www.ijaret.org Vol. 1, Issue VIII, Sep. 2013
ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

Trace and Blocking Of Pollution in Anonymzing Networks Using Entropy Variations

D. Kalyana Chakravarthy[1]

HOD, Department Of Computer Science & Engineering,
Sri Venkateswara Engineering College,
Piplikhera, Sonepat, Haryana, Pin-131039
kc7610@gmail.com

Abstract: *Misbehaving User attacks are a critical threat to the Internet. It is important to protect the resource and trace from the Misbehaving User attack, but it is difficult to distinguish normal traffic and Misbehaving User attack traffic because the attackers generally hide their identities/origins.* ***.*** *And these attackers use incorrect or spoofed IP address. These attacks Users to access Internet services privately by using a series of routers to hide the client's IP address from the server. So it is difficult to trace this type of IPs. The web site administrators routinely rely on IP-address blocking for disabling access to misbehaving users, but blocking IP addresses is not practical if the abuser routes through an anonymizing network. The administrators block all known exit nodes of anonymizing networks, denying anonymous access to misbehaving and behaving users alike. To address this problem, we propose a trace and blocking method for Misbehaving User attacks and we present Nymble, a system in which servers can "blacklist" misbehaving users. The main feature of our proposal there by trace and blocking of users their anonymity that is based on entropy variations between normal and Misbehaving User attack traffic.*

Index Terms: *Anonymous blacklisting, privacy, revocation, IP traceback, entropy variation.*

1. INTRODUCTION

Misbehaving User attacks in the Internet. In Misbehaving User attacks, attackers generate a huge amount of requests to victims through compromised computers (zombies), with the aim of denying normal service or degrading of the quality of services. It has been a major threat to the Internet since year 2000, and a recent survey [1] on the largest 70 Internet operators in the world demon-started that Misbehaving User attacks are increasing dramatically, and individual Attacks are more strong and sophisticated. Furthermore, the survey also found that the peak of 40 gigabit Misbehaving User attacks nearly doubled in 2008 compared with the previous year.

Web, such as the dynamic, stateless, and anonymous nature of the Internet [10-11]. IP trace back means the capability of identifying the actual source of any packet sent across the Internet. Because of the vulnerability of the original design of the Internet, we may not be able to find the actual hackers at present. In fact, IP trace back schemes are considered successful if they can identify the zombies from which the Misbehaving User attack packets entered the Internet. Research on Misbehaving User detection [3], [4], mitigation [5] and filtering [6] has been conducted pervasively. However, the efforts on IP trace back are limited. A number of IP trace back approaches have been suggested to identify Misbehaving attackers and there are two major methods for IP trace back, the probabilistic packet marking (PPM) [15], [16], and the deterministic packet marking (DPM) [18]. Both of these strategies require routers to inject marks into individual packets. Unfortunately, some users have misused such networks under the cover of anonymity; users have repeatedly defaced popular Web sites such as Wikipedia. Since Web site administrators cannot blacklist individual malicious users' IP addresses, they blacklist the entire anonymizing network [2]. Such measures eliminate malicious activity through anonymizing networks at the cost of denying anonymous access to behaving users. The anonymity provided by the anonymizing network. Anonymous credential systems employ group signatures. Basic group signatures allow servers to revoke a misbehaving user's anonymity by complain into a group manager. Servers must query the group manager for every authentication, and thus, lacks scalability.

Traceable signatures allow the group manager to release a trapdoor that allows all signatures generated by a particular user to be traced; such an approach does not provide the backward unlink ability [9] that we desire, where a user's accesses before the complaint remain anonymous. Backward unlink-ability allows for what we call subjective blacklisting, where servers can blacklist users for whatever reason since the Privacy of the blacklisted user is not at risk. In contrast, approaches without backward unlink ability need to pay careful attention to when and why a user must have all their connections linked, and users must worry about whether their behaviors will be judged fairly[2].

1.1 A Sample Network with Misbehaving User Attacks.

An Anatomizing Network with Misbehaving User attacks In order to clearly describe our trace back mechanism; we use Fig 1 as a sample network with attacks to demonstrate our trace back strategy. In a Misbehaving User attack scenario, as shown in below Fig. 1.

