A Model for Covert Botnet Communication in a Private Subnet

Brandon Shirley and Chad D. Mano

Department of Computer Science
Utah State University
Logan, UT 84322, USA
{brandon.shirley, chad.mano}@usu.edu

Abstract. Recently, botnets utilizing peer-to-peer style communication infrastructures have been discovered, requiring new approaches to detection and monitoring techniques. Current detection methods analyze network communication patterns, identifying systems that may have been recruited into the botnet. This paper presents a localized botnet communication model that enables a portion of compromised systems to hide from such detection techniques without a potentially significant increase in network monitoring points. By organizing bot systems at the the subnet level the amount of communication with the outside network is greatly reduced, requiring switch-level monitoring to identify infected systems.

1 Introduction

One of the most potent threats in the Internet society is the botnet [4, 9, 15]. A botnet is a collection of compromised computer systems throughout the world which are under the control of a single entity, the "botmaster." The distributed nature of a botnet may enable an attacker to gather larger amounts of confidential personal information over a shorter period, or execute a denial-of-service attack in a quick and overpowering manner.

Botnets have traditionally taken a centralized approach to the management of the compromised bot systems. Recently, a powerful botnet that utilizes peer-to-peer style communication, Storm Worm, has been the cause of various incidents of malicious activity. Tracking the origin and activity of this botnet is much more difficult due to the Peer-to-Peer (P2P) communication infrastructure.

2 Related Work

Traditionally botnets have relied on a centralized command and control (C&C) communication infrastructure utilizing IRC servers as a means to manage the remote bot systems [3, 7, 12]. A security administrator is able to monitor these systems by identifying the location of the C&C IRC server and logging in posing as a compromised bot system [1, 2]. Depending on the configuration of the server, various characteristics of the botnet can be identified including population and command instructions. Honeypots are systems deployed by security administrators that act as infection targets and allow administrators to detect

botnet activity [4, 10, 18]. Various techniques for evading detection exist, which hide or masquerade botnet communication activity [6, 9, 16].

More recently, botnets utilizing a peer-to-peer (P2P) [13, 17] style communication infrastructure have been proposed [7, 16] and even discovered in the wild [11]. It is very difficult to ascertain the characteristics of a P2P botnet because it is not possible to monitor a centralized location where all infected bots connect as such a location does not exist.

To the best knowledge of the authors', Bothunter [8] is the most effective tool for detecting the existence of a bot within a local network, including bots utilizing P2P style communication. The following section discusses Bothunter in more detail and discusses our approach to evading detection by Bothunter.

3 Evading BotHunter Detection

BotHunter relies on the ability to identify an infection dialog in order to detect the existence of a bot within a managed network. This paper proposes a communication method that negates the effective detection techniques of traffic monitoring systems by minimizing the amount, and controlling the type, of traffic that passes through a network monitor node, while at the same time enabling infected systems to receive all appropriate commands and instructions from the botmaster.

Bothunter is centered on the ability to detect five distinct events:

- E1: External to Internal Inbound Scans
- E2: External to Internal Inbound Exploits
- E3: Internal to External Binary Acquisition
- E4: Internal to External C&C Communication
- E5: Internal to External Outbound Infection Scanning

In order to declare a bot infection an E2 AND one E2-E5 events, or two E2-E5 events must be observed. Thus, to evade detection a bot must be able to actively participate in the global botnet while eliminating events which may lead to its detection. As such, we propose the following alternatives to the previously described events that accomplish the same purpose, but in a different manner.

- A2: Internal to Internal Exploits
- A3: Internal to Internal Binary Acquisition
- A4: Internal to Internal C&C Communication

A key factor in IDS techniques is that event correlation is tied to a single interior host. As the following section shows, infected systems can coordinate with each other to share the burden or communication with the external botnet parties, thus eliminating the bond between a single system and all external communication events.

4 Local Network "Sub-Botnet"

This section describes the process of creating and managing a local area network "sub-botnet." The creation process results in a coordinated group of bots in a switched portion of a network. The following are important assumptions about the network environment. Note that these assumptions may not be true of all networks, but we believe are common in typical enterprise networks.

– Communication within a switched network does not penetrate the router, other than defined protocols such as DHCP proxy.
– IDS-type monitors do not process information that does not move from the internal to external interface, or vice-versa, of the router.
– Switches utilized as network monitoring points do not have (or are not used for) IDS capabilities (typically used for monitoring bandwidth).

