Botnet Tracking: Tools, Techniques, and Lessons Learned

Dr. Jose Nazario
Arbor Networks
jose@arbor.net

Summary

The threat posed by botnets has become increasingly high profile in the past several years, most recently at the World Economic Forum in Davos, Swizerland, where Dr. Vint Cerf (Google) noted that botnets are the biggest threat to Internet stability and security [1]. Arbor Networks has been studying botnets for some time in an effort to understand and trace back DDoS attacks and to assess attacker's motivations. Our experience in this research has shown that tracking botnets requires both hard resources and soft skills, and that there are many considerations that are not immediately apparent. We have found that in the past year the pace of fragmentation away from the standard IRC-based bots to custom tools and new communications channels has accelerated, due in part to increased interest in tracking botnet activity. Furthermore, we have found that botnets are a growth industry and have attracted a growing number of botnet managers, complicating the problem and adding pressures on professional Internet criminals.

Note: Parts of this paper have appeared at the Virus Bulletin 2006 Conference, Montreal, Canada.

Introduction

Botnets, networks of compromised computers used for nefarious means, are a major problem source in the climate of modern-day Internet security. They are significant contributors to the malicious and criminal activity on the Internet today, and, more importantly, are an underground network whose size and scope is not fully known. The information security community does know that botnets are a major source of Internet-scale problems, including host scans, exploit attempts and attacks, and spam. Botnets are also a common tool used to conduct distributed denial of service (DDoS) attacks due to the immense aggregate bandwidth that botnets command. Especially with the large amount of home users on broadband Internet, a few hundred cable modems can easily flood even a medium-sized host off the Internet. With some botnets as large as tens of thousands of machines, few hosts could withstand a sustained assault without severe degradation of service.

Botnet operations and the features in bots have been explored at length in prior papers [2-4]. Because bots have rich command syntax and are increasingly fully featured, they provide unfettered access to the infected host. This facilitates any number of nefarious activities. Bot infected hosts are often the victims of many kinds of malware installations, including spyware, rootkits, adware, and software that has often been installed using the bot's download facilities. En masse, bots give outsiders a large attack surface area from which to harvest information and commit various crimes. Understanding this aggregate problem, and developing means to respond to it, is a pressing need in current Internet security [5].

The key benefit to botnets tracking for researchers is the direct observation of malicious activity. Secondly, these observations give a researcher insight into the people behind a botnet's creation, its use, and their motivations. All of these features must otherwise be either inferred from captured malware or measured by honeypot technologies. Active tracking of botnets by participating can yield a partial view of the botnet's activities.

However, this approach has quickly begun to lag behind the more interesting botnets responsible for high profile DDoS events and the significant growth in the global spam problem. Previous botnet tracking efforts have focused on actively monitoring a limited number of IRC-based botnets. In this paper we outline limitations in this approach, both in tracking these IRC-based botnets and also in tracking the growing prevalence of HTTP-based botnets.

Botnet Tracking Goals

Before beginning any botnet monitoring program, researchers should have a clear set of goals outlined they wish to achieve. The requirements, both in terms of

resources and skills, differ between the broad goals outlined below and are key to the success or failure of such a project.

The goals of tracking one or more botnets can be loosely grouped into three major categories. The first category is akin to the goals a biologist and focuses on malware sample collection. The main goal of the second category is much more like those of a research anthropologist and focuses on studying the behaviors of the people behind botnets. Finally, the goals of the third category are similar to a detective's goals and focus on forensic uses. Typically, while one of these goals may be a focus of the project, elements of the other categories are included.

Antivirus companies, and increasingly security vendors, have taken an interest in tracking botnets to discover new malware samples that may otherwise remain unseen for some time. This approach makes sense when the goal is to protect a customer, and is similar to a field biologist who collects specimens for a catalog. Botnet tracking is often complementary to honeypot use where new malware samples, including bots and Trojans, are collected and analyzed. These are standard techniques in the malware analysis community and active botnet monitoring techniques extend this somewhat. Together with automated analysis tools such as sandboxes, new samples can be gathered and analyzed quickly with little human intervention.