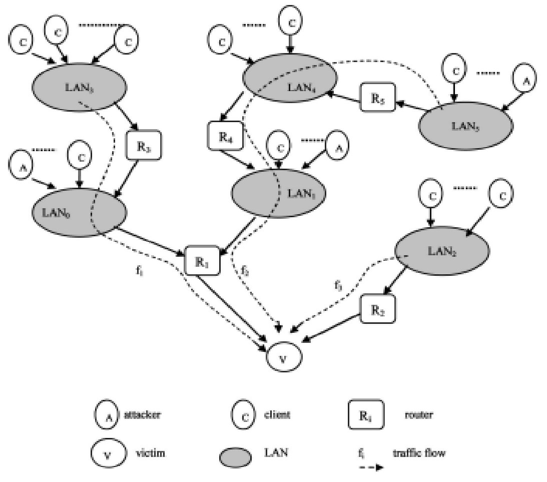

Figure 1: A Simple Network with Misbehaving User attack

The flows with destination as the victim include legitimate flows, such as f_3, and a combination of attack flows and legitimate flows, such as f_1 and f_2. Compared with no attack cases, the volumes of some flows increase significantly in a very short time period in Misbehaving User attack cases. Observers at routers R_1, R_4, R_5, and V will notice the dramatic changes; however, the routers who are not in the attack paths, such as R_2 and R_3, will not be able to sense the variations. Therefore, once the victim realizes an ongoing attack, it can push back to the LANs, which caused the changes based on the information of flow entropy variations, and therefore, we can identify the locations of attackers [1]. The trace back can be done in a parallel and distributed fashion in our proposed scheme. In Fig 1, based on its knowledge of entropy variations, the victim knows that attackers are somewhere behind router R_1, and no attackers are behind router R_2. Then the trace back request is delivered to router R1. Similar to the victim, router R_1 knows that there are two groups of attackers, one group is behind the link to LAN_0 and another group is behind the link to LAN_1. Then the traceback requests are further delivered to the edge routers of LAN_0 and LAN_1, respectively. Based on entropy variation information of router R_3, the edge router of LAN_0 can infer that the attackers are located in the local area network, LAN_0. Similarly, the edge router of LAN_1 finds that there are attackers in LAN_1, furthermore, there are attackers behind router R_4. The traceback request is then further passed to the upstream routers, until we locate the attackers in LAN_5.

1.2 System Modeling

In this paper, we categorize the packets that are passing through a router into flows. A flow is defined by a pair—the upstream router where the packet came from and the destination address of the packet. Entropy is an information-theoretic concept, which is a measure of randomness. We employ entropy variation in this paper to measure changes of randomness of flows at a router for a given time interval. We notice that entropy variation is only one of the possible metrics. Chen and Hwang used a statistical feature, change-point of flows, to identify the abnormality of Misbehaving User attacks however, attackers could cheat this feature by increasing attack strength slowly. We can also employ other statistic metrics to measure the randomness, such as standard variation or high-order moments of flows. We choose entropy variation rather than others in this paper because of the low computing workload for entropy variations. First, let us have a close investigation on the flows of a router, as shown in Fig. 2. Generally, a router knows its local topology, e.g., its upstream routers, the local area network attached to the router, and the downstream routers. We name the router that we are investigating now as local router. In the rest of the paper, we use as the set of positive integers, and R as the set of real numbers. We denote a flow on a local router by $< u_i, d_j, t>$, i, $j\varepsilon I$, $t\varepsilon R$, where u_i is an upstream router of a local router R_i , d_j is the destination address of a group of packets that are passing through the local router Ri, and t is the current time stamp. For example, the local router R_i in Fig. 2 has two different incoming flows—the ones from the upstream routers R_j and R_k, respectively. We name this kind of flows as transit flows. Another type of incoming flows of the local router R_i is generated at the local area network, we call these local flows, and we use to represent the local flows.

We name all the incoming flows as input flows, and all the flows leaving route R_i are named as output flows. We denote u_i, $i\varepsilon I$ as the immediate

www.ijaret.org Vol. 1, Issue VIII, Sep. 2013
ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

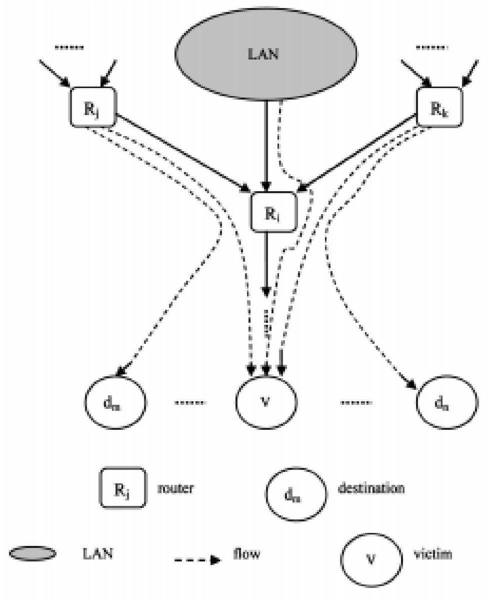

Figure 2

Traffic flows at a router on an attack path upstream routers of the local router R_i, and set U as the set of incoming flows of router R_i Therefore, U={ u_i, iεI } + {L}. We use a set D={d_i,iεI} to represent the destinations of the packets that are passing through the local router $R_{i,}$ If v is the victim router, then vεD. Therefore, a flow at a local router can be defined as follows

