Sub-Botnet Cordination Using Tokens in a Switched Network

Brandon Shirley

Chad D. Mano

Sub-Botnet Coordination Using Tokens in a Switched Network

Brandon Shirley and Chad D. Mano
Department of Computer Science
Utah State University
Logan, Utah 84322
Email: {brandon.shirley, chad.mano}@usu.edu

Abstract—**Botnets have evolved to incorporate peer-to-peer communication for the purpose of better hiding the administrative source of the botnet. Current botnet detection mechanisms identify network traffic patterns at strategic locations within a network such as the gateway. As detection techniques improve, botnet design will continue to evolve to evade detection; thus, it is advantageous to identify potential future botnet models for the purpose of developing defense mechanisms before an actual new attack type is seen in the wild. This paper presents a model for coordinating external communication among bots located within the same switched network. This model prevents a gateway-based monitor from correlating external communication dialogs as the internal source of the communication is not a single bot-host. Future phases of this project include developing efficient techniques for mitigating this potential future botnet model.**

I. Introduction

Currently, botnets are considered one of the greatest threats to the security of the Internet [4], [8], [15]. A botnet is a collection of compromised computer systems (called zombies or bots) that are controlled by a single entity (called a botmaster or botherder). The power of the botnet lies in the ability of a botmaster to execute a large-scale distributed attack while remaining hidden as the true source of the attack. Botnets can be used for DDoS, SPAM, phishing, and other types of Internet crime.

Most botnets are administrated using a centralized Command and Control (C&C) approach. In this model all bots receive instructions directly from a central system such as an IRC server. The centralized server, however, presents a weak-point as the botnet can potentially be made impotent if the server is identified and made inaccessible from the Internet. Peer-to-Peer (P2P) style communication is one way a botnet can overcome this weakness.

In a P2P botnet the C&C is decentralized as individual bots receive instruction from other bots, not necessarily the botmaster. Thus, even if a number of bots are identified, security professionals may not have any information as to the actual location of the botmaster. The advent of P2P botnets was not unexpected as researchers were actively engaged in forward-looking research when the first large-scale P2P botnet was identified in January 2007 [10].

This paper takes a similar forward-looking approach in presenting a token-based model for coordinating communication among bots that are part of the same switched network. Our model prevents network administrators from identifying communication patterns between an internal host and external systems that would be indicative of P2P botnet communication. This is accomplished by strategically controlling the amount and type of external communication each bot takes part in, and then sharing any obtained data among all identified bots in said switched network.

It should be noted that this model is designed to evade current detection techniques, but is not immune to detection altogether. Detection of a botnet utilizing a model such as this requires much more in-depth monitoring of network traffic than is typically done. Future work includes developing an efficient switch-level monitoring system that provides the ability to correlate a complete analysis of internal only traffic with the border-crossing analysis provided by current detection systems. This paper includes a proposal for the requirements of such a system, but implementation and assessment of such a system is not included here.

II. 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], [6], [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], [9], [18]. In order to evade detection various techniques exist to hide or masquerade botnet communication activity [5], [8], [16].

More recently, botnets utilizing a P2P [13], [17] style communication infrastructure have been proposed [6], [16] and even discovered in the wild [10]. 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 of the authors' knowledge, BotHunter [7] is the most effective tool for detecting the existence of a bot within a local network, including bots utilizing P2P style communication. Our proposed model is designed to evade an edge-based system such as BotHunter by creating a cooperative group of infected bots within a locally switched network. The following section discusses Bothunter in more detail and outlines our approach to evading detection by a system like Bothunter.

III. Sub-Botnet Creation Background

The details of creating a sub-botnet are provided in [14], but are summarized here briefly. After a system is compromised by an external source, the newly compromised system must determine if other systems within the switched network are already part of the botnet. If no other systems are detected then the newly infected system attempts to identify and infect other systems within the switched network. This is done in an intelligent manner in order to minimize the chance of being detected by network security devices. This method of infection is advantageous as infections originating external to the network have a greater potential for being identified by existing IDS systems. It may not be feasible, however, to infect all systems in this manner and external infections still play an important role in botnet propagation.

