

QuiGon : the First Tool Against Clone Attack on Internet Relay Chat

Thibaut Henin and Corinne Huguennet

arsouyes.org,
{tbowan,aryliin}@arsouyes.org,
WWW home page: <http://arsouyes.org>

Abstract. Clone attacks are a way to perform DDOS attack on Internet Relay Chat networks. There is no tool which prevent an IRC network against such attacks despite some handcrafted methods which are not so efficient.

Here, we present qui-gon, the first tool which defeat clones attacks. It use a temporal oracle which detect attacks by measuring the connection rate; and another oracle which distinguish clones using probabilistic automata. These oracles learn the characteristic of the network in order to fit better to the network.

1 Introduction

Deny Of Services Attack (DOS) is a kind of easy yet powerful attack. The aim of such attack is to make a system, a service or a website not working as it should. This can be done by sending some handcrafted packets or too many request to the server. This method is quite popular and simple [6–8].

When they are distributed, it's more difficult to detect and prevent them. Such attack is performed by compromising a lot of hosts and then order them to attack any service. Because of the many-to-one aspect of this kind of attack, common techniques can not be applied [6, 7].

These Distributed Deny Of Services (DDOS) also occurs on Internet Relay Chat and are called *clone attacks*[8]. The attack is performed by connecting lots of clones to a targeted server. The effect range from a server crashing down to networks split.

Here, we present Qui-Gon, the first tool against clone attacks on Internet Relay Chat servers. Despite two handcrafted methods, there is no released tool against such attack. Our method can detect the attack and recognize clones.

In the first part, we will present Internet Relay Chat and the clone attack. Next, we will explain some usual protections. We will then introduce Qui-gon and the methods used to detect the attack and clones. Finally, we give some test results and a conclusion.

2 Clone Attack on Internet Relay Chat

2.1 Internet Relay Chat

Internet Relay Chat (IRC) is a form of real time communication [1]. It's mainly used for group and broadcast communications. Once connected to a server, users join so-called channel where they can communicate with the users presents in this channel. IRC is also used for private communications, file sharing and games.

Users are identified by their nicknames. Once connected to a server, a user chooses his nickname and sends it to the server. After that, he can join channel, speaks with others, and so on. Since many users can use the same public IP address, it can not be used to identified any user.

An IRC network is made by connecting servers to form a tree. Each server is responsible of some users and channels and routes any information to each concerned server. This network architecture was chosen to avoid duplication but, obviously, a server crash cause a network split. This lead to two disconnected IRC networks with duplicated channels.

In addition of simple users, there are so called IRC-operators. These users have privileges on the entire IRC networks and the servers. Their job is to administrate the network and keep it working. For example, they can disconnect users, ban users and IP addresses.

The first user who joins a channel gains the ownership of this channel. This first user is called the founder of the channel. The founder gains the most privileges on the channel and becomes a *channel operator* and can give same privileges to others users on the channel.

If a channel becomes empty, the first user to re-join the channel gains the ownership.

2.2 Bots and Botnets

Bots (abbreviation of "robots") are programs connected to any network [10, 13]. They perform simple and repetitive tasks such as searching the web and indexing web-sites. They can also replace players in online games.

On IRC, these bots are very commons. There are service bots, whose purpose is to maintain some order [4] such as *NickServ* which register and protect nicknames. Some bots provides more general features such as weather-reports, quiz-games or conversation finding [5]. Finally, there are some bots trying to pass the Turing test by speaking with users like *ALICE* [14].

Botnets are mostly networks formed by IRC bots which are waiting for commands [11]. These bots are installed in compromised hosts and connect themselves to an IRC network and join a specified channel where their Master can order them some actions. These actions vary from compromise others hosts, update their code and make a DDOS attack against another host [10, 13].

2.3 Clone attacks

A clone attack is when an attacker mastering a botnet order his bots to DDOS a server of the IRC network. The bots make multiple connections to the server (these connections are called "clones") [8, 10, 13]. Because for each connection, the server keep lots of informations, there is no need for too many clones to crash down a server.

Clone attacks are used to shutdown an IRC network, or to take-over a IRC channel. These kind of attack is an easy way to crash down a server. So, it's an easy way to disturb users and IRC-operators. Once the network split occur, the attacker may join some channels and gain ownership.

These clone attacks can't be defeat only by IRC-operators. A clone attack is too brief and IRC-operators don't have time to respond it. These attacks are also too intense. So, IRC-operators notice the attack only after its effect.

There are also clone attacks against channel instead of servers. In this case, the bots are already connected to a IRC server. When the attacker ask them, they simply join a specified channel and some time flood it (by sending lots of messages). This channel is then unusable.