4.1 Initial Infection

A system can become infected by a malicious bot program in a number of ways, each ending with the same result of the system becoming part of a botnet. Initially, at least one system in the network must become infected and obtain the botnet binary in some way. We assume that this initial infection avoids detection by some means such as proposed in [8]. Once the initial bot has been created, the organization of the sub-botnet can begin.

The first order of business for the initial bot is to compromise local network systems, minimizing the chance of external E2 events. First, the initial bot checks its host system for vulnerabilities and assumes other local systems have the same vulnerability. Thus, the initial bot can attempt targeted exploits at other systems, without using noisy port scanning activities.

Next, the initial bot monitors network traffic to identify the IP and MAC addresses of systems within the locally switched network. DHCP requests, ARP messages, and other broadcast traffic enables the initial bot to identify many of these systems without using port-scanning activities.

After determining probable valid exploits and potential targets, the initial bot can proceed to create a sub-botnet by attempting to compromise other systems, an A2 event. When a system is compromised, vulnerabilities can be patched, preventing future E2 events.

4.2 New E2 Discovery

While A2 infections are desirable over E2 infections, it is not feasible to infect all possible systems with A2 events alone. When an E2 infection occurs, the newly infected system can better protect itself from being detected if it avoids unnecessary E3-E5 events. To do so, the newly infected system must discover if another system in the local network is part of the botnet. If such a bot exists, the newly infected system can obtain the bot binary and other communication from systems within the local network. The newly infected bot relies on bots already infected to provide the necessary information.

While the initial bot monitors the network to identify potential victims systems, it also seeks to identify newly infected systems. This can we accomplished in a number of ways including identifying a pre-determined pattern of DHCP Request messages [5]. This is enabled by an error prevention mechanism in the DHCP protocol. When a system issues a DHCP request it can repeat the request approximately four seconds later [5] if a response is not received. As this part of the protocol does not occur commonly, it can act as an “announcement” of a newly infected system. Upon observing this duplicate request sequence, the initial bot identifies the MAC address of the source machine and can direct a message to the newly infected system “welcoming” it to the botnet.

5 Sub-Botnet Management

As mentioned in Section 3 there are certain event dialogs which can be detected as a bot infection by BotHunter. In order to avoid detection the events that each bot may initiate must be controlled. To illustrate this point let $(A2 \vee A3 \vee A4) \equiv \alpha$, $(E3 \wedge E4) \equiv \beta$, and $(E2 \wedge (E3 \vee E4)) \equiv \gamma$. It follows that $\alpha \wedge \neg\beta \wedge \neg\gamma \equiv \alpha \wedge \neg(\beta \vee \gamma) \equiv \delta$ where δ is an undetectable state and that $\beta \vee \gamma \equiv \eta$ where η is a detectable state. Therefore, it is necessary to control bot actions such that only δ occurs within a given timespan as BotHunter or any IDS Correlator will have to eventually prune old events from each internal host's infection dialog.

The core goal of the management system is to limit the botnet's monitorable exposure to a single bot at any given time. This offers the potential to avoid detection by a perimeter-style IDS Correlator by exceeding the correlator's pruning threshold or by event sharing. Since the IDS Correlator must eventually prune old events, a round-robin type duty passing algorithm has the potential to allow external botnet communication without any of the bots every being detected.

To this end the sub-botnet will make use of a Token Bot (TB), which will perform the internal to external actions, and an internal report peer list to share traceable events amongst all the bots in the the subnet. In the pursuit of detection avoidance the Sub-Botnet management scheme must assure that each bot only performs one internal to external (E2-E4) event within a given timespan. Further, it is desirable that each bot performs the same E3 or E4 event each time it initiates said event and that only one bot is engaged in such an event at any given time.

In order to ensure this scenario the sub-botnet institutes a token passing scheme, where only the TB may initiate internal to external bot communication. Assuming there are two or more bots present in the sub-botnet, a token passing and resource sharing framework can be used. The basic token passing process is as follows: 1) Token Acquisition (TAQ), 2) perform Token Action (TAC) known as an E3 or E4 event, 3) issue Token Report Request Broadcast (TRRB) and compile token, 4) start Token Action Result Propagation (TARP) qualifying as an A3 or A4 event, and 5) the Token Pass. If there are any issues which prevent the TB from performing all these actions, then a Token Election (TE) will take place to recover the token. The TB will also acknowledge new bots infected via E2 infection and send them the bot binary. As far as A2 events are concerned the bot that initiated the A2 event will be charged with handling the A3 event for the newly infected bot. Full details of the token passing model are available in [14]

5.1 Token Acquisition (TAQ)

In this phase a bot will have the token passed to it through some variation of what will be referred to as a Token Pass (TP) process. The TP may have occurred explicitly due to TB failure or implicitly as part of the normal TP procedure. The token includes the following:

- Report list - internal only peer list
- Action History - the last two internal to external actions performed (E3 or E4 events)

– Timestamp - indicative of when token was compiled
– Bot Binary Version - denotes the current binary version

The Report list is the primary tool used for token passing and token election should token passing fail. The TAC for a given bot is implied by the information in the Report list. The new TB is listed in the Report list along with the action to be performed. Ideally this will be the same action it performed last time, or it may be a different action if bot action restructuring has occurred. Either way, the action listed should match with the action missing from the two-entry history list.