If other systems on the network have already been infected then the newly infected system is identified and incorporated into the sub-botnet through the use of a stealthy broadcast message via an existing protocol such as DHCP. The sub-botnet continues to monitor for newly infected systems resulting from a system compromise originating from an external network. As the sub-botnet grows in size, it is better able to evade detection by a gateway-based monitor since it can more effectively distribute external communication tasks among the members of the sub-botnet.

IV. Sub-Botnet Management

To the best of our knowledge, BotHunter is the most effective technique for detecting botnet communication, including that of P2P botnets. BotHunter identifies "dialogs" between internal and external hosts including the following events that pertain to our technique.

- E2: External to Internal Inbound Exploits
- E3: Internal to External Binary Acquisition
- E4: Internal to External C&C Communication

A system is identified as being infected by a bot if a certain pattern of communication events, that includes at least two of the aforementioned events, is detected. Our model introduces three events that a bot can execute which evade detection by the monitor, but results in the same disbursement of information to the bots.

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

A sub-botnet can avoid detection by strategically utilizing A* events in order to minimize the number of E* events that are required to maintain effective communication with the overall botnet. Additionally, the E* events must be executed in a way that is not only minimal on a sub-botnet basis, but is strategic on a bot-basis, controlling the external exposure of any single bot.

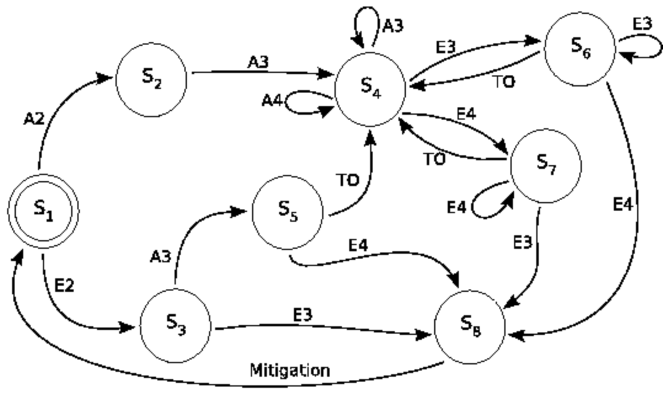

Fig. 1. S_1 is an uninfected host, S_4 is the ideal state, S_8 is a detectable state, and all other states are considered semi-safe or undetectable. Old events are pruned as indicated by a Time Out (TO).

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.

Figure 1 illustrates the goal of limiting the monitorable exposure of a single bot at any given time. Note that missing transitions mean that a transition does not apply or that the transition does not result in a state change. As an example, consider a bot that executes an E2 event followed by an A3 event. If the next event is an E4 event, the bot will be detected by the monitoring system. The bot must still obtain C&C communication to be an effective botnet participate, thus, it can receive the necessary information through an A4 event. However, if an E4 event occurs before a Time Out (TO) period, an estimate of the pruning period of the monitor, it will still be detected.

We utilize a round-robin style token-passing scheme to control E* events, thus protecting bots from executing communication dialogs that would expose the bot to an IDS monitor. The token acts as an authority key for the bot that possesses it, termed the Token Bot (TB). The TB performs internal to external actions and maintains a report of past E* events executed that is shared with all bots in the sub-botnet. 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, or that each bot performs the same E2, E3 or E4 event each time it initiates said event. Preferably, both criteria would be met, but this depends on the size and volatility of the sub-botnet. The token also ensures that only one bot is engaged in such an event at any given time.

To maintain the restrictions on E* events we utilize a token passing scheme that transfers TB duties within the sub-botnet. The basic token passing process from the perspective of a new

TB is as follows: 1) Token Acquisition (TAQ), 2) perform Token Action (TAC), 3) Token Action Result Propagation (TARP), and 4) the Token Pass (TP). If a problem prevents the TB from completing all steps of the process, a Token Election (TE) takes place to select a new TB. The TB also holds the responsibility of identifying new bots infected via E2 events and sending the new bot the current binary.

