholing*, has been widely used to capture registrar-registered command-and-control domain names. Our work is the first to apply this technique to short-lived RAT command-and-control domain names.

Our high-fidelity sinkhole, RAT-Hole, implements the handshake protocol and error triggering tests for two common RATs, DarkComet and njRAT. Based on extensive empirical testing, we designed a set of heuristics to accurately differentiate sandboxes, scanners, and victims. We found that only 6,710 (0.8%) of the over 800,000 IP addresses that connected to RAT-Hole were likely victims. Further, only 3,231 (69%) of the unique hosts that completed a full RAT handshake with RAT-Hole were likely victims.

Our targeted RAT controller scanner, RAT-Scan can differentiate some sinkholes (including RAT-Hole) from actual RAT controllers by emulating DarkComet and njRAT victims. We deployed RAT-Scan to scan the entire IPv4 address space and found 6,401 IP addresses hosting suspected RAT controllers. Differentiating sinkholes from real RAT controllers is complicated by VPN proxy services that host both sinkholes and real RAT controllers. One such service, IPJetable, hosted over 40% of the suspected RAT controllers that we found.

In summary, in this work, we:

- Propose and evaluate methods for classifying RAT sandboxes, scanners, and likely victims based on connection to a sinkhole.
- Conduct a large-scale measurement study based on sinkholing 6,897 RAT controller domains. We find that only 6,710 (0.8%) of the over 800K IP addresses that connect to our sinkhole are likely victims.
- Propose and deploy a RAT controller scanner that is able to differentiate some sinkholes from real RAT controllers. Based on our analysis, we find that IPJetable, a free VPN service, hosts over 40% of the suspected RAT controllers that we contact.
- Identify potentially improved interventions that can mitigate the threat of RATs.

2 Background

The subject of this work is the relationship between victims and operators of two commodity RATs, njRAT and DarkComet. In this section, we provide the necessary background on RATs for the rest of the paper.

2.1 RAT Components

A typical RAT software package consists of two components: a *builder* program and a *controller* program. At the start of a malware campaign, the attacker uses the builder program to create a *stub* for installation on a victim's computer. The stub contains the code that will run on the victim's computer with parameters such as the host name of the command-and-control server to contact upon infection. During the campaign, the attacker runs the controller software on the command-and-control server to interact with the victims. In most cases (e.g., for njRAT and DarkComet), the controller provides a graphical user interface and runs directly on the attacker's computer. The attacker, also called the RAT *operator*, interacts with the victim via the controller interface.

2.2 RAT Command and Control Protocol

For most RATs, including the ones we study in this paper, communication between stub and controller begins with the stub opening a TCP connection to the controller host name hard-coded in the stub. The attacker provides this host name to the builder program, which produces the stub. Once the stub establishes a connection to the controller, RATs can be divided into two classes. In RATs such as DarkComet, the application-layer handshake is *controller-initiated*: the controller speaks first by sending a *banner* to the stub immediately after accepting the stub's TCP connection. In RATs such as njRAT, where the handshake is *victim-initiated*, the stub sends the first handshake message immediately upon connecting (receiving the SYN-ACK from the controller). Whether a protocol handshake is victim-initiated or controller-initiated determines how we scan for controllers and sinkhole stubs, as described below.

The initial message sent by a stub contains both information configured by the builder (e.g., password, campaign ID) as well as information unique to the victim machine (e.g., username, hostname, operating system, active window). This information allows the operator to manage multiple campaigns and also to obtain a summary of the victim. Some of the information sent by the stub is potentially Personally Identifiable Information (PII), introducing ethical challenges to RAT sinkholing, as we discuss in Section 2.5.

In addition, many RAT protocols use symmetric encryption to obfuscate the command stream and as a form of access control to the stub. The encryption key is often embedded in the stub's configuration, requiring a modest reverse-engineering effort to extract the key.

2.3 Sinkholing

Sinkholing is a term used to indicate the redirection of infected machines' connections from their intended destinations (e.g., attackers' command-and-control servers [36]) to the sinkhole owner. Local sinkholing efforts, implemented by organizations or individual ISPs, often involve reconfiguring DNS servers and routers to block communication with malicious domains or IP addresses. Larger, coordinated sinkhole operations are often part of broader takedown efforts, requiring cooperation between domain registrars and international authorities. [29, 49]

A prior study found that RAT operators often utilize Dynamic DNS (DDNS) services [17], which allow their controllers to migrate between IP addresses without disruption of operation. Services such as No-IP [35] offer free DDNS hostname registrations that expire after 30 days. As we will show, operators often allow their hostnames to expire and this provides a large pool of RAT domain names that can be claimed and sinkholed.

Ideally, a DNS sinkhole operation would be able to identify all victims associated with its acquired domains and to accurately measure victims. Unfortunately, scanners and sandboxes introduce a significant amount of intelligence pollution, as we will show in our study.

2.4 Scanning

Internet-wide scanning is a popular technique for Internet measurement, particularly in the security community. It was recently used to measure the Mirai botnet [3], and is used by academic groups and security vendors alike. Open-source tools such as ZMap [14] make rapid scanning of IPv4 space accessible to researchers. Services such as Censys [13] (based on ZMap) and Shodan [32] also provide access to the results of such scans.

Scanning for RAT controllers presents a similar set of challenges to sinkholing. RATs often use victim-initiated handshake protocols to communicate, so simple port scanning or banner grabbing is often not sufficient to confirm the existence of a RAT controller. One must also implement the RAT's handshake, which can be complicated by the inclusion of encryption and custom passwords. Proxies may also conceal multiple controllers behind the same address, while a single controller may reside behind changing addresses (using DDNS, for example). Finally, many academic groups and security vendors operate sinkholes, which makes identifying real RAT controllers more challenging.

2.5 Ethical Framework

Our methodology was approved by our institution's Institutional Review Boards (IRB) and general legal counsel. The ethical framework that we operated under is that we only completed the protocol handshake with peers that contacted us and controllers that are publicly reachable. We did not attempt to execute any commands on infected peers. During the handshake there is some potentially Personal Identifiable Information (PII) that the peer sends to us, such as the PC name (often the name of the victim) or full website URLs a person is visiting if the active window is a browser. In order to mitigate the potential harm caused by our study, we immediately encrypted any fields that might contain PII and did not ever store an unencrypted version of these fields (PII listed at Table 3). Our IRB takes the position that IP addresses are not personally identifiable. In no cases did we attempt to tie our measurements to an actual identity.

3 RAT-Hole Methodology and Dataset

Our system consists of two primary components: a high-fidelity sinkhole (*RAT-Hole*) that imitates RAT controllers, and a high-fidelity scanner (*RAT-Scan*) that imitates RAT victims. We present the details of our RAT-Scan system in Section 5. Figure 3 shows a timeline of when each part of our methodology was deployed.

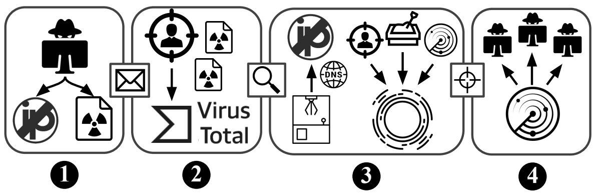


Figure 2: The major components of our operation and their interactions with the subjects of our study.

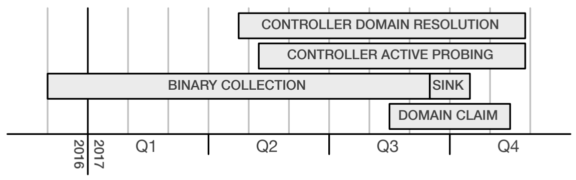


Figure 3: Timeline of data collection phases of our study; Binary Acquisition (3.1), Controller Domain Resolution (3.2), Scanning for Controllers (5.4), Domain Claiming (3.2), and Sinkholing (3.4).

Figure 2 depicts the system's operation. An attacker first registers a hostname with a DDNS provider such as No-IP, creates malware binaries configured with this hostname, and spreads the binaries to victims in the wild (❶). Some of the malware is detected and uploaded to VirusTotal [52] (❷). Our VirusTotal-deployed YARA rules [54] find these malware samples, from which we extract configurations, including controller IP addresses, hostnames, and passwords (❸). Many hostnames belong to No-IP. Our DDNS Claimer registers any expired, No-IP-owned hostnames and configures them to resolve to RAT-Hole's IP range. RAT-Hole then receives all connection attempts to these hostnames. Simultaneously, RAT-Scan continuously probes all extracted IP addresses and hostnames for controller activity in the wild (❹).

We limit the scope of our study to two RAT families, DarkComet and njRAT, since reverse engineering and implementing parts of their protocols in RAT-Hole and RAT-Scan is a time-consuming manual effort. These two were chosen because they are the most popular RAT families for which we were able to obtain source code for multiple versions, and there is existing documentation of their protocols to assist with the reverse engineering process [6, 10, 20]. As an approximate measure of prevalence, we count the number of unique binaries associated with RAT families uploaded to VirusTotal by using up-to-date YARA rules for sample hunting. We found that njRAT and DarkComet were the third and fifth most popular RATs on VirusTotal when we began our study.

3.1 RAT Binary Acquisition

Using YARA signatures for all known versions of DarkComet and njRAT, we monitored VirusTotal for 9 months (2016-12-01 to 2017-08-17), obtaining 33,560 samples in all. Each sample has a configuration, including the address of its controller (domain:port or IP:port), its campaign ID, its password, and its version. We attempted to extract configurations from each sample using an existing Python tool [5]. Table 1 shows that we were able to obtain configurations for 22,124 *unique* samples of DarkComet and 4,535 *unique* samples of njRAT. Our njRAT YARA rule can detect subfamilies of DarkComet and njRAT that our decoder does not support. This is one of the primary reasons why we fail to decode RAT samples.

Family	# Sample	% Sample	# Unique
DarkComet	22,362	66.6	22,124
njRAT	5,049	15.0	4,535
Other	5	<0.1	-
Failed Decoding	6,144	18.3	-
Total	33,560	100.0	26,659

Table 1: Counts of RAT samples downloaded, both total and unique, by family. *Other* are RAT samples that matched our YARA signatures incorrectly. *Failed Decoding* are samples from which configurations could not be extracted.

3.2 Domain Claiming

Domain dataset. We performed an analysis of the domain names found in the RAT configurations. Table 2 shows that most of the domains we found that were used by DarkComet and njRAT are associated with free Dynamic DNS (DDNS) providers, and that No-IP managed 60% of all discovered domains (77% of DDNS).

Controller Type	# Domain	% Domain
No-IP	8,564	60.0
DuckDNS	2,459	17.2
FreeDNS	92	<0.1
DynDNS	38	<0.1
Total Dynamic DNS	11,153	78.1
Unknown	3,120	21.9
Total	14,273	100.0

Table 2: Breakdown of C&C domains in our RAT sample population by Dynamic DNS provider. *Unknown* encompasses all domains unrelated to a known DDNS provider.