For those who wish to profile the people behind botnets, much as an anthropologist might, botnet tracking becomes more labor intensive and requires a human analyst to study the logs and at times interact with the botnet managers. This is a largely unexplored field of research and is commonly pursued, in part, by any malware analyst. In this scenario, the researcher wishes to conduct both a personality profile to discover the means of the attacker as well as a threat estimate through assessing the knowledge and skills the attackers possess. By conducting such research, groups can be singled out for additional monitoring.

Finally, botnets can yield a tremendous amount of forensic information about global attacks, specifically spam and distributed denial of service (DDoS) activity [6]. Botnets provide an effective way to grow and manage a distributed traffic source army, and many popular IRC bots have built-in DDoS attack routines covering all of the standard DDoS attack methods: SYN floods, ICMP and UDP floods, and more specialized attacks such as specific packet types and options as well as connection floods.

The Arbor botnet tracking project's goals were primarily to gather direct observations from the sources of many DDoS attacks that could be used in traceback and response. A secondary goal was to assess the threat against the critical infrastructure posed by botnets and their operators. Previously, Arbor

Networks has tracked botnet activity to discover the origins of several DDoS attacks in an effort to shut down the attacks at the source [7]. Our studies found that 50% of the IRC botnets we tracked launched DDoS attacks, although not all were against globally interesting targets. As described later in this paper, over the past year, shifts in botnet tactics have made achieving this goal more labor and resource intensive.

Prior Research from Other Groups

Most previous botnet research has focused on analyzing common bot software and its capabilities [3, 4]. While this research is important and interesting, it does not provide the kind of visibility into current botnet activity that tracking live botnets offers. Other research has used honeypots as the mechanism for tracking botnets [2]. Though useful, point systems such as honeypots cannot provide broad or directed visibility into the botnet problem. This is because a honeypot must wait to be infected, and once infected by malware the researcher has no choice of what networks to monitor. Standard honeypot systems do not allow one machine to monitor multiple networks at a time, since there is a limit to how many infections one machine or machine image can support at the same time. This means that a large-scale botnet tracking project would require immense resources to provide global botnet visibility. Additionally, there is always a risk posed by infecting a machine with malicious software in that one can never be totally sure that a firewall will filter out all malicious traffic.

Previous research similar to ours has attempted to infiltrate botnets to provide root-cause analysis for DDoS attack mitigation [6]. Using the feature-rich tool "drone," the authors have demonstrated that an approach that infiltrates and observes a number of botnets in DDoS attack tracking is a promising but limited approach in dealing with the problem.

Because of these limitations, dedicated tools are needed to capture active malware samples from "the wild" and participate in botnets. The next section describes these tools and considerations, as well as limitations, in their designs.

Botnet Tracking Tools

The tools and techniques needed for effective botnet tracking can be flexible, although several caveats apply. One of the base requirements for tracking is to avoid detection as a "snoop" and to try and achieve the appearance of being an infected bot. Many botnets, specifically the active and interesting botnets, will be cautious about suspicious hosts and block their access to the network. Amateur botnet operators, in contrast, may not notice or will greedily accept any client in an effort to increase the size of their botnet.

Malware collection and sharing is one of the most effective ways to discover new botnets to track, and honeypots are a popular way to gather up new malware samples such as

bots. While it is tempting to simply place a live system with real vulnerabilities on the Internet to receive malware, this has neither the scale nor the efficiency to capture many samples quickly.