This kind of attack can easily be defeat and is then out of the scope of this article. To avoid bots to send message on a channel, one can simply set the mode "m" (moderated) to the channel. This mode allow sending message only to users with the mode "v" (voice). This way, privileged users can set manually this mode to legitimates users they know, or ask chanserv to do this automatically. Thus, in the next sections, we will only focus on clone attacks against servers.

2.4 How to prevent such attacks ?

Obviously, IRC-operators need an automated tool to detect, prevent and defeat clones attacks. Such tool should act just after the connection of any client and before his first join of any channel. This way, when a Clone Attack occurs, the tool will refuse any clone connection as early as possible and keep the network working. This tool need to implement in some way the three following features :

Detect the attack. Clone attacks occur a few time but are too brief and can not be detected by IRC-operators. So, a tool who want to defeat such attacks need to detect them before their effects occur.

Distinguish clones by their nicknames. Once the attack is detected, we need to refuse clones but accept legitimate users. One could simply refuse all connections during the attack, however, this would prevent anybody from joining the network creating another kind of DOS [9]. As mentioned before, the only way to recognize users is their nickname. So, our tool needs to distinguish legitimate users by reading their nickname in order to accept them and refuse the clones.

Learn. Each IRC-network has its own characteristics. A connection which could be considered as an attack in a small network should be considered as normal traffic in a bigger one that may have a lot of connection per hour. Because of different cultural habits between networks, nicknames differ between networks [2]. A nickname which sound like clones in one network could be one of a legitimate user in an other.

3 Usual protections

We have found no publication nor tools concerning the defense against clone attacks despite one feature of IRC networks and two handcrafted methods. These methods have been seen in some IRC networks but have not been released.

3.1 Passwords to enter the network

Access to IRC networks can be protected by a password. This way, when a user want to connect to a server, he must give the good password. Such networks are called *private networks*. While this technique is effective to protect the network against unguests users, it obviously does not prevent it against clones mastered by a user of this network knowing the pass.

3.2 Blacklist

This method distinguish clones from their IP addresses. The main idea is to keep a black-list of IP Address. These IP addresses are manually added to the list after an attack and send to other IRC-operator so they could use the list for other IRC networks.

Obviously, this is not an effective method to defeat a Clone Attack. As already mentioned, many users can use a single IP Address. If this address have been added to the black-list, legitimate user will be refused by the system. Furthermore, new bots, whose address is not already in the list, will be accepted.

3.3 Use simple regexp

Since clone nicknames are automatically generated, the second method use simple regexp¹ to distinguish clones. The regexp is made by IRC-operators who find some characteristics in the nicknames of the clones after an attack. For example, any nickname ending with two digits can be considered as a clone nickname.

Again, this method is not effective. A simple regexp is too restrictive and lots of legitimate nicknames are considered as clones' nicknames. In addition, when the regexp is known, it is very easy to create nicknames that match the regexp.

¹ a regexp: regular expression [21]

4 Qui-Gon

Qui-Gon is our tool against clone attacks. It detects attacks, distinguishes clones and learns to fit better his network.

Its architecture is based on two oracles. The first one (the temporal oracle) detects attacks and when asked, answers whatever or not an attack occur. The second oracle (the distinguishing oracle) distinguishes clones and users. This way, we can improve each oracle individually and if another method to detect attacks or distinguish clones is found, it can be incorporate simply by replacing the oracle.

Both oracle are working together to improve the system. In fact, when the temporal oracle detect a false attack (to much users connecting themselves), the distinguishing oracle let them connect because it recognize them. Furthermore, the only negative false the distinguishing oracle can make are while a detected attack and won't be embarrassing.

4.1 The temporal oracle

To detect an attack, the oracle measures the connection rate. As already stated, an attack consist of a lot of connections during a short time. Thus, during an attack, the connection rate increases significantly and then can be detected.

The oracle considers an attack occurs when the average rate exceed a learned value. We can simply measure the time between two connections. But if two user connect themselves at the same time, the oracle see an attack. Thus, we take the time taken by a fixed number of connections as main measure. The learn phase take place in peace period; the oracle observed the maximum value and take it as reference.

Since the connection rate is very fast, this method is effective. Using a large number of connections, we are able to smooth the impact of few simultaneous legitimate connections. Although, with a large number of connection, we will not refuse the first clones but all the next ones. Despite the few clones accepted, the others will not crash down the server and the attack will be defeat.

4.2 The distinguishing oracle

As stated before, users from an IRC network use nickname which follows some rules[2, 3]. For cultural or habits reasons, users do not choose their nicknames as random but throughout some mental process[2].

Furthermore, clone nicknames are automatically generated and follow some computational rules. Usually, they are choose randomly over the set of all possible nicknames made by letters and digits [15].

The main idea is to use probabilistic automata to distinguish clones and users.

