Industrial Science
ISSN : 2347-5420

Volume-1 | Issue-7 | Oct-2014

COMMAND AND CONTROL CHANNEL USING DNS

Andrea Noreen D'silva[1] **and Nagalakshmi S Shagouftataskeen**[2]

[1]Student (MTECH CNE 3rd Sem).
[2]Assoc prof. Department of ISE, Dr.AIT, Bangalore, Karnataka, India.

Abstract:-A botnet is a group of compromised computers, which are remotely controlled by hackers to launch various network attacks, such as DDoS attack and information phishing.Botnet is a hybrid of previous threats integratedwith a command and control system and hundreds of millions of computers are infected. Attackers, in particular botnet controllers, use stealthy messaging systems to set up large-scale command and control. To systematically understand the potential capability of attackers, we investigate the feasibility of using domain name service (DNS) as a stealthy botnet command-and-control channel. We describe and quantitatively analyze several techniques that can be used to effectively hide malicious DNS activities at the network level. We conclude that the DNS-based stealthy command and- control channel (in particular, the codeword mode) can be very powerful for attackers, showing the need for further research by defenders in this direction. The statistical analysis of DNS payload as a countermeasure has practical limitations inhibiting its large scale deployment.

Keywords: Command , Control Channel , Domain Name System (DNS) .

1 INTRODUCTION

BOTNET command-and-control (C&C) channel refers to the protocol used by bots and botmaster (i.e., botnet controller) to communicate to each other, e.g., for bots to receive new attack commands and updates from botmaster, or to submit stolen data. A C&C channel for a botnet needs to be reliable, redundant, noncentralized, and easily disguised as legitimate traffic. Many botnet operators used the InternetRelay Chat protocol (IRC) or HTTP servers to pass information. Botnet operators constantly explore new stealthy communication mechanisms to evade detection. HTTP-based command and control is difficult to distinguish from legitimate web traffic. The feasibility of email as a stealthy botnet command and control protocol was studied by researchers in.

We systematically investigate the feasibility of solely using Domain Name System (DNS) queries for botnet command and control. DNS provides a distributed infrastructure for storing, updating, and disseminating data that conveniently fits the need for a large-scale command and control system. The HTTP protocolis for the end-to-end communication between a client and a server. In comparison, DNS provides not only a means of communication between computers, but also systematic mechanisms for naming, locating, distributing, and caching resources with fault tolerance. These features of DNS may be utilized t fulfill a more effective command-and-control system than what HTTP servers may provide.Botnets are like killer applications which are oganised and managed by some smart worker or we can say hackers.The software that creates and manages a botnet makes this threat much more than the previous generation of malicious code. It is not just a virus; it is a virus of viruses. The botnet is modular one module exploits the vulnerabilities it finds to gain control over its target. It then downloads another module that protects the new bot by stopping antivirus software and firewalls; the third module may begin scanning for other vulnerable systems.The botnet can be designed to download different modules to exploit the specific things that it finds on the victim.

The decentralized nature of DNSs with a series of redundant servers potentially provides an effective channel for covert communication of a large distributed system, including botnets. To play the devil's advocate, we focus on systematically analyzing the feasibility of a pure DNS-based C&C. The DNS channel is aided by being a high-traffic channel such that data can be easily hidden. As virtually anyone can create and register their own domain names (if available) and DNS servers, it is a system that can be easily infiltrated by hackers and botnet operators. DNS tunneling is a technique known for transmitting arbitrary data via DNS protocol, e.g., DNScat and DeNiSe. One application of DNS tunneling is to bypass firewalls, as both inbound and outbound DNS connections are usually allowed by organizational firewall rules. Because DNS is often overlooked in current security measures, it offers a C&C channel that is unimpeded. Because nearly all traffic requires DNS to translate domain names to IP addresses and back, simple firewall rules cannot easily be created without harming legitimate traffic. botnet simply tunnels its command and control traffic by sending it in DNS format for the end-to-end communication between bots and the bot master. The domains used by them are not registered and cannot be resolved.

Three items to consider to evade detection for attackers are:
1. Query activities. When and how frequent do bots issue DNS queries to pull updates from or submit data to the bot master? How to modify the victim's operating system to implement automatic query strategies?
2. Domain names. What domain names to use for communication and how to synchronize the generation of new domain names between bots and the bot master?
3. DNS payload. How to evade the detection through deep packet inspection by defenders on DNS payload?