One of the more popular lightweight honeypot systems to gather such samples is the Nepenthes platform, an exploit simulation tool [8]. Bots and malware executables that propagate autonomously will hit a sensor and launch an exploit. The Nepenthes sensor will react to the exploit and any subsequent commands, downloading the malware if it is so instructed. While such a system is obviously biased towards bots in the scan-and-exploit cycle and limited to its own collection subnets, it is a proven and effective malware collection tool. When combined with a network of sensors, such as the MWCollect Alliance, dozens of new malware variants can be found per hour. Furthermore, this can be combined with email inbox traps, instant messaging accounts that act as honeypots, and "Honeyclients" to collect malware that propagates through other means.

```
[02022007 18:46:29 debug net mgr] Accepted Connection Socket TCP
(accept) xx.xx.247.107:4062 -> xx.xx.21.196:139
125 Sockets in list
[02022007 18:46:29 debug net mgr] Deleting Socket TCP  (accept)
xx.xx.87.118:2001 -> xx.xx.17.27:139 due to closed connection
[02022007 18:46:29 spam net handler] <in virtual
nepenthes::TCPSocket::~TCPSocket()>
[02022007 18:46:29 spam net handler] Socket TCP  (accept)
xx.xx.87.118:2001 -> xx.xx.17.27:139 clearing DialogueList (2
entries)
[02022007 18:46:29 spam net handler]    Removing Dialogue
"NETDDEDialogue"
[02022007 18:46:29 warn module] Unknown NETDDE exploit 76 bytes
State 1
[02022007 18:46:29 module] =------------------[
hexdump(0x08c0da78 , 0x0000004c) ]------------------=
[02022007 18:46:29 module] 0x0000  81 00 00 48 20 43 4b 46  44 45
4e 45 43 46 44 45  ...H CKF DENECFDE
[02022007 18:46:29 module] 0x0010  46 46 43 46 47 45 46 46  43 43
41 43 41 43 41 43  FFCFGEFF CCACACAC
[02022007 18:46:29 module] 0x0020  41 43 41 43 41 00 20 45  4d 45
50 45 44 45 42 45  ACACA. E MEPEDEBE
[02022007 18:46:29 module] 0x0030  4d 45 49 45 50 46 44 46  45 43
41 43 41 43 41 43  MEIEPFDF ECACACAC
[02022007 18:46:29 module] 0x0040  41 43 41 43 41 41 41 00  00 00
00 00              ACACAAA. ....
[02022007 18:46:29 module] =------------------------------------
-----------------------------------=
```

Example log portions from an active Nepenthes sensor. Nepenthes is able to classify payloads into their protocols and families and discern if they represent a known attack.

Discovering which botnets to track is another challenge, although one that is easily overcome. IRC-based botnets often use the IRC protocol's capabilities to protect the server and channels with passwords to block unwanted visitors. However, the malware

sample itself will carry these credentials with it, and analyzing the malware can discover these parameters. Antivirus write-ups often do not contain the necessary information about the server, port and passwords needed to infiltrate the botnet, meaning that the botnet investigator will have to analyze the malware on their own. This then necessitates independent malware analysis.

Often this information is gathered by automated dynamic analysis of the malware, using tools often dubbed a malware "sandbox" [9, 10]. When run in such an environment, the malware will use these credentials to attempt to join the botnet. Because the environment is instrumented to log all actions, these details are available as part of the report as shown below. Although they can be useful for many things, sandbox reports fail to reveal the advertised version of the bot (visible if the botnet operator were to query it using a "CTCP version" request).

```
 [ Network services ]
    * Looks for an Internet connection.
    * Connects to "billybobpizza.biz" on port 22345 (TCP).
    * Connects to IRC Server.
    * IRC: Uses nickname |803400.
    * IRC: Uses username ezkieyac.
    * IRC: Joins channel ##j,##jdownload with password
cannot enter.
    * IRC: Sets the usermode for user |803400 to +x+i.
    * Attempts to delete share named "IPC$" on local system.
    * Attempts to delete share named "ADMIN$" on local system.
    * Attempts to delete share named "C$" on local system.
    * Attempts to delete share named "D$" on local system.
```

Part of a Norman Sandbox analysis of a malware sample. This sample was uploaded on April 11, 2006, to the Norman Sandbox Live website (http://sandbox.norman.no/), a public resource provided by the Norman ASA company to demonstrate their Sandbox product. Similar reports with this kind of information are available from similar tools and products.

Once these credentials are gathered, the botnet infiltration tools can be targeted against the botnet. It is tempting to consider using modified bot binaries, made by modifying the source code to remove the attacking functions, to maximize the illusion that the monitor is a real infected host. Although this is easy if the source is readily available, the number of minor variants to the popular bot families means that an extensive number of modifications must be taken into account to reliably mimic the bot's behavior. Furthermore, these modifications must be made for every different bot operator who has customized the software. In the end, a custom client is simply far easier to use.