$F_{ij}(u_{i,}d_j)= \{<u_i,d_j,t>|u_i \, \mathcal{E} \, D,I,j, \mathcal{E} \, I\}$ (1)

We denote $|f_{ij}(u_i,d_j,t)|$ as the count number of packets of the flow f_{ij} at time t. For a given time interval ΔT, we define the variation of the number of packets for a given flow as follows:

$N_{ij,}(u_i,d_j,t+\Delta T)=|f_{ij}(u_{i,}d_j,t+\Delta T)|=|f_{ij}(u_{i,}d_j,t)|$ (2)

If we set $|f_{ij}(u_{i,}d_j,t)|=0$, then $N_{ij}(u_{i,}d_j,t+\Delta T)$ is the number of packets of flow f_{ij} which went through the local router during the time interval *ΔT*. In order to make the presentation tidy, we use $N+ij+(u_i,d_j)$ to represent $N_{ij}(u_{i,}d_j,t+\Delta T)$ in the rest of this paper Based on the larger number theorem, we have the probability of each flow at a local router as follows:

$$P_{ij}(u_i,d_j) = \frac{N_{ij}(u_i,d_j)}{\sum_{i=0}^{\infty}\sum_{j=1}^{\infty} N_{ij}(u_i,d_j)} \qquad (3)$$

Where $p_{ij}(u_{i,}d_j)$ gives the probability of the flow f_{ij} over all the flows on the local router and

$$\sum_{i=0}^{\infty}\sum_{j=1}^{\infty} P_{ij}(u_i,d_j)$$

Let F be the random variable of the number of flows during the time interval ΔT on a local router, therefore, we define the entropy of flows for the local router as follows:

$H(F) = -\sum_{i,j} P_{ij}(u_i,d_j) \, log P_{ij}(u_i,d_j)$ (4)

In order to differentiate from the original definition of entropy, we call H(F)as entropy variation in this paper, which measures the variations of randomness of flows on a given local router.

1.3 Solution

We present a secure system called Nimble, which provides all the following properties: anonymous authentication, backward unlink ability, subjective blacklisting, fast authentication speeds, rate-limited anonymous connections, revocation audit ability (where users can verify whether they have been blacklisted), and also addresses the Sybil attack [8] to make its deployment practical. In Nymble, users acquire an ordered collection of nymbles, a special type of pseudonym, to connect to Web sites. Without additional information, these nymbles are computationally hard to link, 4 and hence, using the stream of nymbles simulates anonymous access to services. Web sites, however, can blacklist users by obtaining a seed for a particular nymble, allowing them to link future nymbles from the same user—those used before the complaint remains unlinkable. Servers can therefore blacklist anonymous users without knowledge of their IP addresses while allowing behaving users to connect anonymously. Our system ensures that users are aware of their blacklist status before they present a nymble, and disconnect immediately if they are blacklisted. Although our work applies to anonymizing networks in general, we consider Tor for purposes of exposition. In fact, any number of anonymizing networks can rely on the same Nymble system, blacklisting anonymous users regardless of their anonymizing network(s) of choice Practical performance. Our protocol makes use of Inexpensive symmetric cryptographic operations to significantly outperform the alternatives. Open source implementation. With the goal of contributing a workable system, we have built an open-source implementation of Nymble, which is publicly available. Sect. 4 We provide performance statistics to show that our system is indeed practical. Some of the authors of this paper have published two anonymous authentication schemes, BLAC [12] and PEREA [13], which eliminate the need for a trusted third party for revoking users. While BLAC and PEREA provide better privacy by eliminating the TTP, Nymble provides authentication rates that are several orders of magnitude faster than BLAC and PEREA. Nymble thus represents a practical solution for blocking misbehaving users of. Anonymizing networks. We note that an extended version of this paper is available as a technical report [14].

2. AN OVERVIEW OF NYMBLE

We now present a high-level overview of the Nymble system, and defer the entire protocol description and security analysis to subsequent sections.