A. Token Data

The token is a data structure that contains all required information to enable the TB to make intelligent decisions about executing internal to external communication, forwarding information to the sub-botnet, and passing the token to other bots. The data structure includes the following information:

- Report list - internal only peer list
- Action History - last two internal to external actions performed (E3 or E4 events)
- Timestamp - indicative of when the token was compiled
- Bot Binary Version - denotes current binary version

The Report list is a local peer list which only contains bots in the same switched subnet. Each entry in the report list contains the bot's MAC address, the last TAC it performed, a timestamp associated with the last TAC performed or with the original infection if no TACs have been performed, and a dormancy counter to track unresponsive bots. The Report list is the primary tool used for token passing and token election should token passing fail. The complete Report list is not distributed over multiple bots as this is not beneficial for sub-botnet management and does not achieve the same botnet hiding characteristics as it does in the Internet where access to the other bots is restricted. The TAC for the given bot is implied by the information in the Report list.

B. External Communication

Following a TAQ the new TB is listed in the Report list with the associated TAC to be performed. This should be the same action it performed previously, but may be a different action if sufficient time has passed since the previous action was executed. In any event, the action to be performed should not be one of the two previous actions performed by former TBs.

The three possible TACs that a TB can execute are: command update, an E3 event; peerlist update request, an E4 event; or a binary update check, an E4 event. The two E4 events are considered separate types of communication in the interest of token management. In the case where only two bots exist in the sub-botnet, a single bot will perform the two E4 event tasks.

Prior to executing a TAC, the TB issues a TAQ acknowledgment that contains the action that it will perform. This message acts as a heartbeat indicator for the bots and forces each bot to reset its timeout period. If the timeout period for a bot expires prior to receiving the next TARP message, that bot will issue a TE request to recover the token. This TE process will be presented shortly.

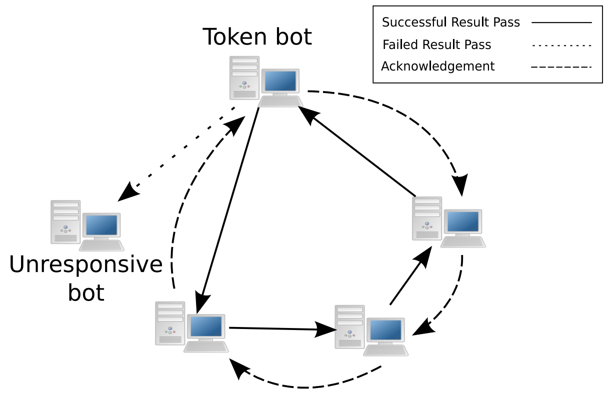

Fig. 2. Token Bot passing the token as part of TARP.

1) Command and Binary Updates: The command update request and binary update check for a sub-botnet work the same for the TB as they would for an external peer that is not part of any sub-botnet. Thus, external bots do not need to distinguish between a sub-botnet and a non-sub-botnet request. The only modification in the process is that a TB will then forward any updates on to the other members of the sub-botnet.

2) Peerlist Update: The peerlist update is executed via proxy using an external peer on the TB's peerlist. The TB sends a peerlist update request to an external peer, propagating through the list until an acknowledgment is received. If no external peers respond, the TB falls back on the default peer discovery method as denoted by the P2P protocol used by the botnet. Assuming the TB finds a responsive peer, the external peer generates an updated peerlist and forwards it to the TB. The TB then forwards this new list to the sub-botnet and each individual bot updates its list accordingly.

C. Sub-Botnet Message Propagation

After a successful TAC, the TB must propagate the results to the members of the sub-botnet, this is the TARP phase of the process; to achieve this propagation the TB sends the Token data to each bot as shown in Figure 2.

The TB first places itself at the bottom of the Report list then attempts to forward the Token to the first bot in the list. If this bot is unresponsive the TB continues on to the next bot on the list. Once the Token has been successfully transmitted, the receiving bot forwards the Token to the next bot on the list. This process continues until all bots have received the Token and the TB that initiated the process receives the Token back.

If a bot is unresponsive to the TARP attempt, its dormancy counter is incremented and the next bot on the list is contacted. During future TARP processes a bot with a non-zero dormancy counter will be selected for a connection request in a probabilistic manner based on the value of the counter. If it is still unresponsive the counter will be incremented. If the dormant bot does not respond prior to the counter reaching a predetermined value, it is assumed to be permanently unreachable and is removed from the Report list. If, at some point, the bot is responsive the dormancy counter is reset to zero.