Standard IRC clients such as irssi [11] and mIRC [12] cannot be used to monitor the majority of the interesting, private server botnets for a number of reasons. Firstly, upon connecting to a server the software will often issue a series of commands, such as listing the channels and clients. If the botnet operator controls the IRC server, they can spot the

actions sent by the client upon its connection and usually become suspicious. Some IRC servers used by botnet operators will send an alarm to the IRC operators when these commands are used. Bots do not perform these actions, and usually the botnet managers know all people who connect to their IRC server. Secondly, the IRC client can be fingerprinted using the CTCP "version" request. These version strings are usually different from the standard bot versions announced. Finally, when modified IRC servers are encountered that do not behave in a standard way, these IRC clients will fail to connect reliably.

Bladerunner

To achieve our project's primary goal, we have developed a small Python program we have dubbed "Bladerunner" - a script that monitors botnets. We chose this approach for several reasons. Firstly, the use of a scripting language made initial development easy and provides flexibility so that new features can be added without much difficulty. Secondly, Python is a relatively secure language, with IRC bot functionality already implemented using the standard barebones library called IRClib [13], which we have slightly modified to augment functionality specific for botnet activity. Finally, Bladerunner is multithreaded, allowing us to monitor an arbitrarily large number of botnets in real-time with minimal load on the host system. We have tested Bladerunner with over 100 simultaneous, sustained IRC connections from a single client instance.

Bladerunner is designed to infiltrate and actively monitor botnets for long periods of times, so any such client must be difficult for the botnet managers to identify or suspect. One of the major design principles for the client is to prefer silence to mistaken replies. To evade detection, Bladerunner responds to common botnet commands like ".login" and ".join" and also sets the correct nicknames, usernames, and modes for the client. The tool is configurable for all relevant information for each botnet, including the C&C server's address and relevant channels and passwords. Bladerunner can also perform some botnet server inventory, such as making a list of channels and known clients by issuing standard IRC commands. However, this option is disabled by default due to the increased suspicion it brings to the client. We typically operate Bladerunner in this inventory mode from separate addresses.

Bladerunner logs all messages sent to it and all server actions it receives, and it can also recognize URLs in botnet communication (which are usually updates to the malware). The client automatically downloads them for future disassembly, and indicates from where the file was downloaded. A newer version of the Bladerunner client automatically identifies attacks and catalogs the activities of various botnets.

We operate Bladerunner from a Linux host within a virtual machine environment to minimize security concerns. Additionally, to mask its true origins, we use addresses in a source network that is not registered to Arbor Networks. We have also used other addresses from around the world to examine botnets using standard IRC clients, hoping

that the distance between the addresses used by the scripts, the information gathering phase, and our true location will be difficult for the botnet manager to triangulate.

Bladerunner is not publicly available.

Other Considerations for a Botnet Tracking Client

IRC uses a multipart client identifier. This includes the "ident" string from the client, the source host address, and a nickname. Bots typically generate the nicknames using a system to represent their geographic location (inferred from the language pack on the system), their operating system, and sometimes their uptime, which is usually combined with a random number of string of characters [3, 4]. These random identifiers change every time the infected host executes the bot malware. A common mistake for a botnet tracking tool to make is to use a static nickname string, which is obvious if the tracking tool disconnects and reconnects automatically. This kind of behavior is obvious to professional botnet managers and will lead to the tracking tool becoming banned from further monitoring.

```
[0|USA|221038]
USA|59138
XP R`767253216
USA|803
```

Example IRC bot nicknames presented in live botnets.

Just as important is the source address from which the tracking tool originates. Clearly any origin should be masked as much as possible from the real tracking tool managers, and a diverse set of sources should be available, as well. While it is tempting to use a proxy network such as Tor or CoDeeN, this is obvious to many skilled botnet operators and often these hosts are banned from joining the network. As a complement to this fact, botnet tracking tools that come from "interesting" sources, such as .mil or .gov addresses, can be used to determine if these bots are being targeted for more specific purposes.