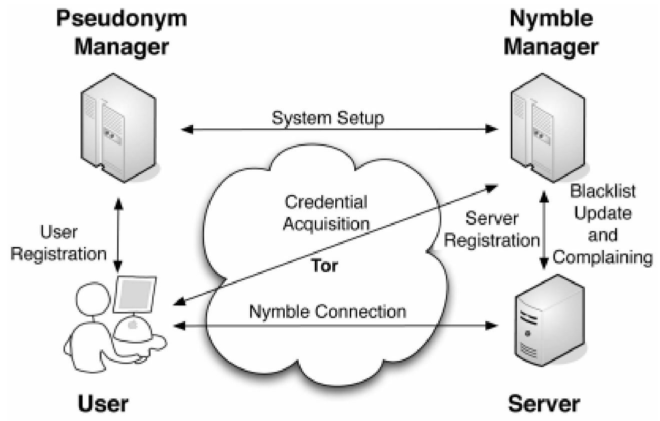

Figure 3: The Nymble system architecture.

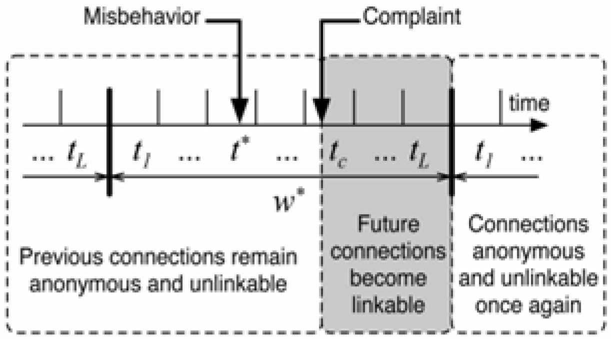
Misbehavior
Complaint
time
... t_L t_1 ... t^* ... t_c ... t_L t_1 ...
w^*
Previous connections remain anonymous and unlinkable
Future connections become linkable
Connections anonymous and unlinkable once again

Figure 2: The life cycle of a misbehaving user

2.1 Resource-Based Blocking

To limit the number of identities a user can obtain (called the Sybil attack [8]), the Nymble system binds nymbles to resources that are sufficiently difficult to obtain in great numbers. For example, we have used IP addresses as the resource in our implementation, but our scheme generalizes to other resources such as email addresses, identity certificates, and trusted hardware. And suggest other alternatives for resources. We do not claim to solve the Sybil attack. This problem is faced by any credential system [15], and we suggest some promising approaches based on resource-based blocking since we aim to create a real-world deployment.

2.2 The Pseudonym Manager

The user must first contact the Pseudonym Manager (PM) and demonstrate control over a resource, for IP-address blocking, the user must connect to the PM directly (i.e., not through a known anonymzing network), as shown in Fig. 1. We assume the PM has knowledge about Tor routers, for example, and can ensure that users are communicating with it directly Pseudonyms are deterministically chosen based on the controlled resource, ensuring that the same pseudonym is always issued for the same resource. Note that the user does not disclose what server he or she intends to connect to and the PM's duties are limited to mapping IP addresses (or other resources) to pseudonyms. As we will explain, the user contacts the PM only once per link ability window (e.g., once a day).

2.3 The Nymble Manager

After obtaining a pseudonym from the PM, the user connects to the Nymble Manager (NM) through the Anonymizing network, and requests nymbles for access to a particular server (such as Wikipedia). A user's requests to the NM are therefore pseudonymous, and nymbles are generated using the user's pseudonym and the server's identity. These nymbles are thus specific to a particular user-server pair. Nevertheless, as long as the PM and the NM do not collude, the Nymble system cannot identify which user is connecting to what server, the NM knows only the pseudonym-server pair, and the PM knows only the user identity-pseudonym pair. To provide the requisite cryptographic protection and security properties, the NM encapsulates nymbles within nymble tickets. Servers wrap seeds into linking tokens, and therefore, we will speak of linking tokens being used to link future nymble tickets. The importance of these constructs will become apparent as we proceed.

2.4 Time

Nymble tickets are bound to specific time periods. As illustrated in Fig. 2, time is divided into link ability windows of duration W, each of which is split into *L* time periods of duration T (i.e., W=L*T). We will refer to time periods and link ability windows chronologically as $t_1,t_2,\ldots,t_L$ and $w_1,w_2\ldots$, respectively. While a user's access within a time period is tied to a single nymble ticket, the use of different nymble tickets across time periods grants the user anonymity between time periods. Smaller time periods provide users with higher rates of anonymous authentication, while longer time periods allow servers to rate-limit the number of misbehaviors from a particular user before he or she is blocked. For example, could be set to five minutes, and W to one day (and thus, L=288). The link ability window allows for dynamism since resources such as IP addresses can get reassigned and it is undesirable to blacklist such resources indefinitely, and it ensures forgiveness of misbehavior after a certain period of time. We assume all entities are time synchronized (for example, with time.nist.gov via the Network Time Protocol (NTP)), and can thus calculate the current link ability window and time period.