1.1 Infection Mechanism

There are various types of methods for attacker to
distribute a particular bot. The three methods of bots propagation.
1) Web Download: A recent google study showed that web-based infection vectors are now commonplace. Web-based malware creates botnet-like structures in which compromised machines query web servers periodically for instructions and updates.
2) Mail Attachments: E-mail attachments with mass mailing worms can contain bots. Spam techniques simplify and enable fast spreading of bots easily.
3) Automatically Scan, Exploit and Compromise: The bots automatically infect the host that have vulnerabilities.

Typical botnet life-cycle as shown in Figure 1.

Botnet usually recruit new victims by remotely exploiting a vulnerability of the victim's machine via mentioned methods above. When the infection has achieved, the victim executes a script and downloads bot binary from some location. The bot binary installs itself to the victim and automatically run. The new bot contact a DNS server for getting the IP address of the IRC server. Then it will establish an IRC session with the server and join the C&C channel. Then the bot can automatically parse and executes the channel topic, which contains the default commands. Infected host executes commands from Botmaster via IRC server, for example, launching a distributed denial-of-service attack, sending mass spam mailings, or logging keystrokes.

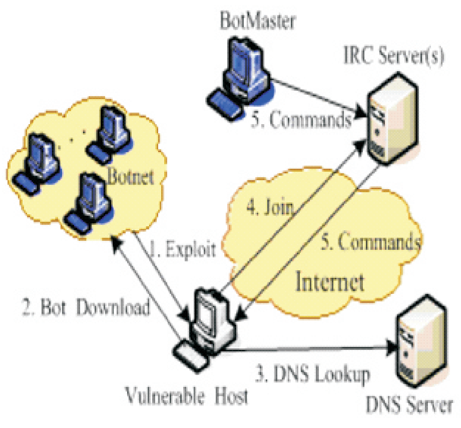


Fig1:The process of Botnet Infection

1.2 Command and Control Models

The C&C mechanism in botnet has great importance for us defending against botnet.

There are three possible C&C communication topologies and investigated their associated benefits and weaknesses.

1) Centralized C&C Model: A centralized model is characterized by a central point that forwards messages between clients.The centralized model has some advantages such as simple implementation and customization. However, the centralized C&C model will be detected and destroyed easier. Although this model has certain drawbacks, most bots use the centralized C&C model such as AgoBot, SDBot, and Zotob.
2) P2P-based C&C Model: For drawbacks of centralized model, the Botmaster shifts to P2P-based botnet. Compared with the centralized C&C model, the P2P based C&C model is much harder to discover and destroy. Botmaster can send commands from any peer. However, it is a more complex job for designing p2p systems. Some bots such as Phatbot and Peacomm , have used P2P communication as a means to control botnet.
3) Unstructured C&C Model: A bot will not actively contact other bots or the Botmaster, and would listen to incoming connections from its Botmaster. The Botmaster randomly scan the Internet and pass along the encrypt message when it detected another bot.

2 COMMUNICATION MODES

This section describe protocols that pass messages over the DNS between distributed entities, and illustrate the ease of setting up large-scale C&C via DNS. We describe two forms of communication modes: codeword mode and tunneled mode. Codeword communication allows one-way communication from botmaster to a bot client,

which is suitable for issuing attack commands. Tunneled communication allows for the transmitting of arbitrary data in both directions between bot and botmaster, which may be used for both issuing commands and collecting stolendata. The former only requires the ability to set a particulardomain name response, this could be done via any free DNS service, while the latter requires setting up an authoritative domain server. The controller of the botnet first needs to create a domain or subdomain, which is administered from a special DNS server. This DNS server waits for special name lookups, which it then translates into incoming data. The DNS server then responds with the appropriate data using the agreedupon semantics. We assume that the botnet controller (i.e.,botmaster) has access to the authoritative domain name server for some domains or subdomains. Bots across the Internet frequently receive commands and updates from a botmaster and launch attacks accordingly, as well as submit stolen data to the botmaster.

DNS resources records. The DNS system allows a name server administrator to associate different types of data with either a fully qualified domain name or an IP address. To send a message to a bot, an adversary can store data in any one of these types of records:

•A record specifies an IP address for a given host name.
•CNAME and MX records can point to textual data representing the alias or mailing host of a particular host name.
•TXT records are designed to store arbitrary textual data up to 255 characters.
•EDNS0 record allows storing up to a 1,280-byte payload. EDNS0 was introduced in RFC261 to extend the DNS protocol. When a capable server or client encounters this field, it can decode the packets, allowing several improvements to the basic DNS protocol. These features include larger UDP packet size, a list of attribute value pairs, and several extra bytes for commonly used flags.