DDNS Claimer. We developed a web automation toolkit to automate the process of identifying expired DarkComet and njRAT domains controlled by No-IP and claiming them. We only claimed domains from No-IP, since there is manual effort involved in building the web automation toolkit and No-IP was by far the most popular free DDNS provider used by DarkComet and njRAT. We limit our study to only expired domains that we can claim freely; we do *not* attempt to seize owned domains, in order to avoid additional disruption to the ecosystem that we were measuring. As future work we plan to expand our claiming infrastructure to other DDNS providers and actual registered domains to understand if this affects the results of our analysis.

Our DDNS domain claiming operation, which started on 2017-07-15 and ended on 2017-11-17, claimed 6,897, or 81%, of the 8,564 DDNS domains managed by No-IP.

3.3 RAT-Hole Operation

Architecture. A RAT-Hole node has three sets of interfaces. (1) *Management:* All management traffic, such as data backup and database iterations, are performed using this interface. (2) *Sinkholing:* This set of virtual interfaces (206 IP addresses) are assigned to the sinkholed domains by our domain claiming system. We randomly claimed 200[2] expired DarkComet and njRAT from the set of No-IP domains that we discovered. These domains were monitored for one hour. After that one hour we released these domains, waited for 5 minutes for the old domain names to expire from DNS caches, and then claimed a new set of 200 randomly selected domains.

[2]Our paid account with No-IP allowed us to hold 500 domains at once, but we only claimed 200 at a time due to our limited number of sinkhole IP addresses.

(3) *Decoy:* We intentionally did not use 11 of the IP addresses in our subset as sinkholing interfaces so that we could identify connection attempts to them that indicate scanning behavior. We randomly selected IP addresses to be decoy interfaces.

RAT-Hole includes a Deep Packet Inspection (DPI) pattern-matching based state machine that maintains the state of each active TCP connection from the sink-hole and decoy interfaces. These states are:

• **Incoming.** RAT-Hole allocates a data structure indexed by connection ID for any TCP connection that completes a TCP three-way handshake with the RAT-Hole. Here, connection ID is a tuple of source IP address, destination IP address and TCP port. This ID is used as an index by RAT-Hole to locate the previous states of the connection and to update the connection's state.

• **Protocol detection.** Once a TCP connection is established RAT-Hole waits three seconds to receive data. After three seconds it will probe the peer by sending a DarkComet command-and-control banner to determine if it is an njRAT stub and update the state of the connection.

• **Victim-initiated RAT protocol detection.** When RAT-Hole receives an initial message from a completed TCP connection before the three second timer expires, it will examine all of our regular expression-based RAT family detection signatures over the payload to identify the incoming peer's protocol. We have created 16 signatures that are able to detect the initial message sent by common versions of njRAT, XtreamRAT, ShadowTech, NetWire, H-Worm, LuminosityLink, Black WorM, and KJ w0rm. However, we implement the complete njRAT handshake to determine if it is an actual njRAT stub or intelligence gathering operation. Although we did not claim any domains used by these variants, we did see initial messages for these other RAT families.

• **Controller-initiated RAT protocol detection.** If the three second timer expires, then RAT-Hole will probe the peer by sending a DarkComet banner. For DarkComet (refer to Appendix D for more details on DarkComet protocol behavior), we send a series of 125 banners. After each banner is sent the connection is terminated and the stub will attempt to connect again if the stub conforms to the DarkComet protocol. This ordered set of banners is comprised of: (1) a randomly generated banner (this is to test if the stub deviates from the protocol by accepting any banner), (2) a set of 20 default DarkComet banners (this is to see if the stub will incorrectly accept the default banner), (3) a complete set of valid banners for the current set of domain(s) being sinkholed by that interface (this is to see if the stub responds correctly to a valid banner), and (4) the remainder of the 125 banners are randomly generated (based on our reverse engineering we found that after 124 banners are received by a stub it will lock up and stop attempting further connections until the RAT process is restarted).

• **Handshake test.** We implemented a handshake test for both DarkComet and njRAT which implements each full RAT handshake protocol. In addition, it includes specifically malformed messages in order to trigger error handling and identify likely RAT stub execution as opposed to a scanner that has reimplemented the protocol. This methodology is similar to that of Marczak *et. al.* to fingerprint malware command-and-control servers [30]. We send a malformed command at the end of the njRAT handshake (see Appendix C) and if the expected error handling response is received, we mark the connecting peer as conforming to the standard implementation of the protocol. In the same vein, for DarkComet (see Appendix D) we observe and enumerate the execution pattern. For instance, we expect a true DarkComet infection to stop operating after connecting to RAT-Hole as a result of 124 attempts or more. Note that our handshake test does not distinguish between stub execution in sandbox and victim machine. However, it does perform well at identifying RAT scanners specifically those that do not implement the complete RAT protocol handshake and true execution behavior of the RAT protocol as standardly implemented.

• **Clean up.** After handshake probing is completed, RAT-Hole closes the connection and removes the state for that connection ID. RAT-Hole logs the final state of the connection, which can be one of three states: (1) no valid banner was received; (2) a valid initial banner was received, but the handshake was not completed; (3) the handshake was completed successfully.

Handshake metadata. During the RAT handshake, the stub includes a number of fields in the handshake that we parse and store in the database. A list of the fields that we parse and store is shown in Table 3. Using AES we encrypt any fields that might contain potential PII before storing them in our database. From these fields, we create a **fingerprint** that is largely unique for each infection by combining the `PC-NAME`, `USERNAME`, and `HWID`. `VSN` and `UUID` define the `HWID` for njRAT and DarkComet, respectively. (See Appendix C, D.) This fingerprint enables us to persistently identify and thus enumerate unique RAT peers despite victim IP address churn.

RAT-Hole classification. Based on this, we classify peers that connected to RAT-Hole into five categories:

1. Unknown: These peers send a payload that is not known to RAT-Hole. Internet wide scanners (e.g., ZMap) and other custom IP intelligence operations are examples of these peers. We have also found that these unknown payloads could be from RAT families that RAT-Hole does not support, since some of the sinkholed domains are used by multiple RAT infection campaigns.

2. Low Fidelity (LF) Scanner: These peers do not complete the RAT handshake. Based on our analysis they often send many fingerprints, connect from many Src-IP addresses, connect to our decoy interfaces, and their Src-IP address might be included in known scanner list(s).
3. High Fidelity (HF) Scanner: These peers complete the RAT handshake and present one or more highly anomalous characteristics described in Table 4.
4. Sandbox Execution: These peers complete the RAT handshake, but exhibit one or more of the characteristics commonly associated with a sandbox execution, such as a short execution duration which we defined as slightly more than the longest default execution duration value (600 seconds) of the major sandboxes we analyzed. Table 4 includes a compete list of these heuristics.
5. Victim: These are likely real infections.

3.4 Operation Summary

Field	Description	PII
`ACTIVE_WINDOW`	Title and content of currently open window	✓
`CAMPAIGN_ID`	Stub's identity which operator defines	✓
`COUNTRY`	Geo-Location of victim's machine	✓
`HWID`	Hardware identity of victim's machine	✓
`INSTALL_DATE`	First day on which stub was executed	-
`LAN_IP`$^{\mathbb{DC}}$	Private IP address of victim's machine	-
`LANGUAGE`$^{\mathbb{DC}}$	Language setting of victim's machine	-
`OS`	Operating system name of victim's machine	-
`PC_NAME`	PC name of victim's machine	✓
`USERNAME`	Username of victim's machine	✓
`PORT`$^{\mathbb{DC}}$	Port number of stub	-
`VERSION`	Version of RAT	-
`WAN_IP`$^{\mathbb{DC}}$	Public IP address of victim's machine	-
`WEBCAM_FLAG`	Webcam capture is supported	-

Table 3: Fields extracted from handshakes for DarkComet and njRAT families. *PII* indicates whether we consider the field to be potential PII of the victim, and determines whether we AES encrypt the value. ($\mathbb{DC}$) identifies DarkComet specific fields.

Over 31 days (from 2017-08-15 to 2017-09-16), we sinkholed 6,897, or 81%, of the 8,564 No-IP domains. 4,493 of these domains came from DarkComet samples, 2,381 from njRAT samples, and 23 were found in samples of both families. Over the 31 days it was deployed, RAT-Hole was in *possession* of domains for 23.1 total days - an average of 17.7 hours per domain, distributed randomly. During this time, it received 153,100,000 TCP connections. Table 5 provides a high-level view of these connections, broken down by determined peer type.

We performed an analysis of the "Unknown" peer type from Table 5 which composed 815,455 (98.5%) of all IP addresses that completed a three-way TCP handshake connection to RAT-Hole, but were not classified by RAT-Hole as peers related to either njRAT or DarkComet. We suspected that some of these connections might be other RAT families when an operator reuses the same DDNS domain for other RAT campaigns. In order to provide some measurements of this phenomenon, RAT-Hole implements a simple payload parser for the first message of the handshake for 19 other popular RAT families in addition to the complete RAT handshake protocol for njRAT and DarkComet RAT families. Of these peers, 73.6% sent no additional TCP messages after the handshake, 31.9% sent unknown payloads, and 1,463 ($<$1%) were detected as other types of RAT families. The small degree of overlap indicates that some IP addresses presented multiple behaviors; see Table 17 in Appendix E.

We also wanted to understand if this pollution from likely sandboxes and scanners could be filtered using data from IP telescopes (unused IP address subnets that act as large sinkholes). To evaluate this possibility, we looked for overlap in IP addresses during our deployment period from a /24 sized (256 IP addresses) IP telescope located in India. We find that there is not much overlap. Only 31,014 (3.8%) of the IP addresses we classified as *Unknown* appear in our telescope data, and less than 0.01% overlap with any other category of IP addresses. This suggests that most of RAT-Hole's pollution is targeted and thus not filterable. See Appendix B for details.

4 RAT-Hole Validation

This section details our efforts to validate our methodology for differentiating RAT scanners, sandboxes, and victims. Validating our methodology is challenging since we have limited ground truth, except in some instances where we could create it (e.g. Section 4.3). In this section, we describe our methods for building a high-confidence set of RAT scanners, sandboxes, and victims based on heuristics specific to DarkComet and njRAT.

4.1 Low Fidelity (LF) Scanners

Recall that low fidelity scanners are peers[3] that sent valid initial handshake messages, but did not complete the handshake process. In Table 6, we separate 1,421 IP addresses that are all njRAT into five clusters based on their behavior. The first cluster are source IP addresses that attempted to connect to one of our decoy IP addresses. This is a fairly strong indication of broader IP address scanning being performed by this source IP address and we are confident that these are scanners. As a point of reference no high fidelity scanner, sandbox, or victim connected to one of our decoy IP addresses.

For cluster ②, 7,607 (99.7%) of the fingerprints only attempted to establish one connection. This is a strong indication of a scanner that is randomizing its fingerprint. Another two fingerprints had multiple unique `INSTALL_DATE` fields, indicating the possibility that they are sandboxes. For 19 of the fingerprints, we did not detect any anomalies. These 19 peers could be real victims

[3]In the context of connections to RAT-Hole, a peer is identified by its unique fingerprint. See Section 3.3 for details.