Probabilistic automata used to distinguish clones Distinguishing clones could be done with Markov chains. Markov chains are well suited for classification problems such as distinguishing set of first names [17].

Instead of using directly Markov chain, we conceptualize our sets of clone nicknames and legitimate users with Probabilistic automata [18, 20]. There is no formal definition of probabilistic automata and we will use the one from Henin [18]. A probabilistic automata is an automata with probabilities on transitions and final states. To get the probability of the nickname, one simply read it throughout the automata's transitions. The probability of the nickname is the product of the probabilities on these transitions and the probability of the ending state to be final. This way, languages conceptualized by Markov chains are included into those conceptualized by our automata [18].

The distinction is made by comparing the probability given by two automata. The first automaton (called *the bad* one) identifies clones' nicknames while the second (called *the good* one) identifies legitimate users' nicknames. If the probability given by the good automaton is bigger than the one given by the bad automaton, the nickname is considered as a legitimate user's nickname.

The good automaton learn during peace period of time. Because a clone attack is always fast, during that phase, we consider all users are legitimate ones. The automaton uses then all these nicknames as forming the learning set.

The bad automaton learn only on demand. Because of a lot of users during the clone attack are not all clone, we can't learn automatically. After an attack, the IRC-operators must sorts nicknames from the attack. The set of nicknames considered as clones is then given to the bad automaton. These set of clones names can be sent to other IRC-operators of other networks to prevent them for such attack.

5 Tests and validation of the distinguishing oracle

We have tested the distinguishing oracle with some sets of nicknames. We have used a set of 259 nicknames found in [2] (set A_1), another set of 107 nicknames found on the IRC network *geekirc*² (set A_2). These are real nicknames of human users. We have also used clones nicknames: a set of 70 nicknames generated by a real tool performing clones attacks[15] (set B_1) and another set of 20 nicknames from a real attack we have saw on *geekirc* (set B_2).

The set B_2 from the real attack does not follow the same pattern as the set generated by the tool (set B_1). The tool generates nickname by choosing 9 random letters. The nicknames from the real attack are also chosen randomly and are formed with 5 letters and 1 to 3 digits append to the end. Both pattern are use to perform real clone attacks and are not just artificial patterns.

Here, we show two scenarios of tests. The first one tests the oracle against existing attacks and tools. Next, we present a more clever attack.

² The authors were IRC-operators of this IRC network.

5.1 Test against existing attacks

In this test, the oracle have learned a subset of 200 nicknames of A_1 as legitimate ones and a subset of 50 nicknames of B_1 . Theses sets represent a set of users the oracle have already seen on the networks and some nicknames of bots it has also seen.

Then, we have tested the oracle on all the sets of nicknames. In this test, we consider that some new human user want to connect to our networks while an attack. This attack consist of clones generated by some real tools (sets B_1 and B_2). Results are shown on figure 1.

Fig. 1. Classification of nicknames into legitimate users' nicknames. After learning 200 legitimate nicknames and 50 generated by a real clone attack tool each set has been tested. For each set, we give the amount of nicknames considered as legitimate users and the amount of errors the oracle has done.

Population	considered as legitimate
200 Legitimate learned nicknames (subset of A_1)	98.5 %
59 Legitimate tested nicknames (subset of A_1)	57.6 %
107 Legitimate tested nicknames (set A_2)	36.4 %
50 Learned Clones from pru.c (subset of B_1)	0.0 %
20 Tested Clones from pru.c (subset of B_1)	0.0 %
20 Tested Clones from a real attack (set B_2)	0.0 %

Firstly, we notice that the oracle made no negative false. As we can see (FIG. 1), all the clones are well recognize. The oracle considered as clones all those that have been learned and all those generated by the same pattern. It show that the learning process is effective. Furthermore, the 20 clones from geekirc were also well recognized showing us that the oracle can recognize unknown clones made by other tools.

Next, There's little positive false in the learning set. There was only 3 nicknames badly recognized. So, during an attack, the oracle will let connect almost all legitimate users it has already seen.

Finally, new legitimate users could sometime connect while an attack. As we can see, more than a half of the nicknames from the same network as the learning set (A_1) are well recognized. One out of three nicknames from another network (A_2) are well recognized. Thus, the oracle let connect some legitimate new users and refuse some others. It's not a bad point since the distinction is made only during an attack. The QoS³ will be maintain. Finally, as we can infer [2, 3], users from the same background are better recognized than those from another background.

³ Quality of Service

5.2 Test against a more clever attack

Obviously, one can also pick some nicknames from others networks, websites, dictionaries, ... In this test, we consider the set A_1 of nicknames from geekirc as the picked ones and use them as nicknames of clones. As the previous test show, the first time this attack occurs, only 36 percent of clones will succeed.