When a bot has the Token it checks its binary version with the current version listed in the Token. If an update is in order, the bot will make a request to the TB from which it receives the next TAQ broadcast. This places the burden of initiating an update on the individual bot rather than on the TB. Additionally, each bot may update any outdated information in the Report list.

When a bot receives the Token data it is, in essence, the TB until it forwards the Token data. However, it does not take on the communication duties of the TB. When the TB that originated the TARP process receives the Token again it proceeds to the TP phase. If the original TB is not available for some reason, the bot possessing the Token takes on the responsibility of executing the TP. The bot knows it must assume this responsibility if it cannot successfully communicate with any bots in the remainder of the Report list. This assumption of responsibility allows event propagation and token passing to continue even if the TB is removed from the network mid-process.

D. Changing the Token Bot

After receiving the Token back the TB must decide to which bot it will pass the token. This is done by identifying the next TAC to be performed and selecting an appropriate bot to carry out the action. The TB references the Action History to identify the least recently executed TAC. The TB then searches the Report list for new bots, those that have not executed a TAC and have an initial infection timestamp older than a pre-defined minimum age; if they exist, any one of them may be used to perform the required TAC. Otherwise, the TB references the Report list to identify the bot with the oldest timestamp associated with said TAC.

The TB initiates the process by sending a TP offer to the target bot. If the target bot identifies itself as having an old bot binary during the TARP phase, it replies with a message stating such and receives the updated binary from the TB. Once the binary has been updated, or if the binary was already up-to-date, the target bot issues a TP acceptance to the TB. The outgoing TB responds with a final acknowledgment, including the current token data, and the incoming TB takes over.

The new TB receives the current version of the token even though it just received a copy during the TARP process. During the TARP process it is possible for any bot to update information in the Report list before the new TB receives the Token, thus the new TB must obtain the new version to prevent updated information from being lost.

E. Token Bot Fault Tolerance

In the event that the TB becomes unavailable for any reason, a TE is executed to reestablish the management authority within the sub-botnet. The TE request can be initiated by any bot after the timeout period of the bot has expired (the timeout period is reset each time communication from the TB is received). As the TE request is initiated by an independent bot, it is possible that multiple bots initiate near simultaneous requests as shown in Figure 3. In the case of simultaneous TE

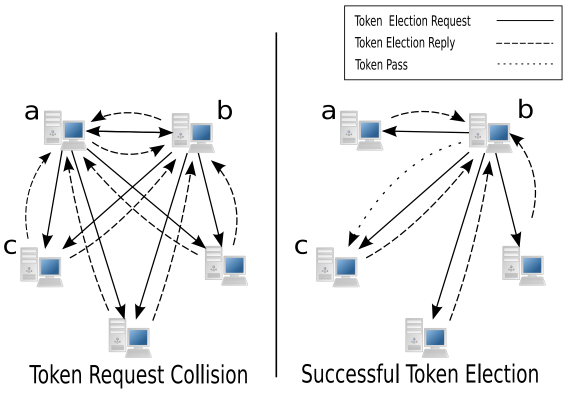

Fig. 3. In the Token Requestion Collision bots a and b initiate a TE request within the same timespan. In the Successful TE, bot a finally passes the token to bot c, at which point bot c would make a TAQ broadcast.

requests, special handling must be considered. This situation may arise if two or more bots issue the request at the same time or if there is a broadcast propagation delay that allows for another bot to issue the same request before receiving the broadcast. If a bot that issues a TE request receives a TE request from another bot, it will cancel its TE. Each bot will then generate a random backoff that must elapse before they make another TE request, this will insure that only one TE request succeeds at any given time. When a bot receives a TE request it will send its report list with the report list's timestamp to the requesting bot. If there is a collision, the requesting bot will still accept responses but will not actually perform the election, as shown in Figure 3.

Upon receiving the Report lists, the bot that initiated the TE will look for the Report list with the most recent timestamp and use that to replace its own Report list. Anytime a bot that has not made a TE request receives a request, it will reset its timeout period. This will keep the number of bots trying to initiate a TE to a minimum.