Peer Type	Anomaly	Anomaly Type	Description
HF Scanner	Empty Install Date	Field Format	Peer(1) sent RAT payload with an empty installed date
	VSN Format$^{\mathbb{NJ}}$	Field Format	Peer sent RAT payload with malformed VSN
	HWID Format$^{\mathbb{DC}}$	Field Format	Peer sent RAT payload with malformed UUID
	Empty GeoLoc	Field Format	Peer sent RAT payload with an empty Geo-location data
	GeoLoc Format	Field Format	Peer sent RAT payload with malformed Geo-location data
	Mismatch SRC-IP$^{\mathbb{DC}}$	Protocol Behavior	Peer sent RAT payload with Src-IP address other than peer Src-IP address
	Mismatch DST-PORT$^{\mathbb{DC}}$	Protocol Behavior	Peer sent RAT payload with Dst-Port number other than peer Dst-Port number
	124+ Banners: Session$^{\mathbb{DC}}$	Protocol Behavior	Peer tried to connect to RAT-Hole more than 124 times during a session
	Multiple OS Name: Session(2)	Protocol Behavior	Peer sent RAT payload with different OS names across different connections
	Multiple Passwords: Session$^{\mathbb{DC}}$	Protocol Behavior	Peer tried to connect to RAT-Hole using multiple passwords during a session
	Solo Connection Attempt: Global	Peer Behavior	Peer tried to connect (probe) to RAT-Hole only once
	Unexpected Dst IP: Decoy	Peer Behavior	Peer contacted one of the Decoy interfaces
Sandbox	Multiple Install Date: Session	Protocol Behavior	Peer sent RAT payload with multiple install date during a session
	Multiple Campaign ID: Session	Protocol Behavior	Peer sent RAT payload with multiple Campaign ID during a session
	Multiple Passwords: Global(3)$^{\mathbb{DC}}$	Peer Behavior	Peer tried to connect to RAT-Hole using multiple passwords across multiple sessions
	Small Activity Duration: Global	Peer Behavior	Peer were active for small durations (less than 600 seconds) for all sessions
	Low Active Windows: Global	Peer Behavior	Peer sent RAT payload with small number of active windows(4) during all sessions
	Multiple Dsts: Session	Peer Behavior	Peer contacted multiple(5) Dsts (Dst-IP and Port) during a session

Table 4: Anomaly, Anomaly type and their descriptions used by RAT-Hole peer classifier.
(1) Peers are identified by Fingerprint. (2) Session = FP + Src-IP + Dst-IP + Dst-Port. (3) Global: All the sessions belonging to a fingerprint. (4) Condition in Row 2 is checked first and Row 3 is followed. (5) We account for domain rotation where a domain is registered under the different RAT-Hole interfaces. ($\mathbb{DC}$) DarkComet specific rule. ($\mathbb{NJ}$) njRAT specific rule

	Connection		Src-IP		Fingerprint (FP)		ASN†		Country†	
Peer Type	Count	Pct.	Count	Pct.	Count	Pct.	Count	Pct.	Count	Pct.
Victim	5,320,297	3.5	6,710	0.8	3,231	0.1	1,079	10.1	108	50.0
Sandbox	372,883	0.2	1,181	0.1	877	<0.1	418	3.9	85	39.4
HF Scanner	563,019	0.4	1,349	0.2	589	<0.1	347	3.2	73	33.8
LF Scanner	17,746,010	11.6	1,421	0.2	4,114,064	99.9	390	3.6	80	37.0
Unknown	129,097,791	84.3	815,455	98.5	N/A	N/A	10,418	97.2	216	100.0
Total	153,100,000	100.0	828,137	100.0	4,118,761	100.0	10,722	100.0	216	100.0

Table 5: Summary of connections received by RAT-Hole, grouped by peer type, fingerprint, Src-IP, ASN, and country. The first three rows (Victims, Sandboxes, and HF Scanners) are detailed in Table 8, while LF Scanners are described in Table 6. †Note that ASN and country show a significant amount of overlap across peer types.

	FP		Src-IP	
Cluster Name	Count	Pct.	Count	Pct.
① Decoy Interface	4,105,659	99.8	28	2.0
② Many FPs Per Src-IP	7,628	0.2	39	2.7
③ Many Src-IPs Per FP	261	<0.1	827	58.2
④ Many FPs, Many Src-IPs	6	<0.1	17	1.2
⑤ Single FP, Single Src-IP	510	<0.1	510	35.9
Total	4,114,064	100.0	1,421	100.0

Table 6: Breakdown of LF (Low Fidelity) Scanners

that speak a version of the protocol that is incompatible with RAT-Hole, or that have persistent connectivity issues that prevented them from completing a handshake. We conservatively label these peers low fidelity scanners.

Our anomaly analysis for fingerprints in cluster ③ shows that 140 (53.6%) of the fingerprints have multiple unique `INSTALL_DATE` fields, likely indicative of scanners that update `INSTALL_DATE` based on the current time. Another 24 (9.2%) had an incorrectly formatted `HWID`, indicating scanners with protocol formatting errors. The remaining 97 (37.2%) had no anomalies, but again we conservatively label them low fidelity scanners.

Three of the six fingerprints in cluster ④ had multiple unique `INSTALL_DATE` fields, again indicating likely scanners that update `INSTALL_DATE` based on the current time. The remaining three did not have anomalies, but we conservatively label these peers as low fidelity scanners.

Anomaly Type			FP	
Field Format	Peer Behavior	Protocol Behavior	Count	Pct.
		✓	60	11.8
✓			47	9.2
✓		✓	9	1.8
	✓	✓	6	1.2
✓	✓		1	0.2
Remainder			387	75.9
Total			510	100.0

Table 7: Breakdown of ⑤: Single FP Single Src-IP

We did not observe any RAT protocol violations from 387 out of 510 fingerprints (remainders at Table 7) belonging to the peers that had one fingerprint and one IP address (⑤). On the average peers in this cluster failed 3,000 (90%) attempted connections and a minimum of 100 (2%) attempted connections. Thus it is unlikely that intermittent connectivity issues prevented the completion of the handshake at least once. It is unclear if these are victims that implemented a version of the protocol that is incompatible with our RAT-Hole, persistent connectivity issues, or if they are low fidelity scanners that did not implement the entire protocol. We conservatively label these peers as low fidelity scanners.

4.2 Victims, Sandboxes, & High Fidelity (HF) Scanners

We classify a peer as a high fidelity scanner if it is able to complete the handshake, but it violates the field formatting, exhibits peer behavior, or protocol behavior that indicates it is likely a scanner that is reimplementing the njRAT or DarkComet stub instead of an actual stub execution. A peer is conservatively classified as a sandbox if it exhibits peer or protocol behavior that indicates it is likely a sandbox. Finally, if a peer does not violate the protocol or exhibit any anomalous behavior we classify it as a likely victim. Table 8 shows that 69% (3,231) of all peers that complete the handshake with our RAT-Hole are classified as victims. This indicates the significant degree to which high-fidelity scanners and sandboxes will pollute sinkhole results if the sinkhole eschews a deeper analysis of the peers similar to RAT-Hole.

Table 8 also shows the breakdown of types of anomalous behavior and protocols violations observed by likely high fidelity scanners and sandboxes. For high fidelity scanners they had an incorrectly formatted field or an empty `INSTALL_DATE` for 238 (40.4%) and 174 (29.5%) of the fingerprints accordingly. Sandboxes exhibit short execution durations 634 (72.2%) and multiple unique `INSTALL_DATE` fields in 259 (29.5%) of the fingerprints. While we cannot compute error rates for our classifications due to the lack of ground truth, we are fairy confident that our methodology, while not perfect, is reasonably accurate. In the next section we present the results of seeding malware analysis portals to further validate our classification methodology. Finally, what we classify as victims are the fingerprints that do not exhibit any anomalous behavior and are likely to be actual victims.

4.3 Honey Sample Seeding

In order to evaluate our classification in a setting where we have ground truth, we conducted an experiment where we uploaded DarkComet and njRAT samples to malware analysis services. Our expectation for this experiment is that all of the connections will be from scanners or sandboxes, which will enable us to validate our classification methodology.

		Anomaly Type			FP	
Peer Type	**RAT Family**	**Field Format**	**Peer Behavior**	**Protocol Behavior**	**Count**	**Pct.**
HF Scanner	DarkComet	✓			130	46.1
HF Scanner	DarkComet	✓	✓		35	12.4
HF Scanner	DarkComet	✓		✓	11	3.9
HF Scanner	DarkComet		✓		16	5.7
HF Scanner	DarkComet		✓	✓	5	1.8
HF Scanner	DarkComet			✓	85	30.1
Subtotal					282	100.0
HF Scanner	njRAT	✓			200	65.2
HF Scanner	njRAT	✓	✓		31	10.1
HF Scanner	njRAT	✓	✓	✓	1	0.3
HF Scanner	njRAT	✓		✓	6	2.0
HF Scanner	njRAT		✓		6	2.0
HF Scanner	njRAT		✓	✓	7	2.3
HF Scanner	njRAT			✓	56	18.2
Subtotal					307	100.0
Sandbox	DarkComet		✓		318	63.0
Sandbox	DarkComet		✓	✓	26	5.2
Sandbox	DarkComet			✓	161	31.9
Subtotal					505	100.0
Sandbox	njRAT		✓		294	79.0
Sandbox	njRAT		✓	✓	6	1.6
Sandbox	njRAT			✓	72	19.4
Subtotal					372	100.0
Victim	DarkComet				841	26.0
Victim	njRAT				2,390	74.0
Subtotal					3,231	100.0
Total					9,191	

Table 8: Breakdown of Anomalies for Different Peer Types

		Anomaly Type			FP	
Peer Type	**RAT Family**	**Field Format**	**Peer Behavior**	**Protocol Behavior**	**Count**	**Pct.**
HF Scanner	njRAT	✓	✓		4	66.7
HF Scanner	njRAT	✓	✓	✓	2	33.3
Subtotal					6	100.0
Sandbox	DarkComet		✓		58	98.3
Sandbox	DarkComet		✓	✓	1	1.7
Subtotal					59	100.0
Sandbox	njRAT		✓		48	96.0
Sandbox	njRAT		✓	✓	2	4.0
Subtotal					50	100.0
Victim	njRAT				2	100.0
Subtotal					2	100.0
Total					117	

Table 9: Breakdown of Anomalies for Different Peer Types for Honey Sample Seeding Experiment

Using our automated RAT Seeder, we generated 84 DarkComet and 84 njRATsamples. Each of these samples has a unique campaign ID, IP address, and TCP port configuration that directed the sample to connect to one of our RAT-Hole IP addresses on a different network segment, which we only used for this experiment. We uploaded 4 DarkComet and 4 njRAT samples to 21 different malware analysis services, of which only 9 of the services initiated a connection for at least one of our samples. A full list of these services and the ones that initiated a connection can be found in Table 18.