5.2 Perform Token Action (TAC)

The TB will perform the action designated by the Report list; either a command update request, a peerlist update, or a binary update check. The peerlist update will always be performed, but the other two actions may result in no update. The peerlist update will be largely handled by an external peer on the TB's peerlist. Assuming the TB finds a responsive peer, the TB will have that peer get an updated peerlist and send it back so the TB can propagate the list to the subnet. The command update request and binary update check will essentially work the same for the TB as they would for an external peer, except that the TB must propagate the result of its action to the subnet.

5.3 Token Report Request Broadcast (TRRB)

Upon completion of the TAC the TB will issue a TRRB to which any viable local bot will reply, as shown in Figure 1. Each bot will reply with its last performed action, a timestamp indicative of when that action was performed, and its bot binary version. The TB will compile and update the action history and report lists, and associate a timestamp with the update.

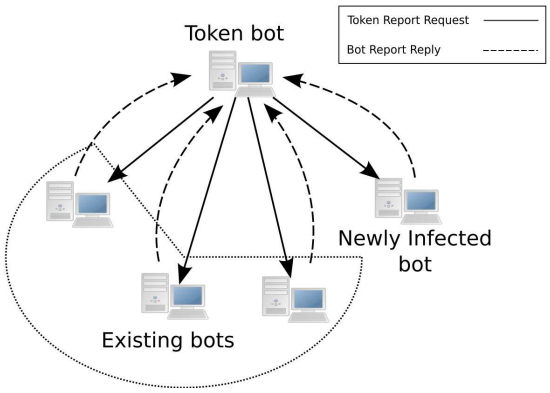

Fig. 1. Token bot issuing TRRB, and the subsequent responses from the currently active bots in its subnet.

5.4 Token Action Result Propagation (TARP)

Assuming a successful TRRB, the TB will have an updated report list which includes all currently active bots in its subnet. The TB will use the Report list as the means for passing the update information to the sub-botnet. The TB will send the Report list, the update, the current Bot Binary version and the action history to a bot in the report list. In essence it is passing a copy of the token, as shown in Figure 2.

5.5 Token Pass

The TB will use the information in the token to decide which bot will receive the token next. It will consult the action history in order to see which action needs to be performed next. At this time the TB will scan the report list looking for bots whose last action matches the next necessary action. Once the TB has a list of these bots it will select the bot with the oldest timestamp and pass the token to that bot.

5.6 Token Election (TE)

The TE request is a catch-all method for handling any token action failure. The TE request will generally occur after some set waiting period has expired without a response from the TB, therefore, it is possible that multiple bots will request a TE within the same timespan. A bot that issues the TE request will wait for responses for a set time period and if no other TE requests are received it issues a broadcast that TE was successful.

The bot that issues the TE success broadcast will use the history list and the most up-to-date report list that it received to pick a new TB. The method for choosing a new TB will be essentially the same as the method used for passing the token. Once the bot finds a suitable responsive bot it will pass the token to that bot. At this point the new bot will make the TAQ broadcast and the sub-botnet may resume normal activity. Details on handling simultaneous TE requests is presented in [14]

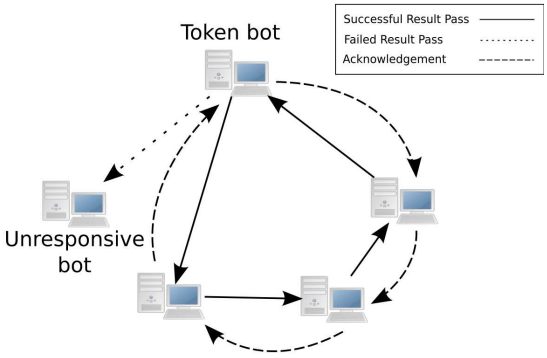

Fig. 2. Token bot passing a "copy" of the token as part of TARP.

6 Analysis

While the proposed system is able to effectively evade detection from traditional IDS and dialog-based botnet detection systems, it is not immune from other defense mechanisms. An increased level of monitoring and traffic analysis would enable this system to be detected.