2.1 Codeword Mode

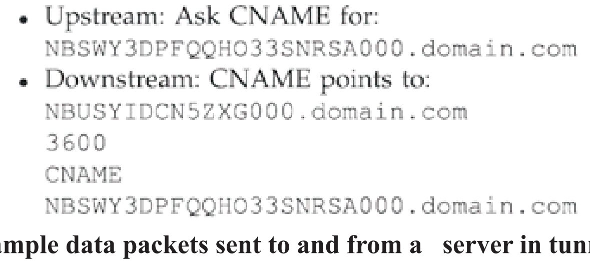
The codeword mode is a stealthy communication mechanism. It requires a botnet operator to decide upon a set of agreed upon codewords a priori. Each codeword represents a specific type of commands or attacks. The codeword appears in the DNS query as an innocent hostname. This hostname may be stored as any type of record (e.g., A, MX, CNAME). A request for an A or CNAME record tends to be the most common, and therefore, a preference should be given to these records types, so that queries would appear most like legitimate traffic. The client queries waits for a particular value in the server's response. Upon receiving the query, the DNS server (controlled by the botnet operator) returns the preset response that contains command information. If the codeword corresponds to denial-of-service (DoS) attacks, then the response may represent a target of DoS attacks. If the codeword corresponds to update, the client may contact the IP address returned for updated code or other instructions. It is important to note that the codeword can be chosen arbitrarily and does not need to correspond to a specific host or service. The codeword method allows a stealthy one-way commanding system. It can effectively evade detection approaches based on nonconforming packet sizes.

2.2 Tunneled Mode

The purpose of tunneled mode is to allow the two-way transfer of arbitrary binary data between a server and a client.This mode is referred to as tunneled mode, as one can tunnelstreaming data over this DNS communication method:

❖ Upstream communication is for a client to submit datato a (malicious) domain server. The client submitsthe data as a CNAME query by:
•encoding the data using a base32 encoding, using the encoded string to construct a host name, and
• send a CNAME DNS query. An example is shownin Fig. 2

❖ Downstream communication is for the server to issuecommands to clients. Upon receiving the abovequery from the client on a hostname h, the server:
•encodes the response as base32 data, and
•constructs and returns a CNAME record for h. An example is shown in Fig. 2.



- Upstream: Ask CNAME for:
```
NBSWY3DPFQQHO33SNRSA000.domain.com
```
- Downstream: CNAME points to:
```
NBUSYIDCN5ZXG000.domain.com
3600
CNAME
NBSWY3DPFQQHO33SNRSA000.domain.com
```

Fig. 2. Example data packets sent to and from a server in tunnelled mode

To prevent DNS caching from disrupting the communications, the server may set a short time-to-live (TTL). This tunneling method gives an operator the most options after implementation as the data stream can be arbitrary. Because of the arbitrary payload, the distribution of packet bytes may differ significantly from conventionally DNS payload. DNS protocol does not allow the server to initiate a connection with the client, the client needs to continually pull updates from the server. Both the tunneled mode and codeword mode require clients to frequently pull updates from name servers by querying the corresponding botnet's domain. Direct querying patterns are easy to detect (e.g., periodically sending DNS queries) and susceptible to simple aggregate analysis, such as counting DNS queries for each unique domains and identifying domains with abnormally large query volume at the host, local area network, or internet service provider levels.

2.3 Communication Protocols

Botnet usually use well defined communication protocols. Knowing about the communication protocols can help us determine the origins of a botnet attack and decode conversations between the bots and the Botmasters.

The communication protocols was be classified in three different categories.
1) IRC Protocol: This is the most common protocol used by Botmasters to communicate with their Bots. IRC protocol mainly designed for one to many conversations but can also handle one to one, which is very useful for Botmasters control their botnet. Secutiy devices can be easily configured to block IRC traffic.
2) HTTP protocol: The HTTP protocol is a popular communication method by botnet which is difficult to be detected. Using the HTTP protocol, botnet usually bypass security devices.
3) P2P protocol: Recently, more advanced botnet used P2P protocols for their communications[20]. Some recent variants of Phatbot and AgoBot, Nugache, Peacomm used P2Pcommunication protocols.

3.BOTNET DETECTION

Detecting and possibly defense botnet is an important research task. There are mainly two approaches of botnet detection and tracking methods. One is signatures based method and the other is based on anomaly.