We chose these services based on their popularity among malware researchers and threat hunters, ease of utilization and being relativity cheap or free. Only 9 of these services executed one or more of our honey samples during the course of our experiment. The configuration uniqueness of these samples allowed us to associate received connections to RAT-Hole with a sample and portal. Table 9 shows the breakdown of fingerprints and associated categorization of peers by our classification engine. Note that we used the same classification methodology as for our in the wild sinkholing experiments and only incorrectly classified 2 out of 117 (1.7%) fingerprints as victims. We inspected the active windows for these two fingerprints and found that both appear to be manually reverse engineering the samples using executable debugging and network protocol analysis tools. Recall that for this experiment, we did not encrypt the active windows since we did not expect any real victims. We also classified some njRAT peers as high fidelity scanners. We can confirm that when we tested these samples before submitting them they did not have any protocol violations. This gives us further confidence that our classification methodology is fairly accurate.

5 RAT-Scan Operation

5.1 Controller Tracking

In order to maintain an updated list of potential command-and-control addresses, we resolved each of the 14,273 domains we extracted from our malware samples hourly, beginning on 2017-04-21 and ending on 2017-11-26. Over this period, we recorded 67,023 resolutions to unique IP addresses. We augmented these with passive DNS records dating back to 2010 from Farsight [19], VirusTotal, and PassiveTotal [43].

5.2 Active Scanning

We continuously probed each of these 67,023 IP addresses hourly for evidence of RAT controller software. We checked for services running on any port configured in any sample related to the IP address or related to a domain that resolved to said IP address at any time.

RAT-Scan probes for controllers of both DarkComet and njRAT, emulating a newly-infected *victim* contacting the controller for the first time. RAT-Scan first approaches every connection passively, waiting to receive an initial DarkComet handshake banner. If it does not receive a banner before a three second timeout, it restarts the connection and treats it actively, sending the initial njRAT handshake banner. Regardless of which handshake proceeds, the scanner completes the entire handshake with the controller if possible.

Sinkhole identification. RAT-Scan can distinguish between legitimate controllers and some sinkhole operations, including our own RAT-Hole. If a controller begins a handshake but does not complete it, it is labeled as a sinkhole. Additionally, after successfully completing a handshake with a controller, our scanner attempts to elicit an improper response to a second handshake with a different configuration (e.g. different password). Any response is cause for sinkhole classification.

5.3 Detected Service Classification

The actors that our scanner probed during its operation fall under one of the following classes: (1) a *controller* completes an njRAT or DarkComet handshake flawlessly and does not respond to solicitation for improper behavior; (2) a *sinkhole* either makes an error during a RAT handshake, or accepts an improper second handshake after the first (like RAT-Hole).

Important caveats. RAT-Hole and RAT-Scan have a significant disparity in the confidence of their classifications. RAT-Hole makes use of several protocol artifacts in the DarkComet and njRAT handshakes to detect imposter victims. Because RAT victims are intentionally loquacious during the handshake, this is possible; however, RAT controllers are oppositely taciturn, revealing practically nothing to RAT-Scan during the handshake. DarkComet controllers acknowledge a victims' correct password and njRAT controllers do not acknowledge this. Therefore, when we classify a host as a DarkComet sinkhole we are fairly confident, but when we label a host a controller it is possible that it is a high-fidelity sinkhole or sandboxed controller.

Attempted validation. The investigation by Recorded Future and Shodan in 2015 that resulted in Malware Hunter [22] flagged 696 IP addresses as RAT controllers, 10 of which appear in our dataset. However, Malware Hunter has since flagged RAT-Hole as a RAT controller and high-priority threat, so we question the value of any such threat intelligence feed as proper ground truth. We leave developing a method for improved validation of our scanning results as future work.

5.4 Operation Summary

Controller Type	# IP	% IP
njRAT	4,584	71.6
DarkComet	2,032	31.7
DarkComet (Unknown Password)	11	0.2
Total	6,401	100.0

Table 10: Breakdown of RAT controllers detected on IP addresses responsive to RAT-Scan. Some IP addresses hosted multiple types of RAT controller.

Our scanning operation began on 2017-05-11 and ended on 2017-11-25, for a total of 198 days. During this period, we established 86,694 connections to 6,401 IP addresses exhibiting behavior indicative of RAT controller software; 2,032 DarkComet controllers and 4,584 njRAT controllers, with some IPs hosting both. Table 10 provides a summary of our scanning operation.

Other than on RAT-Hole itself, our sinkhole detection methods did not trigger during this study. We are led to believe that all controllers reported here are either legitimate instances of the controller software, or services that have implemented the handshake properly and maintain a single configuration. We suspect that such services exist; however, we currently have no way of distinguishing them from legitimate controllers. Further, we have no reason to believe that we encountered any high-fidelity sinkholes similar to RAT-Hole.

6 Measurements and Analysis

6.1 Victim Analysis

IP address churn. We find that most victims do not change their IP address. 60% of victims use one IP addresses, with an additional 20% of victims using just two IP addresses. Note that we might not observe all of the victims' IP address changes due to our incomplete sinkholing of domains.

Webcam availability. As part of the handshake, DarkComet and njRAT victims report if they have a camera device. We found that 1,725 (53.4%) of victims have a camera, making them susceptible to visual monitoring (unless they have physically covered the camera).

Infected servers. 21 njRAT victims reported a server version of Windows (i.e., Windows Server 2012) running on the peer. We manually investigated the Autonomous System Numbers for the IP addresses used by these peers and confirmed that they were located on corporate networks or cloud hosting providers. This suggests that some higher profile peers associated with companies are infected with njRAT, providing the operator with an entry point into their systems.

Infection longevity. Our methodology is predicated on victims remaining after the command-and-control dynamic DNS domain used by the attacker expires, which occurs 30 days after registration with No-IP. Figure 4 shows the fraction of domains still receiving legitimate victim connections as a function of time since the dynamic DNS domain expired. Because our sinkholing period does not span our full domain monitoring period (31 days from 2017-08-15 to 2017-09-16, and 220 days from 2017-04-21 to 2017-11-26; see Figure 3), we do not necessarily known victim availability immediately after domain expiration. Figure 4 shows an upper and lower bound curve; the upper bound corresponds to the case that at least one victim connection occurred during the period when the command-and-control domain was not monitored, and the lower bound corresponds to the case that no victim connections occurred during the same period. Thus, 120 days after the command-and-control domain expired, 10% of domains were still receiving connections from legitimate victims.

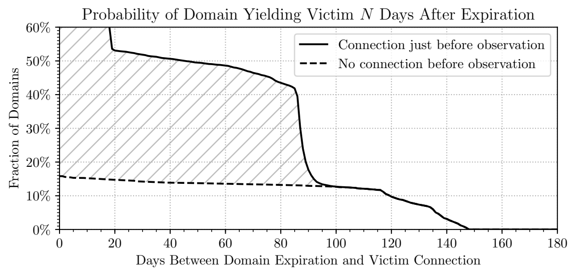


Figure 4: PDF showing the probability that a domain we sinkholed would yield a victim connection N days after its most recent registration by another party. 1,686 of the 6,897 domains we sinkholed had *no* resolution known to us and were excluded, leaving 5,211 domains (824 yielding victim connections).

In all, 975 domains received victims, 14% of the 6,897 we sinkholed. 1,686 of these domains had no known historic resolution from any of our sources, including threat intelligence feeds and our own resolver.

6.2 Attacker Campaign Analysis

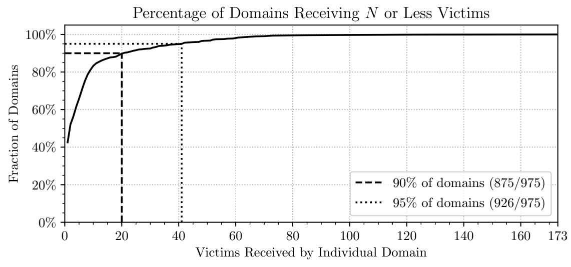


Figure 5: CDF showing the number of victims (by fingerprint) received by a given domain. This plot only includes the 975 domains which yielded victim connections.

Only 975 of the domains we sinkholed yielded victim connections, yet they received connections from 3,231 unique victims. In Figure 5, we examine the number of unique victims any one domain received. 43% of domains received only a single victim; 90% received at most 20 unique victims; 95%, 41 or less. Three outlier domains received over 100 victims. This disparity suggests that some attackers are distributing their malware more widely, or are more proficient at compromising their targets, than others.

We also find that 45% of victims connected to just one domain. 90% of victims connected to four or less different domains, while 95% connected to five or less.

These victims connecting to multiple different domains, and domains receiving tens of unique victim connections, suggest a number of phenomena. Attackers may be using sets of domains interchangeably, or victims may be infected by multiple different attackers.

To investigate the former, we examine the samples we downloaded from VirusTotal. Our 975 domains are found in 1,429 unique samples. Once again, there is bidirectional overlap. Only eight samples contain more than one domain in their configurations; 1,421 have a single domain. Multiple domains being in a single sample is our strongest indicator of them being shared by an attacker. Oppositely, 246 domains are found in more than one sample's configurations; the remaining 729 domains are each present in just one sample. Some domains are shared by many samples - one being found in 24 unique DarkComet samples. Unfortunately, these domains further muddy our analysis. In the case of the domain shared by 24 samples, only two samples clearly belong to the same actor (based on shared configuration parameters).

Our methodology cannot definitively answer whether attackers use rotating domains, or whether victims are multiply infected by different campaigns. Based on our evidence, both appear probable, and confound our attempts at differentiating attackers and their victims.