But once the first attack have finished, QuiGon learn the set of clones and things may change. Thus, in this test, the oracle have learn the set A_1 as legitimate users and a subset of 50 nicknames of A_2 as clones. Then, we have test the oracle on the set A_1 and A_2 . Results are shown in figure 2

Fig. 2. Behavior against more clever attack. After learning A_1 as legitimate nicknames and 50 nicknames from A_2 as clone ones each set has been tested. For each set, we give the amount of nicknames considered as legitimate users and the amount of errors the oracle has done.

Population	considered as legitimate
259 Legitimate learned nicknames (A_1)	91.1 %
50 Learned Clones from A_2	4.0 %
57 Tested Clones from A_2	42.0 %

First, we can see that the oracle still make few positive false. Only around 10 percent of legitimate users have been considered as clones. But since this occur only while an attack, these legitimate users will be able to connect themselves later, which is not so embarrassing.

Next, we can see that this time, the oracle make some negative false but learn efficiently. He have considered as legitimate 4 percent of clones it has already seen and 42 percent of clones it never saw. Once QuiGon have learn a set of nicknames of clones, he will recognize them as clones later if they come back.

6 Conclusion

By using Probabilistic automata, the tool can learn the nicknames patterns and adapt itself to other networks. The probabilistic automata are well suited to classified nicknames [20] and produce no negative false. Despite some positive false, the oracle recognize almost all legitimate user it's already seen.

It use two distinct oracles, both could be replace and allow to simply improve the tool. We are looking to improvements. Obviously, the network activity is not constant and we're trying to conceptualize such characteristic into the temporal oracle. Furthermore, the distinguishing oracle know a fixed automata. Despite the probability on the transitions and stats, the structure can't evolve. We're trying to adapt state melting [18] and state splitting [19] to those automata in order to characterize better the nicknames sets.

Thus, QuiGon is the first and only tool dedicated to prevent clone attacks. Despite some handcrafted method which are not so efficient there is no such tools. Our method can detect attack and distinguish clones. This way, during an attack, the clone are refused, defeating the attack.

References

1. J. Oikarinen, D. Reed: Internet Relay Chat Protocol, IETF, RFC n1459
2. Haya Bechar-Israeli: From Bonehead to cLoNehEAd: Nicknames, Play and Identity on Internet Relay Chat, *Journal of Computer-Mediated Communication*, vol 1, num 2
3. Elizabeth M. Reid: Electropolis : Communication and community on internet relay chat
4. Daniel Robert Karrels: Internet relay chat service framework : GNUWorld
5. Neil W. Van Dyke, Henry Lieberman, Pattie Maes: Butterfly: a conversation-finding agent for Internet relay chat, 4th international conference on Intelligent user interfaces
6. J. Mirkovic, P. Reiher: A Taxonomy of DDoS Attacks and Defense Mechanisms, *ACM SIGCOMM Computer Communications Review*, vol. 34, numl. 2
7. Christos Douligeris, Aikaterini Mitrokotsa: DDoS attacks and defense mechanisms: classification and state-of-the-art, *Computer Networks*, vol 44, num 5
8. Hemavathy Alaganandam, Pravin Mittal, Avichal Singh, Chris Fleizach: Cyber-criminal Activity
9. D. Bernstein: Syn cookies - <http://cr.yip.to/syncookies.html>
10. Bill McCarty: Botnets: Big and Bigger, *IEEE Security and Privacy*, vol. 1, num. 4
11. John Canavan: The evolution of malicious IRC Bots, *Virus Bulletin Conference*, 2005
12. Paul Barford and Vinod Yegneswaran: Malware Detection - An Inside look at botnet, *Advances in Information Security*, vol. 27
13. David Geer: Malicious Bots Threaten Network Security, *IEEE Computer Society*, vol. 38, num. 1
14. Dr. Richard, S. Wallace: The Elements of AIML Style, A.L.I.C.E. Artificial Intelligence Foundation, Inc.
15. <http://packetstormsecurity.org/DoS/pru.c>, Example of clone attack tool
16. O. Capp, E. Moulines, T. Rydn: Inference in Hidden Markov Models
17. Pierre Dupont: Noisy Sequence Classification with smoothed Markov Chains, 8me confrence francophone sur l'apprentissage automatique
18. Thibaut Henin: Reprsentation par jeu du chaos de squences d'ADN, ENS Cachan, University Rennes 1
19. Jérôme Callut, Pierre Dupont: Inducing hidden Markov Models to Model Long-Term Dependencies, *Artificial Intelligence*, European Conference on Machine Learning
20. F. Thollard, A. Clark: Apprentissage d'automates probabilistes dterministes, *Confrence d'Apprentissage*, 2002
21. Ville Laurikari: Efficient submatch addressing for regular expressions, Helsinki University of Technology, 2001