Signatures-based detection: There is some veryrecent work on the signatures based detection of botnet. Honeypots and honeynets are effective detection and analysis techniques at a reasonable cost and without false positives, and hence there has been much recent research inthis area. Signatures-based detection cannot detect unknown attacks for which there is no signature available, and some botnet adopting some circumvention techniques such as polymorphic, metamorphic, obfuscation and packer.

Anomaly-based detection: Anomaly based system that combines IRC statistics and TCPwork weight for detecting IRC-based botnet anomaly-based passive analysis algorithm that is able to detect IRC botnet controllers running on any random port without the need for known signatures or captured binaries. However, anomaly-based detection techniques have high false alarm rate and the complexity involved in determining what features should be learned in the training phase.

3.1 Exponentially Distributed Query andPiggybacking Query

Both distributed query strategy and a piggybacking query strategy can be used to hide bot activities while communicating with a botmaster in a timely fashion.

Exponentially distributed query strategy. The Poisson process is previously believed to be a suitable model for representing stochastic processes, where arrivals are independent on each other, i.e., memoryless. Clientside DNS request arrivals are modeled by Poisson processes with exponential random variables with different rates Γ In exponentiallydistributed query strategy, a bot probabilistically distributes DNS queries so that their intervals follow an exponential distribution with a parameterized arrival rate Γ b. Because of thememoryless feature of the model, the bot does not need to store the previous communication history.Implementation of this query strategy is as follows:

1. The bot sends a DNS query.
2. It computes an interval t by drawing from an exponential distribution with parameter Γ b
3. The bot sleeps for t and repeats from Step 1.

3.2 Piggybacking query strategy

The composition of domains is usually dynamic. The piggybacking query strategy leverages this fact. A bot passively listens on the host's DNS traffic or name-translation related function calls and sends DNS queries when legitimate DNS queries are being made. Thus, the bot's query is blended among a group of legitimate DNS queries.

In the piggybacking query strategy, a bot's communication with the controller is constrained by the host's activities. Therefore, we focus on analyzing its timeliness, in terms of the dissemination efficiency of new command and data. Minimum TTC is a threshold aiming to prevent a bot from sending queries too frequently, whereas maximum TTC is a threshold for keeping the liveliness of the communication between the bot and the bot master in case of an inactive host.

a bot does not send any DNS query if the bot's previous DNS query was sent within the minimum TTC. At time t, a bot needs to send a DNS query to check for update from the bot master if the interval between t and the time when the bot sends the previous DNS query equals the maximum TTC.

In piggybacking mode, bots need to know when a legitimate DNS query is being made.There are different approaches for obtaining this information:

1. Traffic sniffing. Because DNS server runs on port 53, an outgoing packet from the host (client) to a destination IP on port 53 is an indication of an outbound DNS request. The bot may sniff the network traffic to identify the right moment to issue its DNS queries. The bot may either directly program with pcap library or call existing tools such as iftop (in Linux), which uses pcap. This approach gives a cross platform and system-wide monitoring solution, which covers the DNS queries from any application. Filters all TCP and UDP packets, both of which are possible protocols for carrying DNS requests, and reports the time stamps whenever a DNS request is identified. It also distinguishes the bot's own DNS query from legitimate ones, so that piggybacking is only performed on legitimate DNS queries. This prototype demonstrates a feasible way for our proposed piggybacking query strategy.

2. Process hooking. Another approach is to hook DNS related function. In Linux, bots can watch the calls for DNS-related APIs such as gethostbyname()function in lib bind library. gethostbyname looks up all IP addresses associated with a host name and is implemented in the resolver library. One way of hooking into the API function is for bots to register a .so file to the LD_PRELOAD environmental variable, which may or may not require root. In the registered .so file, the target API function is replaced by the attacker's version which can notify the bot whenever this function is called.

3. Rogue library. Alternatively, attackers may replace the network related shared libraries in the OS with an instrumented version (requiring root) so that the shared library is changed permanently on the hard disk. The rogue library is loaded by any application. It reports every DNS request to the bot. This approach is system wide, so the change is once and-for-all.

4 DOMAIN NAMES

Long-lived domain names are easy to manage and cheaper to maintain; however, they are susceptible to aggregate analysis. Domain flux refers to using short-lived domain names in botnet

C&C. Domain flux typically requires bots and botnet controller to independently derive new domain names periodically.