6.3 Geographic Analysis

All IP-based geolocations were performed using MaxMind's GeoIP2 Precision Insights service.

njRAT			DarkComet		
Country	Proxy	Other	Country	Proxy	Other
France (FR)	3,829	69	United States (US)	4,552	1,881
United States (US)	714	167	France (FR)	2,771	1,623
Sweden (SE)	433	19	Sweden (SE)	1,051	318
United Kingdom (GB)	160	63	Netherlands (NL)	706	256
Canada (CA)	152	12	Germany (DE)	511	3,077
Netherlands (NL)	96	9	United Kingdom (GB)	487	1,494
...			...		
Algeria (DZ)	22	7,820	Turkey (TR)	130	21,913
Brazil (BR)	42	7,206	Russia (RU)	233	17,020
Egypt (EG)	27	5,655	Algeria (DZ)	13	13,202
Morocco (MA)	3	4,293	Morocco (MA)	2	6,693
Iraq (IQ)	5	2,001	Egypt (EG)	4	4,872
Tunisia (TN)	0	1,504	Saudi Arabia (SA)	0	4,491
Saudi Arabia (SA)	0	1,297	Ukraine (UA)	75	3,971
Indonesia (ID)	8	732	Brazil (BR)	78	3,257
Libya (LY)	0	682	Pakistan (PK)	28	2,935
Other	524	6,113		1,921	36,919
Total	6,015	37,642		12,562	123,922

Table 11: Geolocations of historic controller IP addresses based on DNS history

Proxies. MaxMind provides information regarding the likelihood that an IP address is a proxy, as well as IP ownership (which can be used to manually determine proxies). We use this information to separate proxies from non-proxies, as in Tables 11 and 12. A large portion of the controllers in our data set appear to be utilizing proxies from certain countries like France, Sweden, and the U.S. We manually investigate the largest in Appendix A. In short, we find two VPN providers (IPjetable [24] and Relakks [42]) account for 40% and 3% of all actively-probed controllers, respectively, while prominent VPS services like Amazon AWS, Microsoft Azure, and Digital Ocean are also frequently abused.

njRAT			DarkComet		
Country	Proxy	Other	Country	Proxy	Other
France (FR)	2,625	4	France (FR)	258	41
Sweden (SE)	184	0	Sweden (SE)	16	0
United States (US)	16	2	United States (US)	12	6
...			...		
Brazil (BR)	2	441	Turkey (TR)	0	594
Morocco (MA)	0	382	Ivory Coast (CI)	0	207
Algeria (DZ)	0	281	Russia (RU)	11	201
Egypt (EG)	0	178	India (IN)	1	128
Korea (KR)	0	80	Thailand (TH)	0	102
Tunisia (TN)	0	65	Vietnam (VN)	0	88
Iraq (IQ)	0	58	Ukraine (UA)	8	63
Saudi Arabia (SA)	0	52	Egypt (EG)	1	41
Thailand (TH)	0	39	Azerbaijan (AZ)	0	37
Turkey (TR)	0	37	Malaysia (MY)	0	33
Other	17	121		35	156
Total	2,844	1,740		342	1,697

Table 12: Geolocations of probed controller IP addresses

As the geolocation results of the proxies only serve to muddle the geospatial relationships between victims and attackers, we filter them from the following analyses. We report only on those results in the *Other* columns of the geolocation tables.

Controller geography. Tables 11 and 12 show the geolocations of historic and actively-probed controller IP addresses, respectively. We find both to have heavy presences in North Africa and the Middle East. Outliers include Brazil and Russia, both of which tend to correspond with victims in bordering nations.

Victim geography. Exploring Table 13, we find that virtually every country has some RAT victims with Brazil being the top location for victims of both DarkComet and njRAT, as shown in Table 13. We find what appears to be geographic concentrations of DarkComet and njRAT victims in South America and North Africa / Middle East, including some bordering countries. We also find that DarkComet is used to infect a larger percentage of victims in Russia and bordering countries. Note that these measurements might be biased by our methodology of acquiring RAT samples and sinkholing DDNS domains.

Controller-victim geography: Recall that during the sinkholing portion of the experiment, we registered the command-and-control domain, directing all potential victims to our server. During this period, we were able to observe all victims that attempted to connect to the controller. Prior to the sinkholing period, controller domains may have been held by the original controller or may have been sinkholed by researchers or vigilantes. In addition, for four and a half months prior to the sinkholing experiment, we resolved all controller domains to de-

njRAT			DarkComet		
Country	#Src-IP	#FP	Country	#Src-IP	#FP
Brazil (BR)	2,416	1,070	Brazil (BR)	318	178
Egypt (EG)	331	94	Turkey (TR)	188	130
Iraq (IQ)	207	82	Russia (RU)	184	127
Argentina (AR)	138	62	Ukraine (UA)	44	38
Algeria (DZ)	149	60	Egypt (EG)	74	36
Peru (PE)	131	55	Poland (PL)	28	26
Vietnam (VN)	117	53	Philippines (PH)	22	21
United States (US)	54	47	Thailand (TH)	35	17
Venezuela (VE)	105	47	Vietnam (VN)	16	14
India (IN)	88	46	Algeria (DZ)	21	13
Turkey (TR)	93	40	Bosnia (BA)	17	13
Thailand (TH)	189	38	Indonesia (ID)	12	11
Mexico (MX)	66	37	India (IN)	11	10
Other	1,401	659		265	207
Total	5,485	2,390		1,235	841

Table 13: Geolocations of victim IP addresses

termine whether they were registered, and, if registered whether they had an associated A record, and whether the corresponding hosts behaved correctly (as a controller). Thus, for each domain, we have the IP addresses of all controllers that held the domain, as well as of all victims that attempted to connect to the domain during the sinkholding period. (Note that two periods are necessarily disjoint: both we and the original controller cannot hold the same domain at the same time.) Figure 6 shows the geographic relationship between responsive controllers and the victims, using the geolocation methodology above. Each cell of the matrix shows the number of distinct campaigns (domains) associated with the given country pair. In cases where a domain resolved to more than one country or where victims where located in more than one country, the domain contributed a fractional weight to each cell in proportion to the number of controller-victim pairs of the domain from the country pair, so that the total contribution of each domain was 1. Figure 6 shows only the top 25 countries, ordered by the greater of the number of victims and controllers in the country. The dominant feature of the data is the controller and victim being located in the same country, visible as a concentration around the diagonal in the matrix. In addition, there were 5 campaigns with a controller in Ukraine (UA) and victims in Russia (RU). This may be due to a common infection vector, as Ukraine has a large Russian-speaking population and its users may frequent the same Russian-language sites. The incidence of controllers and Russia and victims in Brazil (BR) is more puzzling; although both Russia and Brazil have large victim and controller populations, there is no obvious reason why controllers in Russia might target victims in Brazil specifically. Another possibility is that the controllers were using a proxy in Russia that was missed by our filtering.

Figure 7 shows the same type of data, but for all controllers using the historic controller dataset. Note that

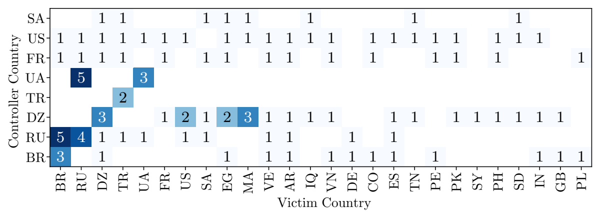


Figure 6: Relational matrix comparing geolocations of actively-probed controller IP addresses to received victim IP addresses, per sinkholed domain. Proxy IP addresses are filtered.

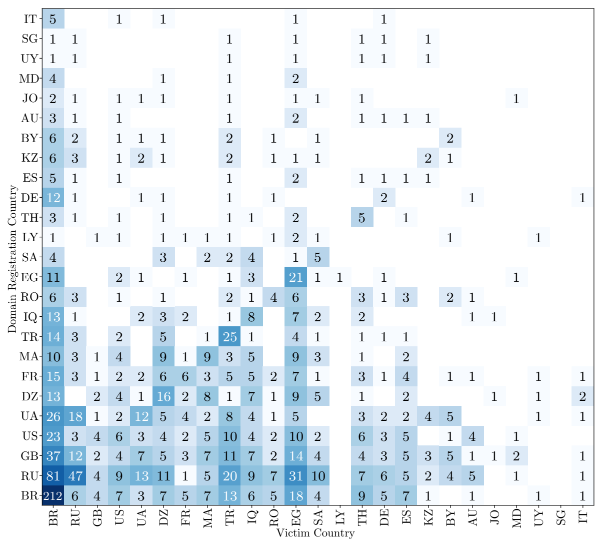


Figure 7: Relational matrix comparing geolocations of historic controller IP addresses to received victim IP addresses, per sinkholed domain. Proxy IP addresses are filtered.

this data spans the period 2010 to 2017 and includes name resolution from passive DNS sources (see Section 5.1), where we did not verify the correct behavior of the controller. As such, this data should be interpreted with caution. Figure 7 exhibits the same concentration around the diagonal as Figure 6, indicating campaigns where both controller and victim are in the same country. As the results of Table 13 suggest, Brazil has by far the largest concentration of victims across both RATs. Moreover, Brazil appears to be victimized indiscriminately. We also note some language clustering, where countries that speak the same language or are geographically proximate are more likely to be paired; e.g., Russia on Ukraine (13), Ukraine on Russia (18), Ukraine on Kazakhstan (4), Ukraine on Belarus (5), Morocco (MA) on Algeria (DZ) (9), Algeria on Morocco (8).

7 Discussion

7.1 Limitations

Adversarial robustness. Our classification methods that RAT-Hole uses to distinguish sandboxes, scanners, and victims are not robust to an adversarial actor that is actively trying to evade our classification. Based on our validation it appears that there are many detectable sandboxes and scanners. As future work, we will explore potentially more robust features, such as inter-arrival timing of connection attempts in order to detect scanners and analysis of active window patterns to detect sandboxes.

Manual reverse engineering. We have not developed a method for automatically decrypting RAT Protocols or parsing out information from fields in the protocol. This caused us to limit our analysis to two common RAT family types. It is unclear what biases might have been introduced into our results due to limiting the number of RAT families and DDNS services included in our study. As future work we will explore how well existing methods for automated protocol reverse-engineering [7,8] and decrypting of messages [47] perform at our task.

Family-specific classification heuristics. Our methodology for building up a set of classification heuristics was again a manual process and in some instances, such as triggering error conditions, was RAT family specific. As future work, we will explore more automated methods, such as semi-supervised machine learning based approaches using inter-arrival timing of connections to differentiate scanners from execution of the actual malware. We will also explore methods based on victim behavior to identify sandboxes. We hypothesize that it will be difficult for a sandbox to mimic the patterns of a real victim.

Validating scanning results. We have little ground truth to evaluate methods for distinguishing between legitimate RAT controllers and sinkhole operations, other than our own sinkhole. As future work we will explore additional methods of ethically probing controllers, such as calling rarely used API functions that are unlikely to be implemented by sinkholes.

7.2 Protecting Victims

Our results show that expired RAT domains still have likely victims attempting to connect to them. The 3,231 likely victims we detected could be further victimized by an adversary that claimed these domains. We are in the process of working with some free DDNS providers to understand if they would be willing to permanently block domain names associated with RAT controllers. We plan to pursue the same dialogue with the VPN proxy services that provide support infrastructure for RAT operators.

7.3 Notification Efforts

We received repeated manual notifications based on scanners misclassifying RAT-Hole as a large-scale RAT controller hosting operation. Our results suggest that more accurate identification of controllers and victims could reduce wasted notification effort.

8 Related Work

Our work is influenced heavily by research projects from industry and academia. We discuss works that informed our study's primary aspects: sinkholing and scanning.

Sinkholing and infection enumeration. A number of early botnet measurement studies mused on its challenges. A Trend Micro industry report from 2001 [29] qualitatively discussed the problems with sinkholing botnet domains, like receiving PII. The ethical issue of victim PII receipt is universal to infection enumeration efforts; Han *et al.* [23] built a system for sandboxing phishing kits explicitly designed to protect victim privacy.

Always prominent has been the issue of accurate infection size estimation. Ramachandran *et al.* [41] proposed a method of estimating botnet infection size based on frequency of DNS lookups to C&C domains. A subsequent pair of botnet size estimation studies used DNS lookups [9] and IRC channel monitoring [1], but arrived at different estimates due to errors caused by churn [40].

