

AIM FOR BOT COORDINATION

Lysa Myers

McAfee Security, Inc., 20460 NW Von Neumann
Dr., Beaverton, OR 97006, USA

Tel +1 503 466 4311

Email amyers@avertlabs.com

ABSTRACT

In the last few years, there has been increasing interest within the virus-writing community in Internet Relay Chat (IRC) based malware, due to the power afforded by the IRC scripting language and the ease of coordinating infected machines from a chat-room type of structure. What has developed is a very modular, open-source sort of threat which is very rapidly adapted to include new functionality and new infection vectors.

More recently, there has also been an increase in the number of threats spreading through Instant Messaging (IM) clients, particularly OSCAR-based clients like *AOL Instant Messenger (AIM)*. IRC bots have begun using this functionality to spread, but there is more capability available within OSCAR than is currently being exploited.

As there has also been an increase in the number of bots using Command and Control (C&C) channels that utilize something other than IRC (primarily web-based currently), it stands to reason that there may be a possibility of virus-writers using OSCAR as a means of control. This paper looks to explore the capabilities of OSCAR for being used in C&C scenarios, and what steps could be taken to mitigate this proactively.

IN THE BEGINNING

Internet Relay Chat was one of the first methods for Instant Messaging/chat over the Internet. The architecture was intended to be very open and very powerful, with the user being able to control almost anything within the channel. As the popularity of IRC increased along with the size and traffic of individual channels, channel operators felt the need for automated mechanisms to monitor and control their own channel, including kicking out or banning people the channel operator deemed problematic.

Separate executables, as well as IRC's own scripting language, were used to perform these channel-control functions. Eggdrop is an early example of a bot which became widely used to help with this automation of channel operations.

Naturally, problematic users felt the need to gain retribution against those who'd removed them from the channel. The most popular method of combat was to try to interrupt the functioning of the channel operator's bot, by way of Denial of Service (DoS) attacks. Malicious users would hack into channel operators' machines and replace the operator's bot with their own so that they now owned the channel. This began an arms-war of sorts, between IRC operators and those who were using IRC scripts maliciously.

IRC networks began to include more protection against the flooding that was being used to enact these DoS attacks. This meant that making the attack come from more sources became essential, to evade this filtering. This escalated from taking over channels to taking down entire IRC servers with

Distributed Denial of Service (DDoS) attacks.

In order to perform a DDoS, attackers needed to get their malicious bots onto other machines. To get enough machines to take down large servers, they needed much larger numbers of machines to be part of their effort, which meant tricking innocent users into having their machines run the malicious code. These bot-infected machines could then be used to perform massive DDoS attacks.

Realizing the capability of their powers, the malicious users then began attacking websites such as *Amazon*, *Yahoo!*, and *eBay*. Infected machines became a commodity to be fought over by malicious users, and malicious users began to employ techniques to wrest infected machines from each other.

As the demand for infected machines increased, it became advantageous for a bot to have spreading capabilities as well as having a means for controlling the large numbers of infected machines remotely.

Spreading capabilities came in the form of scanning for machines that were shared through *Windows*, to an infected machine. Lists of common usernames/passwords were included, to give the bots the ability to spread to password-protected shares as well as open shares.

Remote control came in the form of specific IRC channels set up for use as a Command and Control (C&C) medium. Here bot-herders could give specific commands to infected machines to tell them to begin attacks, spread, and eventually a huge number of other commands.

Concurrently with the earliest bot developments, initial experiments in IRC worms were also beginning to take place. The first attempts, as with other types of early email/network worms, were mostly intended for spreading for spreading's sake, perhaps showing a message box to the infected user. Many of these (like IRC/Dmsetup) accomplished their ends by modifying the *Script.ini* for *mIRC* in order to entice other users into accepting a file sent by Direct Client-to-Client messages (DCC).

These DDoS-type bots and the DCC-spreading viruses eventually began to be combined, so that the bots could be propagated more automatically. Bots such as *GT-Bot* combined a wide variety of tools to hide themselves or gain information about infected systems. This information often included details of the OS/applications used on the machine and the speed of its Internet connection.

In 2002, *Sdbot* was created. This began to combine all these functions in one single executable. Initial variants were trojans which had spreading commands, but soon these too were able to spread without being instructed to do so.