4.1 Domain Name Generation with the MarkovModel

One simple approach is for bots and their controller to independently compute the hash value of an incremental counter and a shared secret at each epoch, secret Þ, where H is a one-way collision-resistant hash function. For example, the bot and botmaster may share a secret, which is concatenated with the current epoch, e.g., current date, to derive a hash value for short-lived domain names,. The clocks on all bots and the bot master have to be synchronized, especially when the epoch is short. This time-stamp- based domain-generation algorithm Technical issues associated with hash-based domain names such as domain length, collusion, and unavailability of domain names can be easily dealt with and are omitted.

The domain generation has to be independent, consistent, and synchronized. These can be achieved in this MC-based approach by using the same random number generator and the same initial seed. Γenerators like the linear congruential generator may be used in this respect. The botmaster has to communicate with the bots the following items: the Markov probabilities for state transitions, parameters for random number generator, and parameters for the initial seed and timing interval.

Each domain name is represented as a vector of [x1,x2,…x37]^T37 features; the feature set includes 26 letters in the alphabet, 10 digits, and the character "-". The cosine similarity S between two domain names A and B, represented as vectors, is defined as,

$$S(A,B) = \frac{\sum_{i=1}^{37} A_i \times B_i}{\sqrt{\sum_{i=1}^{37} A_i^2} \times \sqrt{\sum_{i=1}^{37} B_i^2}}. \qquad (1)$$

4.2 Subdomains and Domain Registration

Botnets have been observed to use subdirectories for communication.However, for a DNS-tunneling-based channel,subdirectory approach does not apply, as the botmaster does not run a web server and the communication is based solely on domain name systems.We consider that botnets often use third-level domains instead of subdirectories.

When using short-lived domain names, the botmaster needs to consider whether to use the generated string for SLD or 3LD names Using it as a 3LD makes it easy for botmaster to update the DNS records of a domain, which is static and long lived. From the attacker's point of view, while generating realistic looking domain names, it is reasonable to generate only the SLDs and pick some TLD suitably or randomly. Compared to millions of SLDs, the list of TLDs is very small. However, the SLD and TLD names are not entirely independent; the character-pair distribution of the SLDs may vary, particularly when the TLD includes country codes (ccTLD).The attacker can make use of the dependency to generate more realistic looking domain names.

We expect botnet controllers to create and register thedomains to be used, which allows them to take advantage of the existing and decentralized DNS infrastructure for scalability, fault tolerance, and storage. To register a new SLD associated with a top-level domain (such as .com), a registrant (e.g., a botnet controller) needs to apply to a domain name registrar and pay the (annual) fees. The botnet controllers may specify their own IP addresses of authoritative name servers to host the domain's resource records. This approach gives them the flexibility in managing the C&C operations and customizing the DNS traffic. There is monetary cost associated with registering a new SLD name.

5 CONCLUSIONS

Our research work shows that DNS—in particular the codeword mode combined with advanced querying strategies—can be used as an extremely effective stealthy C&C channel. To address the open problem raised in on how to algorithmically generate short-lived and realistic-looking domain names, we found that using MC produces realistic-looking domain names. Understanding the capacity of botnets communication power helps identify and eliminate nefarious attacks launched from them. DNS based botnet C&C is more stealthy than application-based C&C), and such a C&C system also benefits from the decentralization of DNS.Besides C&C DNS tunneling may also be used for exfiltrating sensitive data by attackers including rogue insiders. Payload inspection has been proposed for detecting data leaks. From defenders' perspective, the approach of user-intention-based anomaly detection has been demonstrated effective in detecting abnormal system events such as unauthorized file creation and malware-triggered outbound traffic. Because DNS queries are usually automatically issued by applications or the OS, the causal relations between user actions and DNS traffic may not be obvious.

REFERENCES

1.Kui Xu, Member, IEEE, Patrick Butler, SudipSaha, and Danfeng (Daphne) Yao, Member, IEEE, DNS for Massive-Scale Command and Control,IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTINΓ , VOL. 10, NO. 3, MAY/JUNE 2013
2.P. Butler, K. Xu, and D. Yao, "Quantitatively Analyzing Stealthy Communication Channels," Proc. Ninth Int'l Conf. Applied Cryptography and Network Security (ACNS '11), pp. 238-254, 2011.
3.D. Dagon, "Botnet Detection and Response, the Network Is the Infection," Proc. Domain Name System Operations Analysis and Research Center Workshop, 2005.
4.Paul Watters Internet Commerce Security Laboratory University of Ballarat, A Survey on Latest Botnet Attack and Defense,2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11
5.Chao Li National Computer network Emergency Response technical,Botnet: Survey and Case Study,2009 Fourth International Conference on Innovative Computing, Information and Control