It is also important to have variable source addresses available for large-scale tracking efforts. Botnet operators or other trackers have approached us after seeing the same source IP addresses across multiple networks that turned out to sometimes be the same botnet on different servers. Similarly, any tracking host that is discovered should expect to receive a DDoS attack as retribution from the botnet operators and a demonstration that they disapprove of being monitored.

For any anthropological botnet tracking purposes, a valuable soft skill worth possessing is the ability to understand a foreign language. Several large botnet operator groups commonly speak languages other than English, including Spanish, Romanian, Russian, or Portuguese. The ability to read and write these languages, often using underground idioms or improper spelling, is useful in conjunction with a cover story if the botnet tracking operator is discovered. Some botnet operators seem to accept an interloper if

they feel that they are not a threat, which would be someone from either law enforcement or a rival botnet.

Finally, it is worth remembering that many like-minded botnet trackers and possibly law enforcement personnel will monitor many of the better-known botnets and web forums where they communicate. Experience has shown that many investigations have, in fact, been hampered by the fact that two botnet trackers were talking to each other, each pretending to be a "bad guy".

Changes in the Botnet Landscape

As expected, with all of this interest in botnet operators' behaviors, and with increased competition, the community is changing their tactics to hide from each other. We have seen this change become more pronounced in the past year as competition heats up, including competition between spam bot groups and rival botnets. Some of this is akin to the Netsky-Mydoom-Bagle wars of early 2004, but the severity of these attacks is growing in sophistication.

The changes we have begun to see have already had a dramatic impact on the utility of common botnet tracking methodologies. As honeynets and automated malware analysis tools gained popularity, botnet operators began to share information about instrumented networks and encouraged others to stay away from them. In some cases, specific malware variants are launched into these networks to determine if they are indeed instrumented. If a botnet tracker appears or if a public sandbox report contains the variant's unique information, then the botnet operator knows that they have run into a monitored network and will move on. Additionally, more botnet operators and malware authors are using more anti-analysis methods to slow down malware analysis. Finally, we are have seen the growth rate of non-IRC botnets increase, and these new networks often use custom built protocols over HTTP or a peer-to-peer network.

In this section we describe some of these changes that have appeared and discuss their implications on botnet tracking efforts.

Increased Use of Anti-Analysis Methods

A growing number of malware samples we are encountering are display and anti-sandbox methods. Some of these are well known techniques to detect or defeat common reverse engineering tools placed inside the main code. An example of such detection is shown below in the function `CDebugDetect::IsDebug()`, which looks for common debuggers, single stepping in a debugger, VMware, or other suspicious environments. This specific routine is called at the top of the main program entry point, although other variants have used some of these techniques inside the main event loop of a program.

```
int CMainCtrl::MainCtrl()
{
…
#ifdef _WIN32
        // Detect if being debugged
        if(m_cDebugDetect.IsDebug()) {
#ifdef DBGCONSOLE
                m_cConsDbg.Log(5, "Being debugged, exiting...\n");
#endif // DBGCONSOLE
                m_bRunning=false;
                ExitProcess(1);
        }
#endif // _WIN32
…
}


bool CDebugDetect::IsDebug() {
#ifdef _DEBUG

        return false;

#else

        if(m_bIsDebug) return true;

#ifndef _WIN32
        // Anti-PTrace
//      if(ptrace(PTRACE_TRACEME, 0, 1, 0)<0) {
//              m_bIsDebug=true; return true;
//      }
#else
        pfnIsDebuggerPresent IsDbgPresent=NULL;
        HMODULE hK32=GetModuleHandle("KERNEL32.DLL");
        if(!hK32) hK32=LoadLibrary("KERNEL32.DLL");
        if(hK32) {
                IsDbgPresent=(pfnIsDebuggerPresent)GetProcAddress(hK32,
"IsDebuggerPresent");
        }

        FoolProcDump();

        unsigned long lStartTime=GetTickCount();

        if(IsBPX(&IsBPX)) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "Breakpoint set on IsBPX,
debugger active...\n");
#endif // DBGCONSOLE
                m_bIsDebug=true; return true;
        }

        if(IsBPX(&IsSICELoaded)) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "Breakpoint set on
IsSICELoaded, debugger active...\n");
#endif // DBGCONSOLE
                m_bIsDebug=true; return true;
        }

        if(IsBPX(&IsSoftIceNTLoaded)) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "Breakpoint set on
IsSoftIceNTLoaded, debugger active...\n");
#endif // DBGCONSOLE
```

```
                m_bIsDebug=true; return true;
        }

        if(IsBPX(&IsVMWare)) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "Breakpoint set on
IsVMWare, debugger active...\n");
#endif // DBGCONSOLE
                m_bIsDebug=true; return true;
        }

        if(IsSoftIceNTLoaded()) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "SoftIce named pipe
exists, maybe debugger is active...\n");
#endif // DBGCONSOLE
                m_bIsDebug=true; return true;
        }

        if(IsSICELoaded()) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "SoftIce is loaded,
debugger active...\n");
#endif // DBGCONSOLE
                m_bIsDebug=true; return true;
        }

        if(IsVMWare()) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "Running inside VMWare,
probably honeypot...\n");
#endif // DBGCONSOLE
                m_bIsDebug=true; return true;
        }

        if(IsDbgPresent) {
                if(IsBPX(&IsDbgPresent)) {
#ifdef DBGCONSOLE
                        g_pMainCtrl->m_cConsDbg.Log(5, "Breakpoint set on
IsDebuggerPresent, debugger active...\n");
#endif // DBGCONSOLE
                        m_bIsDebug=true; return true;
                }

                if(IsDbgPresent()) {
#ifdef DBGCONSOLE
                        g_pMainCtrl->m_cConsDbg.Log(5, "IsDebuggerPresent
returned true, debugger active...\n");
#endif // DBGCONSOLE
                        m_bIsDebug=true; return true;
                }
        }
```

```
        if((GetTickCount()-lStartTime) > 5000) {
#ifdef DBGCONSOLE
                g_pMainCtrl->m_cConsDbg.Log(5, "Routine took too long to
execute, probably single-step...\n");
#endif // DBGCONSOLE
                m_bIsDebug=true; return true;
        }
#endif // WIN32

        return false;
```

```
#endif // _DEBUG
}
```