Subsequent variants started including spreading capabilities through peer-to-peer (P2P) applications such as *Kazaa*, scanning for *Windows* and application vulnerabilities to get in through, and eventually even bolting code from other viruses such as *Mydoom* which gave it email spreading ability. Inevitably, spreading by Instant Messaging clients such as *AIM* was included as well.

After *Sdbot*, myriad other families of IRC bots were created, one of the largest of which was the *Gaobot* family. *Gaobot* was the most 'professional'-style programming effort of IRC bots to date, being the most modular and well-commented, and it is frequently updated with some of the most notable examples of new technology in bots.

One of these new Gaobot variants is now called Phatbot, which incorporated P2P C&C mechanisms. An infected machine utilizes the *Gnutella* cache servers to register itself as a *Gnutella* client. Other infected machines then check in with these cache servers and find other infected machines, which are differentiated from legitimate *Gnutella* clients as they use a different port than the standard. A user who had the proper username and password could then control the network of infected machines. The primary problem with this family of bots is a problem of scale – it is not intended for the several tens of thousands of connections that are often seen in IRC bots.

Nugache is another recent IRC bot (though not in either the Sdbot or Gaobot family) which uses P2P C&C but with encrypted traffic, which could potentially have made filtering for its traffic much harder except that it used a very unusual port.

THE LANDSCAPE NOW

At this point, two main developments are shaping the war over bots. The malicious bot users have come to appreciate an increasing number of financial advantages of having large networks of infected machines. And on the other side of the fence, an interdisciplinary community has sprung up to find and take down C&Cs more quickly and permanently.

Bot ‘owners’ can use infected machines to install adware, send spam and steal passwords, engaging in extortion as well as DDoS attacks, in addition to the other myriad benefits of having near-total remote control capabilities on an infected machine. This has created a sort of open-source programming community around the development of the major bot families. New methods of hiding and spreading are added on a very frequent basis.

As the source code for the major bot families is readily available to the general public, it has been very easy for malicious users to create huge numbers of slightly different variants. As a result of this, the security industry has been able to create powerful generic, heuristic and behavioural scanning techniques to detect these bots based on the similarities between all these variants.

To combat this, bot authors began systematically to explore ways around each of their weak points. One of the first techniques that was used was against string-based AV scanners. Bot authors would pack files using run-time packers, so that a large number of slightly different files could be created in a very short period of time. Many variants also started to include the ability to disable security software, including firewalls or standard *Windows* utilities which would help diagnose infection such as Regedit. To target security researchers specifically, many bot variants have also added detection of *VMware* and debugging software.

From there, stealthing of files continued to get even more complex, with bots including separate kernel mode rootkits to be dropped (and now even the recent proof-of-concept creation of a kernel-mode bot, Rootkit-KIRcbot).

In response to this ever-increasing onslaught, administrators began to use more and different kinds of security software to protect their environments, including using firewalls to shut off ports which they weren’t using and sniffers explicitly to detect anomalous types of traffic. This allowed administrators to identify IRC traffic even when it was sent over non-standard ports, or ports commonly used for other

protocols such as HTTP. This has effectively shut off IRC communication in many environments.

This filtering has prompted the creation of C&C channels which are not IRC-based. At the present time the most popular alternative method is HTTP-based C&Cs. The possibility of slick, intuitive interfaces is a big draw for this method, as well as being traffic for a very common protocol.

With all these stealthing techniques many of these new variants may initially go undetected by a significant portion of AV vendors, which buys them at least a day or two of largely uninterrupted operation. But then, even as detection for these variants is added, this still only removes a portion of infected machines – as long as the C&Cs remain, they can still continue to operate.

CURRENT WEAK SPOTS

The remaining problems in the bot world surround hiding traffic and maintaining control over infected machines. Some discussion has occurred within the virus-writing community with regards to using encrypted traffic and steganography to make traffic less visible.

Hiding IRC traffic is a significant problem as IRC is not a commonly used application in most offices, and even at home or in schools, its traffic is reasonably distinctive such that sniffing network traffic for its presence is a relatively trivial matter. Once this unintentional traffic is detected, infected machines can be tracked down and disconnected. HTTP lacks this problem, as one would be hard-pressed to find an office, home or school that never uses the Internet either directly or through various applications which use it to ‘check in’ in some capacity.