The bot that issues the TE request waits for responses for a set time period optimized for the size of the sub-botnet. If the wait period expires without the bot in question receiving another bot's TE request, then the bot will issue a broadcast that TE was successful. When the other bots receive this message they again reset their timeout period. If the timeout period is exceeded before another message is received then the bot will attempt to contact the bot that issued the broadcast. If the bot is responsive then the wait will continue, otherwise another TE will be initiated in the same manner.

The bot that issues the TE success broadcast will use the Action History and the most up-to-date Report list that it received to pick a new TB. The method for choosing a new TB is 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, as shown in Figure 3. At this point the new bot will make the TAQ acknowledgment broadcast and the sub-botnet may resume normal activity.

V. Detection Requirements

The proposed model is designed to enable a group of cooperating bots to avoid being detected by traditional IDS and dialog-based network monitoring systems. This is accomplished by strategically controlling internal to external botnet traffic, thus eliminating the ability of a perimeter-based device to correlate communication patterns of individual systems with that of known botnet traffic patterns.

This model is founded on the assumption that complete network traffic monitoring is performed at the perimeter of a network or sub-net. Probabilistic monitoring at the switch-level, as done by [11], may not be sufficiently fine-grained to effectively identify the sub-botnet activity.

As with traditional worm, virus, and botnet threats, our proposed system has defined "signatures" that allow it to be detected. Thus, if a network provides complete monitoring of traffic at the switch-level, sub-botnet activity can be detected. While simple in concept, network monitoring at this level will likely generate great amounts of data that may render effective analysis infeasible.

The following are characteristics that are necessary in building an effective system to mitigate a sub-botnet threat.

Complete Switch Coverage: In order to effectively identify communication that passes through at most one switch, it is necessary to monitor every switch within a network. Additionally, the computational power within the switch monitor may be limited, thus any communication that moves towards the gateway router should be ignored by the immediate switch monitor. If the communication does not eventually move to the router it will be analyzed by another switch that identifies that the data will not pass through the router.

Switch Monitor as a Support: The sub-botnet model presented is effective because it prevents a perimeter-based system from identifying certain components of botnet behavior. Thus, the switch monitor should act as a support to the perimeter-based system, filling in pieces and enabling the system to correlate external and internal-only traffic to effectively identify an infected bot system. Creating a stand-alone switch monitor/analyzer would place undue burden on network administrators and would be much more costly to implement in terms of time and equipment requirements.

Efficient Messaging: Acting as a support to a centralized IDS system, the switch monitor must generate efficient messages for two reasons. First, overly verbose reporting may potentially have a negative impact on network performance. Second, all data received by the central system results in additional burden on the resources of the system. Thus, a switch monitor that delivers more data than is necessary creates inefficiencies in the overall network defense system.

VI. Summary and Future Work

This paper proposes a potential model for enabling covert botnet activity within a switched portion of a network. The model evades detection via traditional means by creating a cooperative infrastructure among infected bot systems that allows for strategic control over necessary external network communication. 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 a framework for implementing and testing the proposed model. In addition, the framework will be designed in a flexible manner to allow for the easy implementation of other proposed sub-network botnet models. This framework will act as a testbed for designing a defensive system according to the guidelines presented in the previous section. We envision that our system will interface with existing systems, such as BotHunter, to improve the overall accuracy and ability of existing botnet detection systems.

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] D. Geer. Malicious bots threaten network security. *IEEE Computer*, 38(1):18–20, January 2005.

[6] 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.

[7] 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.

[8] N. Ianelli and A. Hackworth. Botnets as a vehicle for online crime. CERT Coordination Center, December 2007.

[9] B. McCarty. Botnets: Big and bigger. *IEEE Security and Privacy*, 1(4):87–90, 2003.

[10] 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.

[11] Joseph Reves and Sonia Panchen. Traffic monitoring with packet-based sampling for defense against security threats. http://www.sflow.org, 2002.

[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. A model for covert botnet communication in a private subnet. In *Proceedings of IFIP Networking 2008*, May 2008.

[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.