A number of studies explored how to estimate the size of the Storm botnet [16, 21, 38], while Stone-Gross *et al.* [49] actually sinkholed the Torpig botnet, live, and created unique fingerprints for each infection to address infection measurement difficulties, as do we in this study. A follow-up study by Kanich *et al.* [25] showed that pollution caused by interfering measurement operations had inflated the measured size of the botnet. Nadji *et al.* [34] discuss the same issue of measurement pollution while running a domain sinkhole performing botnet takedowns.

Novel approaches for detecting and filtering scanners exist. For instance, Rossow *et al.* [44] proposed a method for detecting sensors based on detecting crawlers injecting themselves into large numbers of points in a P2P network. Successful methods for detecting scanners tend to be highly tailored, as was ours.

Our methodology exploits the fact that DDNS domains used as C&C's will ultimately expire, though victims are still contacting them. This is one of the premises behind work by Starov *et al.* [48]; though they focus on web shells rather than more traditional RATs, their goal of measuring the ecosystem of attackers and victims is similar to ours. Lever *et al.* [28] measure the adversarial possibilities behind re-registering an expired domain.

Part of our methodology focuses on the challenge of detecting malware samples being executed in sandboxes, which we found to be a source of intelligence pollution. Most prior studies on sandbox detection focus on malware sandbox evasion techniques [4, 26, 33, 37, 45, 53]. A more recent study demonstrated that intentionally-designed binary submissions to antivirus companies can exfiltrate sandbox fingerprints [55]. Our approach furthers these efforts to identify Internet-connected sandboxes, using unmodified malware binary submissions and leveraging artifacts of the execution process like short execution duration to inform our detection.

In a 2014 report, researchers at FireEye enumerated infections for an XtremeRAT campaign by sinkholing the controller domain [51]. This study notes the challenges of victim IP address churn, which our work also encountered. We designed our methodology to explicitly handle the challenges this study uncovered, as well as to filter intelligence pollution from scanners and sandboxes, such that we could accurately and ethically enumerate RAT infections based on sinkhole data.

Scanning and controller discovery. BladeRunner [15] was the first scanning-based system to actively discover RAT controllers by emulating RAT victims. Since then, Shodan [31] has added active probing and banner identification for numerous RAT families including DarkComet and njRAT. Marczak *et al.* [30] created a scanner that was able to detect stealthy APT controllers by triggering error conditions. Most recently, Farinholt *et al.* [17] presented a scanner that used ZMap [14], Shodan, and a custom port scanner to detect DarkComet controllers based only on their initial handshake challenges. RAT-Scan's design is based on these systems.

RAT-Scan also contains logic to (attempt to) address the issue of sinkholes polluting controller measurements. The most closely related work is SinkMiner, a system which proposed a method to detect sinkholes based on historic DNS data [39]. Though SinkMiner uses passive DNS to detect sinkholed domains, its research goals - measuring the effective lifetime of a C&C domain and avoiding enumerating fellow security vendors' infratructure - matched ours. We consider RAT-Scan complimentary to SinkMiner in this regard.

9 Conclusion

We presented the results of our study of attacker and victim populations of two major RAT families, njRAT and DarkComet. One of the challenges of studying both operators (attackers) and victims is the noisy nature of the signal. To distinguish real operators and victims, we develop a set of techniques for testing the behavior of a suspected victim to determine if it is a genuine infection or not. Using a similar set of tests, we identify genuine controllers, excluding sinkholes and controllers using VPNs.

Using our collected data, we then report on the population of victims and controllers, their geographic relationship, and periods of activity. Our results show that the RATs we studied are used primarily by operators and victims located in the same country, with the bulk of the population in Russia, Brazil, and Turkey. We also found that victims remain vulnerable long after the controller abandons the campaign, presenting an opportunity for third-party intervention by sinkholing the domains.

Acknowledgments

This work was supported by the National Science Foundation through grants CNS-1237264, CNS-1619620, and CNS-1717062, and by gifts from Comcast, Farsight Security, and Google. We would also like to thank the following: VirusTotal, for the invaluable Intelligence account from which we sourced malware; Richard Harper of DuckDNS, for generous access to a Duck Max account; Matthew Jonkman of EmergingThreats, for generous access to an unlimited Threat Intelligence account; and finally, our reviewers, for their invaluable feedback.

References

[1] Abu Rajab, M., Zarfoss, J., Monrose, F., and Terzis, A. A multifaceted approach to understanding the botnet phenomenon. In *ACM Internet Measurement Conference (IMC)* (2006).

[2] AM523. How to create vpn for rat 2017. `https://www.youtube.com/watch?v=0KQQ0pM3dDU`.

[3] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., and Zhou, Y. Understanding the Mirai Botnet. In *USENIX Security Symposium (USENIX)* (2017).

[4] Bazhaniuk, O., Bulygin, Y., Furtak, A., Gorobets, M., Loucaides, J., and Shkatov, M. Reaching the far corners of MATRIX: generic VMM fingerprinting. *SOURCE Seattle* (2015).

[5] Breen, K. RAT Decoders. `https://techanarchy.net/2014/04/rat-decoders/`, April 2014.

[6] Breen, K. DarkComet – Hacking The Hacker. `https://techanarchy.net/2015/11/darkcomet-hacking-the-hacker/`, November 2015.

[7] Caballero, J., Yin, H., Liang, Z., and Song, D. Polyglot: Automatic Extraction of Protocol Message Format Using Dynamic Binary Analysis. In *ACM Conference on Computer and Communications Security (CCS)* (2007).

[8] Cui, W., Kannan, J., and Wang, H. J. Discoverer: Automatic Protocol Reverse Engineering from Network Traces. In *USENIX Security Symposium (USENIX)* (2007).

[9] Dagon, D., Zou, C., and Lee, W. Modeling Botnet Propagation Using Time Zones. In *Networked and Distributed System Security Symposium (NDSS)* (2006).

[10] Denbow, S., and Hertz, J. pest control: taming the rats. Tech. rep., 2012.

[11] Digital Citizens Alliance. SELLING "SLAVING" - Outing the principal enablers that profit from pushing malware and put your privacy at risk, July 2015.

[12] Dittrich, D., Leder, F., and Werner, T. A case study in ethical decision making regarding remote mitiga-

tion of botnets. In *International Conference on Financial Cryptography and Data Security* (2010).

[13] DURUMERIC, Z., ADRIAN, D., MIRIAN, A., BAILEY, M., AND HALDERMAN, J. A. A Search Engine Backed by Internet-Wide Scanning. In *ACM Conference on Computer and Communications Security (CCS)* (Oct. 2015).

[14] DURUMERIC, Z., WUSTROW, E., AND HALDERMAN, J. A. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *USENIX Security Symposium (USENIX)* (2013).

[15] EISENBARTH, M., AND JONES, J. BladeRunner: Adventures in Tracking Botnets. In *Botnet Fighting Conference (Botconf)* (2013).

[16] ENRIGHT, B., VOELKER, G. M., SAVAGE, S., KANICH, C., AND LEVCHENKO, K. Storm: When researchers collide. *USENIX ;login:* (2008).

[17] FARINHOLT, B., REZAEIRAD, M., PEARCE, P., DHARMDASANI, H., YIN, H., LE BLOND, S., MCCOY, D., AND LEVCHENKO, K. To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild. In *IEEE Symposium on Security and Privacy (S&P)* (2017).

[18] FARIVAR, C. Sextortionist who hacked miss teen usa's computer sentenced to 18 months, Mar 2014.

[19] Farsight Security. `https://farsightsecurity.com/`.

[20] FIDELIS. Fidelis Threat Advisory 1009: "njRAT" Uncovered, June 2013.

[21] GRIZZARD, J. B., SHARMA, V., NUNNERY, C., KANG, B. B., AND DAGON, D. Peer-to-peer botnets: Overview and case study. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)* (2007).

[22] GUNDERT, L. Proactive Threat Identification Neutralizes Remote Access Trojan Efficacy. `http://go.recordedfuture.com/hubfs/reports/threat-identification.pdf`, 2015.

[23] HAN, X., KHEIR, N., AND BALZAROTTI, D. Phisheye: Live monitoring of sandboxed phishing kits. In *ACM Conference on Computer and Communications Security (CCS)* (2016).

[24] ipjetable. `https://ipjetable.net/`.

[25] KANICH, C., LEVCHENKO, K., ENRIGHT, B., VOELKER, G. M., AND SAVAGE, S. The heisenbot uncertainty problem: Challenges in separating bots from chaff. In *USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)* (2008).

[26] KIRAT, D., VIGNA, G., AND KRUEGEL, C. Barecloud: Bare-metal analysis-based evasive malware detection. In *USENIX Security Symposium (USENIX)* (2014).

[27] LE BLOND, S., URITESC, A., GILBERT, C., CHUA, Z. L., SAXENA, P., AND KIRDA, E. A look at targeted attacks through the lense of an ngo. In *USENIX Security Symposium (USENIX)* (2014).

[28] LEVER, C., WALLS, R., NADJI, Y., DAGON, D., MCDANIEL, P., AND ANTONAKAKIS, M. Domain-z: 28 registrations later measuring the exploitation of residual trust in domains. In *IEEE Symposium on Security and Privacy (S&P)* (2016).

[29] LINK, R., AND SANCHO, D. Lessons learned while sinkholing botnets - not as easy as it looks! In *Virus Bulletin International Conference,(Barcelona)* (2001).

[30] MARCZAK, W. R., SCOTT-RAILTON, J., MARQUIS-BOIRE, M., AND PAXSON, V. When governments hack opponents: A look at actors and technology. In *USENIX Security Symposium (USENIX)* (2014).

[31] MATHERLY, J. Shodan - Malware Hunter. `https://malware-hunter.shodan.io/`.

[32] MATHERLY, J. Shodan - The search engine for the Internet of Things. `https://www.shodan.io/`.

[33] MIRAMIRKHANI, N., APPINI, M. P., NIKIFORAKIS, N., AND POLYCHRONAKIS, M. Spotless Sandboxes: Evading Malware Analysis Systems using Wear-and-Tear Artifacts. In *IEEE Symposium on Security and Privacy (S&P)* (2017).

[34] NADJI, Y., ANTONAKAKIS, M., PERDISCI, R., DAGON, D., AND LEE, W. Beheading hydras: performing effective botnet takedowns. In *ACM Conference on Computer and Communications Security (CCS)* (2013).

[35] No-IP. `https://www.noip.com/`.

[36] OPPLEMAN, V. Network Defense Applications using IP Sinkholes. hakin9.org.

[37] PETSAS, T., VOYATZIS, G., ATHANASOPOULOS, E., POLYCHRONAKIS, M., AND IOANNIDIS, S. Rage against the virtual machine: hindering dynamic analysis of android malware. In *European Workshop on System Security* (2014).

[38] PORRAS, P., SAIDI, H., AND YEGNESWARAN, V. A multi-perspective analysis of the storm (peacomm) worm. Tech. rep., Computer Science Laboratory, SRI International, 2007.

[39] RAHBARINIA, B., PERDISCI, R., ANTONAKAKIS, M., AND DAGON, D. SinkMiner: Mining Botnet Sinkholes for Fun and Profit. In *USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)* (2013).