With both IRC and HTTP-based C&C channels, there is still a single point of presence, which means that it is relatively easy to sever the communication mechanism between bot and master. While there has been some experimentation into P2P C&Cs, it is yet in its infancy.

Both Phatbot and Nugache have delved into this, but both have their limitations. Nugache uses a very uncommon port and it is unable to accept many connections, so monitoring and preventing its traffic is still very simple. Phatbot uses an unencrypted protocol based on WASTE, which can also be easily filtered, and again it will not accept a large number of connections, which means that botnets must stay so small as to be useful only for the most targeted attacks.

WHAT’S ON THE HORIZON

At this point, it is likely that virus-writers will continue with small runs of subtly-modified batches of worms, packed with newer and better packers, to keep ahead of string-based AV detection. It’s also likely that they will continue trying different methods of C&C structures including P2P techniques. But they have not yet found a way to have a large number of connections – Instant Messaging clients make a particularly tempting target for this effort.

One single user can be connected via IM to a moderately large number of other users, who can in turn be connected to other users. This can go on almost indefinitely – there are few users who are totally unconnected from the rest of the IM network, and those presumably because that ID was never actually put into use.

Due to things like chat and buddy list limits as well as SPIM-filtering controls, there are a few difficulties in getting a sizeable-enough complement of infected machines. SPIM filtering was set up on the IM servers to keep ads as well as viruses from being distributed by IM. This is intended to prevent individual users from sending too many messages at once, or any number of users sending a large number of similar messages. Chat limits keep chat rooms from getting unwieldy, so they top out at no more than a few dozen. Buddy limits can go into the hundreds, so this could be much more desirable. IM applications also put limits on the number of windows or specific user IDs in use at any one time.

The most important thing that will need to be adopted by bot authors will be a more vertical structure than is currently used. The controller may only be able to send a very small number of messages, but each of these recipients can then spread a small number of messages to their own contacts, and so on. A controller can also use a number of different IDs to send messages, so this could potentially increase the number of recipients as well. In short order, this could reach a very large number of infected users.

Some variants of the popular bot families already have the capability to 'elect' more powerful machines to help herd the others, so this mechanism has already been explored to some extent. Encryption or personalization of commands will also be necessary in order to evade filters, including SPIM filtering. Frequently rotating buddy-lists and controller IDs could also help in evading SPIM filtering.

These techniques are as much a concern for IM clients like *Yahoo!* or *MSN* messengers, as they are for *AOL Instant Messenger (AIM)*. The primary danger for *AIM* comes purely from its popularity, though *MSN* is still a close second as *Microsoft* software is a favourite target among virus writers.

AIM currently holds more than 50% of the market share (over 50 million users), according to the latest *Nielsen/NetRatings* figures. Virus writers have already explored spreading via *AIM* extensively, so they are familiar with the OSCAR protocol, and are more apt to go with something they know how to use already. There are quite a few variants of the oldest and most popular bot family (*Sdbot*) which have included this spreading capability.

There is some indication that *Yahoo!* and *MSN* messengers are gaining in popularity and there are also quite a few worms which spread through *MSN Messenger* in particular, so this should not be indication that these other clients should be considered immune.

MITIGATION

Many of the techniques that will be useful in preventing these potential IM bots will be the same as for preventing the existing IRC bots and other malware; Firewalls with all unnecessary ports shut off, Intrusion Detection Systems, vulnerability scanners, traditional anti-virus software.

It can also be helpful in a corporate environment to use applications which will allow the use of in-house IM servers so that employees can still enjoy the benefits of immediate conversation that IM provides, without having to allow IM traffic to or from the Internet.

As new bots are created which use IM clients to be controlled or spread, researchers can assist the makers of IM

applications by giving their developers information about specific or generic techniques used by these bots. IM makers could then implement a challenge/response system to verify that a real human is sending the messages, when anomalous traffic patterns are found.

If this becomes a more prevalent problem, users may choose to use something like a white-list to accept messages only from known contacts (or, to borrow a concept from social-networking sites, friends of known contacts). This will not stop bot-spreading, but it could potentially affect the prevalence C&Cs.