As with traditional worm, virus, and botnet threats, our proposed system has defined "signatures" that allow it to be detected. An important difference between this model and others, however, is that the network traffic signatures do not pass through network devices traditionally used for identifying malicious network activity.

A2 Events: A2 infection events target specific hosts and thus, do not expose the initial bot the way port scanning activities can. This activity is identical to an E2 event except that it originates interally to the network. Thus, the signature has only changed in the source of the data.

Announcement Message: The announcement message generated by a newly infected system must be a valid broadcast message, but must be a custom message or a relatively uncommon one such that the TB does not respond to non-infected systems regularly. In either case, the broadcast announcement message, once identified, is a signature of the botnet.

Token Passing/Management: In many locally switched networks, such as a university computer lab, it is not common for individual system to communicate directly with each other. The presence of even small data flows between systems in the switched network may be sign of questionable activity.

Detection Requirements

As has been shown, a covert switch-based communication model has a number of unique signatures that can be used by a detection mechanism. Thus, by incorporating a sufficient set of monitoring nodes throughout the network, sub-botnet activity can be detected. In essence, the level of protection is directly correlated to the level of monitoring.

In order to be able to identify all of the covert communication methods proposed in this paper, it becomes necessary to implement IDS capable monitoring at all switch devices in a network. Incorporating this level of monitoring may pose a problem for large networks due to the sheer amount of data that might be generated. However, as monitoring at the switch level seeks to identify very specific type of "special case" attacks, the analysis of the network traffic may be able to be done in a simple manner.

7 Summary and Future Work

This paper presented a potential communication model for covert botnet communication. This model enables a group of systems within a switched subnet to participate in a global botnet infrastructure without generating communication patterns that would allow an external monitoring system to identify the compromised hosts. While the proposed system can be detected, it requires an in-depth monitoring infrastructure above what is typical in many enterprise networks.

The future work of this project is aimed at creating an efficient IDS-type switch-based internal botnet communication monitoring system. The system will

potentially act as a remote data gathering node for BotHunter to be able to correlate internal traffic patterns with internal/external patterns. It is hypothesized that such system will minimize the logging and computational requirements on an individual switch monitor and improve the overall accuracy of the system.

References

1. Paul Barford and Vinod Yegneswaran. An inside look at botnets. In *Special Workshop on Malware Detection, Advances in Information Security*. Springer Verlag, 2006.
2. James R. Binkley and Suresh Singh. An algorithm for anomaly-based botnet detection. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, pages 43–48, July 2006.
3. John Canavan. The evolution of malicious IRC bots. In *Proceeding of Virus Bulletin Conference 2005*, October 2005.
4. Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, pages 39–44, July 2005.
5. R. Droms. Dynamic Hosts Configuration Protocol. Internet Engineering Task Force: RFC 2131, March 1997.
6. D. Geer. Malicious bots threaten network security. *IEEE Computer*, 38(1):18–20, January 2005.
7. Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: Overview and case study. In *First Workshop on Hot Topics in Understanding Botnets*, 2007.
8. Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium (Security'07)*, August 2007.
9. N. Ianelli and A. Hackworth. Botnets as a vehicle for online crime. CERT Coordination Center, December 2007.
10. B. McCarty. Botnets: Big and bigger. *IEEE Security and Privacy*, 1(4):87–90, 2003.
11. Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. A multi-perspective analysis of the storm(peacomm) worm. Technical report, Computer Science Laboratory, SRI International, October 2007.
12. B. Saha and A. Gairola. Botnet: An overview. CERT-In White Paper, June 2005.
13. Vincent Scarlata, Brian Neil Levine, and Clay Shields. Responder Anonymity and Anonymous Peer-to-Peer File Sharing. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, pages 272–280, November 2001.
14. Brandon Shirley and Chad D. Mano. Hiding botnet communication in a switched network environment. Technical report.
15. W. Timothy Strayer, Robert Walsh, Carl Livadas, and David Lapsley. Detecting botnets with tight command and control. In *Proceedings of 2006 31st IEEE Conference on Local Computing Networks*, pages 195–202, November 2006.
16. Ryan Vogt and John Aycock. Attack of the 50 foot botnet. Technical Report 2006-840-33, Department of Computer Science, University of Calgary, August 2006.
17. Beverly Yang and Hector Garcia-Molina. Comparing hybrid peer-to-peer systems. In *The VLDB Journal*, pages 561–570, September 2001.
18. Cliff C. Zou and Ryan Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 199–208, June 2006.