Source code listing from a Phatbot codebase. This is a routine that is a top-level entry to detect the presence of a suspicious environment, i.e. a malware analyst's instrumented workstation. When this executable runs in some environments, it exits before it can demonstrate any useful properties such as IRC credentials.

Modified or updated executable packers have also become a popular way to defeat analysis, either with a built-in call to basic debugger check routines or other code that is used to defeat the emulation used by some sandboxes. One of the most popular executable packers that causes difficulty for sandbox environments is the Themida packer [14]. This can lead to a situation where the analyst must perform manual analysis on the sample to discover what it will do when it executes on a host, or invest more resources to monitor the infected host externally.

In either case, the effect is to slow down the analysis of the bots. Now that this toolkit is available as a pluggable module, more malware authors are able to use it and the number of samples using this packer and related packers is growing. The net effect is to make malware analysis more labor intensive, giving the botnet operators more time to alter the bot's software or behaviors.

Communications Protocol Changes

Many of the newer botnets that have migrated away from IRC are more particular about their clients and will actively blacklist any host that does not actively speak the protocol correctly. In the case of HTTP bots, this can include the client's User-Agent field in the request as well as the arguments passed to the server. The server may react by blacklisting the client by IP address or by serving up the wrong data.

Recent advances in specialized botnets have highlighted how difficult this problem has become. The botnets formed by the Nugache worm, Rustock Trojan, the Storm worm (which uses the eDonkey protocol to communicate), and Warezov Trojans have all used specific, sometimes custom communications mechanisms wildly different from IRC. These botnets have been built to create a global, takedown-resistant network used to send spam. Infected hosts communicate with the botnet handles to receive information about what spam to send. Furthermore, this communications network means that additional binaries can be loaded onto the system. Both the Nugache worm and the Storm worm have used peer-to-peer communications to propagate updated binaries. This makes mapping and takedown of the whole network difficult and traceback of the malware to an origin even harder.

The most direct way to "spy" on these new, specialized networks is to build a honeypot that actively participates in them, and then monitor them using either external tools (i.e. a traffic sniffer) or host-based tracing tools. This approach has obvious limitations,

including scalability and risk; however, it has proven effective at understanding how the Rustock botnet works [15].