[40] RAJAB, M. A., ZARFOSS, J., MONROSE, F., AND TERZIS, A. My Botnet is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)* (2007).

[41] RAMACHANDRAN, A., FEAMSTER, N., AND DAGON, D. Revealing Botnet Membership Using DNSBL Counter-intelligence. In *Steps to Reducing Unwanted Traffic on the Internet - Volume 2* (2006).

[42] relakks vpn. `https://www.relakks.com/`.

[43] RiskIQ PassiveTotal. `https://www.riskiq.com/`.

[44] ROSSOW, C., ANDRIESSE, D., WERNER, T., STONE-GROSS, B., PLOHMANN, D., DIETRICH, C. J., AND BOS, H. P2PWNED: Modeling and Evaluating the Resilience of Peer-to-Peer Botnets . In *IEEE Symposium on Security and Privacy (S&P)* (May 2013).

[45] RUTKOWSKA, J. Red pill: Detect VMM using (almost) one CPU instruction. `http://invisiblethings.org/papers/redpill.html`, 2004.

[46] SATTER, R., J DONN, E., AND VASILYEVA, N. Russian hackers hunted journalists in years-long campaign. *Associated Press* (Dec. 22, 2017).

[47] SONG, D., BRUMLEY, D., YIN, H., CABALLERO, J., JAGER, I., KANG, M. G., LIANG, Z., NEWSOME, J., POOSANKAM, P., AND SAXENA, P. BitBlaze: A New

Approach to Computer Security via Binary Analysis. In *International Conference on Information Systems Security* (2008).

[48] STAROV, O., DAHSE, J., AHMAD, S. S., HOLZ, T., AND NIKIFORAKIS, N. No honor among thieves: A large-scale analysis of malicious web shells. In *International Conference on World Wide Web (WWW)* (2016).

[49] STONE-GROSS, B., COVA, M., CAVALLARO, L., GILBERT, B., SZYDLOWSKI, M., KEMMERER, R., KRUEGEL, C., AND VIGNA, G. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *ACM Conference on Computer and Communications Security (CCS)* (2009).

[50] TUTORIALS ON HACK. Example: How To open port windows 10 Free vpn {Tutorials android hack}. `https://www.youtube.com/watch?v=N1mhRUCuyF4`.

[51] VILLENEUVE, N., AND BENNETT, J. XtremeRAT: Nuisance or Threat. `https://www.fireeye.com/blog/threat-research/2014/02/xtremerat-nuisance-or-threat.html`, 2014.

[52] Virustotal malware intelligence service. `https://www.virustotal.com/#/intelligence-overview`.

[53] WILLEMS, C., HUND, R., FOBIAN, A., FELSCH, D., HOLZ, T., AND VASUDEVAN, A. Down to the bare metal: Using processor features for binary analysis. In *Annual Computer Security Applications Conference* (2012).

[54] YARA: The pattern matching swiss knife for malware researchers (and everyone else). `http://virustotal.github.io/yara/`.

[55] YOKOYAMA, A., ISHII, K., TANABE, R., PAPA, Y., YOSHIOKA, K., MATSUMOTO, T., KASAMA, T., INOUE, D., BRENGEL, M., BACKES, M., AND ROSSOW, C. SandPrint: Fingerprinting Malware Sandboxes to Provide Intelligence for Sandbox Evasion. In *Research in Attacks, Intrusions, and Defenses* (2016).

A VPN/VPS Provider Abuse

IPjetable VPN: 141.255.144/20. Of the 6,401 IP addresses RAT-Scan successfully probed, a full 2,635 (or 40.2%) came from this address space. Further, these IP addresses accounted for over 40% of *all* connections made during the six months of active scanner operation, exhibiting abnormal longevity compared to other controllers. This space is owned by IPjetable [24], a French company that provides free VPN services and that is recommended by hundreds of RAT instruction videos available online [2]. IP addresses belonging to IPjetable are even present in the Recorded Future IoC dataset from 2015.

Relakks VPN: 93.182.168/21. Though not nearly as large as the IPJetable address space, this space contained 167 IP addresses probed by RAT-Scan (2.6% of all IP addresses), accounting for nearly 2% of all RAT-Scan connections. This address space belongs to Relakks VPN [42], a Swedish company that provides free VPN services and is likewise recommended by RAT instruction videos [50] and HackForums members.

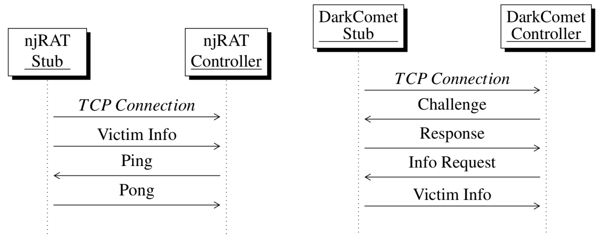
VPS providers. In addition to using VPN's, we found the use of VPS instances from prominent services like Amazon AWS, Microsoft Azure, and Digital Ocean, as well as less reputable providers like OVH.

B Telescope Data

Peer Type	Overlapping /32	% Overlap
Victim	5	<0.1
Sandbox	8	<0.1
LF Scanner	38	<0.1
HF Scanner	1	<0.1
Unknown	31,014	3.8
Total	828,137	100.0

Table 14: Breakdown of the Src-IPs (/32) of our defined peer types that overlap with IP addresses from our telescope dataset.

Table 14 compares RAT-Hole's connection dataset with a prominent network telescope's connection dataset, showing the overlapping connecting source IP addresses. Of import is the *lack* of overlap between the datasets. This refutes our initial hypothesis that network telescope data could be used to filter most indiscriminate scanning operations from a sinkhole's dataset.



(a) (b)

Figure 8: The standard handshake protocol for (a): njRAT (b): DarkComet

C njRAT Network Protocol

njRAT speaks a custom application-layer network protocol over TCP. In Figure 8a, we provide a diagram of the njRAT handshake, the initial exchange of messages between the stub and controller before the command-response cycle begins. We note that, as njRAT is a victim-initiated RAT, its stub sends the first payload of the handshake after establishing a TCP connection to the controller.

njRAT has many variants (or sub-species). The three most commonly found in the wild are `main` (the original version), KilerRAT, and Coringa-RAT. We reverse engineered the protocols of each of these three variants, and as such RAT-Hole fully supports connections from all three.

Each of the messages exchanged in the njRAT handshake from Figure 8a is now further detailed individually.

1. *njRAT Victim Info Message* (*Basic*) †

```
<NI><NS><B><NS><PC_NAME><NS><PC_USERNAME><NS>
<INSTALL_DATE><NS><COUNTRY><NS><OS><NS>
<WEBCAM_FLAG><NS><VERSION><NS><ACTIVE_WINDOW>
<NS><NS><NT>
```

† An example of this message is provided in Figure 9.

1. *njRAT Victim Info Message* (*Extended*)

Victim Info Message (*Basic*) || `<INF_NI><INFO><NS><NT>`

```
00000000  6c 76 7c 27 7c 27 7c 53  47 46 6a 53 32 56 6b 58  |lv|'|'|SGFjS2VkX| # <NI>: lv
00000010  7a 68 46 4d 54 46 43 51  55 4d 34 7c 27 7c 27 7c  |zhFMTFCQUM4|'|'|| # <NS>: |'|'|
00000020  74 65 73 74 2d 50 43 7c  27 7c 27 7c 61 64 6d 69  |test-PC|'|'|admi| # <B>: SGFjS2VkXz... -> base64(HacKed_8E11BAC8)
00000030  6e 7c 27 7c 27 7c 32 30  31 35 2d 30 35 2d 31 32  |n|'|'|2015-05-12| # <CAMPAIGN_ID>: HacKed
00000040  7c 27 7c 27 7c 7c 27 7c  27 7c 57 69 6e 20 37 20  ||'|'||'|'|Win 7 | # <VSN>: 8E11BAC8
00000050  50 72 6f 66 65 73 73 69  6f 6e 6e 65 6c 20 53 50  |Professionnel SP| # <PC_NAME>: test-PC
00000060  31 20 78 36 34 7c 27 7c  27 7c 4e 6f 7c 27 7c 27  |1 x64|'|'|No|'|'| # <PC_USERNAME>: admin
00000070  7c 30 2e 36 2e 34 7c 27  7c 27 7c 2e 2e 7c 27 7c  ||0.6.4|'|'|..|'|| # <INSTALL_DATE>: 2015-05-12
00000080  27 7c 53 57 35 7a 64 47  46 73 62 43 42 68 62 6d  |'|SW5zdGFsbCBhbm| # <OS>: Win 7 Professionnel SP1 x64
00000090  51 67 64 58 4e 6c 49 47  35 71 55 6b 46 55 49 45  |QgdXNlIG5qUkFUIE| # <WEBCAM_FLAG>: No
000000a0  5a 56 52 43 42 30 62 79  42 6f 59 57 4e 72 49 46  |ZVRCB0byBoYWNrIF| # <RAT_VERSION>: 0.6.4
000000b0  42 44 49 43 30 67 57 57  39 31 56 48 56 69 5a 53  |BDIC0gWW91VHViZS| # <ACTIVE_WINDOW>: SW5zdGFsbCBhbmQgdXNlIG5qU...
000000c0  41 74 49 45 64 76 62 32  64 73 5a 53 42 44 61 48  |AtIEdvb2dsZSBDaH| # -> base64(Install and use njRAT FUD to hack
000000d0  4a 76 62 57 55 3d 7c 27  7c 27 7c 5b 65 6e 64 6f  |JvbWU=|'|'|[endo| #           PC - YouTube - Google Chrome)
000000e0  66 5d                                             |f]|               # <NT>: [endof]
```

Figure 9: An example njRAT *Victim Info* packet with individual components extracted, labelled, and decoded in the case of base64 encodings. Note that this is **not** from a real infection.

Field	Description
`<NI>`	Payload header. `lv`, `llv`, `lvv`, and `<SIZE><NUL>ll` are used by different njRAT versions.
`<SIZE>`	Number of bytes in message.
`<NS>`	Delimiter. The default is `|'|'|`, but KilerRAT and Coringa-RAT use `|Kiler|` and `|Coringa|`.
`<B>`	`<CAMPAIGN_ID>_<VSN>`, base64-encoded.
`<CAMPAIGN_ID>`	Identifier set by the operator, used to distinguish attack campaigns.
`<VSN>`	Volume serial number, victim's hard drive serial number.
`<PC_NAME>`	Victim PC name.
`<INSTALL_DATE>`	Date malware infected victim.
`<COUNTRY>`	Geolocation of victim IP address.
`<OS>`	Victim operating system.
`<WEBCAM_FLAG>`	Set if victim has webcam.
`<VERSION>`	Malware version.
`<ACTIVE_WINDOW>`	Victim's active window, base64-encoded.
`<NT>`	Payload end. `[endof]`, `llv`, `<NUL>` used by different versions.
`<INF_NI>`	Information payload header. `inf` and `<SIZE><NUL>inf` are used by different njRAT versions. `<CAMPAIGN_ID>`, port, C&C domain or IP, installation directory, binary name, registry flag, and startup flag.