These two increasingly popular network changes, HTTP-based bots and peer-to-peer bots, also obscure any direct interactions with the botnet operators inline with the botnet. This makes profiling the operators of these networks more difficult and means that any intentions, skills, or identities of the operator must be inferred from the malware itself or other information left behind by the operator.

Project Results and Conclusions

Botnet monitoring provides a way to directly measure a a portion of Internet-scale threats such as DDoS attacks and scanning activity. Disabling a large, active botnet can stop crippling attacks at their source, so it is natural to ask if botnets can be stopped from forming, or at least from growing into substantial threats if enough malware is caught and their botnets tracked.

DDoS Attack Visibility by Direct Botnet Monitoring

One of our primary goals in launching a botnet tracking project was to determine if it we could trace DDoS attacks back to their originators. In assisting customers with DDoS events, it has proven useful to be able to determine which botnet has launched the attack and disrupt its activities by blocking access to the IRC server or by disabling the bots themselves. To that end, we evaluated the coverage of actively launched DDoS attacks observed on a botnet and how many were observed on the Internet.

We used two data sources for this analysis, one representing direct measurement of global DDoS activity and one representing a similar scale of active botnet network tracking. Arbor Networks' Peakflow SP installations have an optional module to provide anonymous DDoS statistics from global service providers [16]. The ShadowServer Foundation provided Botnet activity logs and yielded extensive DDoS command history, showing the botnet command server and channel as well as the exact DDoS commands issued [17]. To determine the amount of coverage of global DDoS activity, we counted the number of DDoS attacks that showed an intersection between the two data sources. To determine if the attack was likely to be the same attack, we looked at the rough times of observation in the two systems, the target IP addresses, and the type of attack (i.e. TCP SYN floods or ICMP floods). Attacks that showed these criteria between the two data sets were considered to be the same attack and counted as a match.

The results of this intersection showed that active botnet tracking is only a moderately effective way to discover the origins of a DDoS attack. In this study, we analyzed over 102,000 DDoS attacks that were logged by the Peakflow SP DDoS statistics system, and over 21,000 from the ShadowServer Foundation logs. When comparing the intersection of the two attack sources against the botnet logs, approximately 13% of the attacks were

seen by Peakflow SP systems around the world. Approximately 2% of the DDoS attacks seen on the Internet were recorded by the botnet tracking operations from ShadowServer.

The disparity between the two data sets comes from multiple sources. For direct measurements from Peakflow SP systems, there are three considerations: coverage, the anonymous nature of the data, and the tuning of the deployment's sensors. Clearly not all possible targets of DDoS attacks are monitored by Peakflow SP systems, meaning that some attacks will go unseen. Also, not all Peakflow SP deployments participate in this experimental data collection program. The data is anonymized to protect the identity of the data source, with the deployment's local half of the data obscured. Because of this, if the target of a DDoS attack was obscured it would not match the attack command observed on an active botnet. To be counted as a global DDoS event, the Peakflow SP sensor must classify the traffic as an attack. Not all DDoS events launched register enough traffic to cause the intended target traffic problems, so not all intended attacks are registered as attacks by the victim.

For the direct observation of the DDoS commands that come from botnet tracking, the two major dataset limitations come from completeness of coverage and the interpretation of the commands. Clearly the number of monitored botnets is fewer than the number of active botnets, and botnet operators who actively launch DDoS attacks have an incentive to keep lurkers away, thwarting any long term tracking operations. Additionally, the commands issued on a botnet channel to launch a DDoS may not be properly classified. The following is an example DDoS command that was launched as part of a massive DDoS against the Prolexic company in the spring of 2006:

```
Sun Apr 23 21:54:08 2006 pubmsg sadf!tsinternetuser@room #usa#
['.tusa ack 72.52.6.3 8080 20']
```

This command structure, ".tusa ack" does not match any common DDoS attack format and was customized by the botnet software operators.