Table 15: Descriptions of the fields in the njRAT handshake.

2. *njRAT Ping Message*

`P[endof]` or `0<NUL>`

3. *njRAT Pong Message*

`P[endof]` or `0<NUL><SIZE><NS><ACTIVE_WINDOW>`

While reversing the njRAT protocol, we uncovered a set of unique behaviors, some of which we used in differentiating between real njRAT victims and imitating scanners.

1. The njRAT stub can send either the *Basic* or the *Extended* version of the *Victim Info* message upon connection. Normally, the stub will send the *Basic* message the first time it contacts a controller, indicating that the stub likely maintains some state regarding past connections.
2. The *Extended Victim Info* message may be followed by multiple *Pong* messages, each containing the victim's active window. This appears to happen when the victim is physically present and interacting with applications at the immediate time of infection (and connection to the controller), prompting the stub to report active window changes in real-time.
3. The *Capture Command* is a command sent by the controller to the stub to request a screenshot. We found that a malformed *Capture Command* is not executed by the stub (as it fails out of the stub's command parser routine), but that instead the stub replies with a defined error response. This fringe behavior was useful in filtering real njRAT stubs from impersonators.

D DarkComet Network Protocol

DarkComet speaks a custom application-layer network protocol over TCP. In Figure 8b, we provide a diagram of the DarkComet handshake. As DarkComet is a controller-initiated RAT, the controller sends the first payload after the stub establishes a TCP to it. In the case of DarkComet, in the first exchange the controller challenges the stub, after which it obtains information about the stub's host. We now detail the individual messages from Figure 8b further.

1. *DarkComet Challenge Message*

`IDTYPE`

2. *DarkComet Response Message*

`SERVER`

3. *DarkComet Info Request Message*

`GetSIN<WAN_IP>|<NONCE>`

4. *DarkComet Victim Info Message* †

```
infoes<CAMPAIGN_ID>|<WAN_IP>/[<LAN_IP>]:<PORT>|
<PC_NAME>/<USERNAME>|<NONCE>|<PING>|<OS>
[<BUILD>]<BIT>bit( <PATH> )|<ADMIN_FLAG>|
<WEBCAM_FLAG>|<COUNTRY>|<ACTIVE_WINDOW>|<HWID>|
<RAM_USAGE>|<LANGUAGE>/ -- |<INSTALL_DATE>|
<VERSION>
```

† An example of this message is provided in Figure 10.

Field	Description
`<WAN_IP>`	Victim machine's public IP address.
`<NONCE>`	Six random digits.
`<LAN_IP>`	Victim machine's LAN IP address.
`<PORT>`	Controller's DarkComet port.
`<USERNAME>`	Victim's user name.
`<NONCE>`	Same nonce sent in *Info Request*.
`<PING>`	Stub response time, in seconds.
`<BUILD>`	Victim machine OS build version.
`<BIT>`	Victim machine architecture (e.g. `64`).
`<PATH>`	Path to drive where OS is stored on victim machine.
`<ADMIN_FLAG>`	Set if stub is running as admin.
`<HWID>`	Victim machine's `UUID`‡ and `VSN`.
`<RAM_USAGE>`	RAM in use on victim machine.
`<LANGUAGE>`	Victim machine's primary language.

Table 16: Descriptions of the fields in the DarkComet handshake. Note that fields from Table 15 are not duplicated. ‡ Universally Unique Identifier.

```
00000000  69 6e 66 6f 65 73 43 72  61 63 6b 65 64 50 68 6f  |infoesCrackedPho| # <CAMPAIGN_ID>: CrackedPhotoshopSeeding
00000010  74 6f 73 68 6f 70 53 65  65 64 69 6e 67 7c 33 32  |toshopSeeding|32| # <WAN_IP>: 32.245.251.132
00000020  2e 32 34 35 2e 32 35 31  2e 31 33 32 20 2f 20 5b  |.245.251.132 / [| # <LAN_IP>: 192.168.53.71
00000030  31 39 32 2e 31 36 38 2e  35 33 2e 37 31 5d 20 3a  |192.168.53.71] :| # <PORT>: 1604
00000040  20 31 36 30 34 7c 41 43  43 4f 55 4e 54 49 4e 47  | 1604|ACCOUNTING| # <PC_NAME>: ACCOUNTING-ADMIN-PC
00000050  2d 41 44 4d 49 4e 2d 50  43 20 2f 20 41 64 6d 69  |-ADMIN-PC / Admi| # <USERNAME>: Administrator
00000060  6e 69 73 74 72 61 74 6f  72 7c 37 36 39 37 33 34  |nistrator|769734| # <NONCE>: 769734
00000070  7c 30 73 7c 57 69 6e 64  6f 77 73 20 58 50 20 53  ||0s|Windows XP S| # <PING>: 0s
00000080  65 72 76 69 63 65 20 50  61 63 6b 20 33 20 5b 32  |ervice Pack 3 [2| # <OS>: Windows XP Service Pack 3
00000090  36 30 30 5d 20 33 32 20  62 69 74 20 28 20 43 3a  |600] 32 bit ( C:| # <BUILD>: 2600
000000a0  5c 5c 20 29 7c 78 7c 7c  55 4b 7c 51 75 61 72 74  |\\ )|x||UK|Quart| # <BIT>: 32
000000b0  65 72 6c 79 20 46 69 6e  61 6e 63 69 61 6c 20 52  |erly Financial R| # <PATH>: C:\\
000000c0  65 70 6f 72 74 20 44 52  41 46 54 20 28 43 6f 6e  |eport DRAFT (Con| # <ADMIN_FLAG>: x
000000d0  66 69 64 65 6e 74 69 61  6c 29 20 2d 20 4d 69 63  |fidential) - Mic| # <WEBCAM_FLAG>:
000000e0  72 6f 73 6f 66 74 20 45  78 63 65 6c 7c 7b 58 58  |rosoft Excel|{XX| # <COUNTRY>: UK
000000f0  58 58 58 58 58 58 2d 58  58 58 58 2d 58 58 58 58  |XXXXXX-XXXX-XXXX| # <ACTIVE_WINDOW>: Quarterly Financial Report
00000100  2d 58 58 58 58 2d 58 58  58 58 58 58 58 58 58 58  |-XXXX-XXXXXXXXXX| #      DRAFT (Confidential) - Microsoft Excel
00000110  58 58 7d 7c 38 33 25 7c  45 4e 47 4c 49 53 48 20  |XX}|83%|ENGLISH | # <HWID>: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
00000120  28 55 4e 49 54 45 44 20  4b 49 4e 47 44 4f 4d 29  |(UNITED KINGDOM)| # <RAM_USAGE>: 83%
00000130  20 47 42 20 2f 20 20 2d  2d 20 7c 30 36 2f 30 31  | GB /  -- |06/01| # <LANGUAGE>: ENGLISH (UNITED KINGDOM) GB
00000140  2f 32 30 31 38 20 41 54  20 30 37 3a 32 33 3a 33  |/2018 AT 07:23:3| # <INSTALL_DATE>: 06/01/2018 AT 07:23:34 PM
00000150  34 20 50 4d 7c 35 2e 33  2e 30                    |4 PM|5.3.0|       # <VERSION>: 5.3.0
```

Figure 10: An example DarkComet *Victim Info* packet with individual components extracted and labelled. Not from a real infection.

Authenticity in this handhshake consists of the stub having the shared RC4 password as well as knowing the correct response to the challenge. All handshake messages are RC4-encrypted with an operator-set key in the stub's configuration.

We discovered the following set of unique behaviors:

1. The stub only attempts up to 124 connections to a controller, provided the controller offers an unexpected challenge banner (e.g. has the wrong RC4 key). This means that if a sinkhole tries to brute force a stub's password, it can only try 124 times to get the correct challenge. After that, the stub will suspend until it is either executed again or the victim machine is rebooted.
2. However, we found that multiple challenges can be attempted during a single TCP connection. The stub's TCP buffer is 16,384 bytes. The stub will accept multiple challenges in a single message if they are joined by `\n\r` followed by 1,024 `0`'s, and will scan the entire message for the correct challenge. As such, trying more than 124 banners is possible, though brute-forcing the 12-character hexadecimal challenge is unrealistic.
3. The stub sends keepalive messages (`KEEPALIVE` encrypted in versions 4.0+, or `#KEEPALIVE#` in plaintext in older versions) during the handshake. However, DarkComet 5.2 never sends keepalives during the handshake.
4. Versions prior to DarkComet 4.0 do not use encryption.
5. Versions prior to DarkComet 4.0 reorder *Victim Info*.

E Other RATs

Peer Type	# Src-IP	% Src-IP
Black WorM	1	<0.1
LuminosityLink	8	<0.1
Xtreme RAT	226	<0.1
NetWire	575	0.1
H-W0rm	653	0.1
Unidentified	256,764	31.5
Passive	600,345	73.6
Total	815,455	100.0

Table 17: "Unknown" connections' detected peer types.

In Table 17, we present the detected peer types of those connections considered "Unknown." We received connections from several other RAT families, though we cannot distinguish between legitimate victims and impersonators. The small degree of overlap in the table indicates that some IP addresses presented multiple behaviors, possibly due to multiple-RAT infections, NAT'ed peers, or multiple scan behaviors.

		DarkComet FPs	njRAT FPs		
Sandbox Service	Conn.	Sandbox	Sandbox	HF Scanner	Victim
Avast	-	-	-	-	-
Avira	-	-	-	-	-
Bitdefender	-	-	-	-	-
Comodo	-	-	-	-	-
F-Secure	-	-	-	-	-
Fortiguard	✓	1	3	-	-
HybridAnalysis	✓	42	25	2	1
Intezer Analyze	-	-	-	-	-
JOESandbox	✓	4	4	-	-
Kaspersky	✓	-	3	-	-
Metadefender	-	-	-	-	-
Microsoft	✓	2	2	4	1
sandbox.pikker	✓	2	1	-	-
SONICWALL	-	-	-	-	-
Symantec	-	-	-	-	-
ThreatExpert	✓	1	-	-	-
ThreatTrack (CWsandbox)	-	-	-	-	-
TotalHash	✓	1	-	-	-
Valkyrie Comodo	-	-	-	-	-
ViCheck	-	-	-	-	-
VirusTotal	✓	13	17	-	-
Total	9	66	55	6	2

Table 18: Public sandbox services to which we submitted honey-samples, as well as detected connections from said services and their automatic classifications by RAT-Hole.

F Internet-Connected Sandboxes

We submitted honey-samples to the 21 public sandbox services in Table 18. Services were chosen based on their popularity among malware researchers, as well as their ease of use and cost. We only detected connections to RAT-Hole from nine of the services, indicating that their sandboxes are Internet-connected and that they did execute our honey-samples. While we detected the majority of connections correctly as sandboxes, a handful of njRAT connections were classified as HF Scanners or Victims. We strongly believe that these classifications are correct, and that some services either scanned RAT-Hole based on configurations extracted from our samples (the HF Scanners) or executed the samples in a non-automated analysis environment (the Victims).