All of these factors combine to yield the low percentage of matching attacks between the two methodologies in our limited experience. However, the general trends are clear: it will take a more significant botnet tracking effort to observe a sizable (more than 30%) number of the world's DDoS attacks launched from IRC-based botnets, useful in both forensic efforts as well as for DDoS attack response. Our own previous efforts to track botnets showed that roughly 50% of the botnets we monitored launched at least one DDoS attack [7]. In most cases, however, these were not high profile attacks and did not show up on global activity logs.

Evaluating Botnet Tracking as a Solution

"Solving" the botnet problem is a very difficult task, and unfortunately, the problem will likely get worse before it gets better. Furthermore, the criteria for evaluating a solution depend largely on the goals. In the case of profiling malicious botnet operators, tracking

their activities is an obvious key data source that when combined with botnet manager website activities provides a more complete picture forms of the people behind botnets. As shown above for DDoS attack data, monitoring botnets for forensics purposes has its uses, but requires a significant undertaking to collect information that will be useful in the future.

However, there are a few things the community can do to help. Internet service providers need to be more proactive and responsive, as well as registrars and those who offer DNS services. Too often, even when confronted with clear evidence of an operational botnet, many hosting providers refuse to take the very simple action of blocking that computer or domain name from being accessible. It is important for companies to learn how to be good Internet citizens. Also, international law enforcement needs to be streamlined and attempts at uniformity in process should be encouraged. Investigations into botnet activity are too often abandoned due to difficulty of finding any legal authority to deal with Internet crime in the specific country. In the end, we hope that botnet tracking projects will aid in these efforts by providing a clearer picture of the Internet's enemy.

In most cases, the direct network or facility operator is the most desirable person to actively shut down a botnet. This can include a hosting facility administrator, an ISP security operations official, or a system administrator if the host is compromised. In the cases where the botnet uses DNS records to locate its C&C hosts, takedown requires an additional participant, the DNS registrar, to respond by locking the DNS records. In somes cases the DNS registrar is not as responsive as the network operator and through their lack of participation in the security community, the botnet can persist. In some situations, the DNS registrar themselves is the victim of compromise, and any changes they make are easily undone by the botnet manager or the their team. Without full participation of all parties, the botnet can persist.

Overall, the issue of response continues to be the most pressing challenge in dealing with the botnet problem as a whole. Because the response is always a few hours (or even a few days) behind the introduction of the problem and so much labor is involved in taking down a botnet, the problem will persist for the foreseeable future.

References

1. Will Sturgeon, Internet guru warns of botnet pandemic, ZDNet UK, January 29, 2007. http://news.zdnet.co.uk/internet/0,1000000097,39285665,00.htm

2. The Honeynet Project and Research Alliance. Know your enemy: Tracking botnets. http://www.honeynet.org/papers/bots/, 2005.

3. Nicholas Ianelli and Aaron Hackworth, Botnets as a Vehicle for Online Crime, CERT/CC Technical Report, 2005.

4. Barford, Paul; Yegneswaran, Vinod. An Inside Look at Botnets, Special Workshop on Malware Detection, Advances in Information Security, 2006.

5. E Cooke, F Jahanian, D McPherson, The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets, USENIX SRUTI Workshop, 2005.

6. Felix C. Freiling, Thorsten Holz and Georg Wicherski, Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks, 2005.

7. J. Nazario, J. Linden, Botnet Tracking Techniques and Tools, Virus Bulletin Conference, 2006.

8. Paul Baecher, Markus Kötter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID), Sept. 2006.

9. Norman Sandbox Information Center, http://sandbox.norman.no/.

10. Carsten Willems, CWSandbox, http://www.cwsandbox.org/.

11. irssi- The Client of the Future. http://irssi.org/.

12. mIRC Resource Center - http://mirc.org/.

13. Python IRC library - http://python-irclib.sourceforge.net/.

14. Themida: Advanced Windows Software Protection System - http://www.oreans.com/ThemidaWhatsNew.php.

15. A Rustock-ing Stuffer, Joe Stewart, SecureWorks Blog, January 10, 2007. http://www.secureworks.com/research/blog/index.php/2007/01/10/a-rustocking-stuffer/

16. C. Labovitz, D. McPherson, Peakflow SP Anonymous Statistics Project, private communications.

17. Shadowserver Foundation – http://www.shadowserver.org/.