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

2.5 Blacklisting a User

If a user misbehaves, the server may link any future connection from this user within the current link ability window (e.g., the same day). Consider Fig. 2 as an example: A user connects and misbehaves at a server during time period t within link ability window The server later detects this misbehavior and complains to the NM in time periodic ($t^* < t_c \leq t_L$) of the same link ability window w* As part of the complaint, the server presents the nimble ticket of the misbehaving user and obtains the corresponding seed from the NM. The server is then able to link future connections by the user in time periods $t_c, ,t_c+1,....,t_L$ of the same link ability window w* to the complaint. Therefore, once the server has complained about a user, that user is blacklisted for the rest of the day, for example (the link ability window). Note that the user's connections in $t_1,, t_2, ..., t^*, t+1, ..., t_c$ remain unlinkable (i.e., including those since the misbehavior and until the time of com-plaint). Even though misbehaving users can be blocked from making connections in the future, the users' past connections remain unlinkable, thus providing backward unlink ability and subjective blacklisting.

2.6 Notifying the User of Blacklist Status

Users who make use of anonymizing networks expect their connections to be anonymous. If a server obtains a seed for that user, however, it can link that user's subsequent connections. It is of utmost importance then that users be notified of their blacklist status before they present a nimble ticket to a server. In our system, the user can download the server's blacklist and verify her status. If blacklisted, the user disconnects immediately. Since the blacklist is cryptographically signed by the NM, the authenticity of the blacklist is easily verified if the blacklist was updated in the current time period (only one update to the blacklist per time period is allowed). If the blacklist has not been updated in the current time period, the NM provides servers with "daisies" every time period so that users can verify the freshness of the blacklist ("blacklist from time period t_{old} is fresh as of time period t_{now}"). As A user is guaranteed that he or she will not be linked if the user verifies the integrity and freshness of the blacklist before sending his or her nymble ticket.

3. NYMBLE CONSTRUCTION

3.1 System Setup

During setup, the NM and the PM interact as follows:

1. The NM executes *NMInitState()* and initializes its state nm State to the algorithm's output.
2. The NM extracts $macKey_{NP}$ from *nmState* and sends it to the PM over a type-Authchannel. $macKey_{NP}$ is a shared secret between the NM and the PM, so that the NM can verify the authenticity of pseudonyms issued by the PM.
3. The PM generates $nymKey_P$ by running Mac. *Key-Gen()* and initializes its state pmState to the pair $(nymKey_P, macKey_{NP})$.
4. The NM publishes $verKey_N$ in *nmState* in a way that the users in Nymble can obtain it and verify its integrity at any time (e.g., during registration)

Algorithm 1. NMInitstate()

```
nmState Ɛ S_N
1. macKey_NP ≔ Mac.KeyGen()
2. macKey_N ≔ Mac.KayGen()
3. seedKey_N ≔ Mac.KeyGen()
4. (encKey_N,dccKey_N) ≔ Enc.KeyGen()
5. (signKey_N,verKey_N) ≔ Sig.KeyGen()
6. Keys ≔ (macKey_NP,macKey N,seedKey_N,encKey_N,decKey_N,signKey N,verKey_N)
7. nmEntries ≔ Ø
8. return nmState ≔ (keys,nmEntries)
```

3.2 Server Registration

To participate in the Nymble system, a server with identity Sid initiates a type-Authchannel to the NM, and registers with the NM according to the Server Registration protocol below. Each server may register at most once in any link ability window.

a) The NM makes sure that the server has not already registered: If *(Sid,..,)ε*nmEntries in its nmState, it terminates with failure, it proceeds otherwise.

b) The NM reads the current time period and link ability window as t_{now} and w_{now}, respectively, and then obtains a surState by running (see Algorithm 2).

$NMRegisterServer_{nmState}(sid,t_{now},w_{now})$.

c) The NM appendssvrStateto itsnmState, sends it to the Server, and terminates with success.

d) The server, on receivingsvrState, records it as its state, and terminates with success

Algorithm 2. NMRegisterServer

```
Input:(sid,t,w) Ɛ H X N^2
Persistent Stete: nmStare Ɛ S_N
Output: serState Ɛ S_S
1.    (keys,nmEntries) ≔ nmState
2.    macKey_NS ≔ Mac.KeyGen()
3.    daisy_L Ɛ_R H
4.    nmEntries' ≔ nmEntries||(sid,macKey_NS,
daisy_L,t)
5.    nmState ≔ (keys,nmEntries')
6.    targer ≔ h^(L-t+1)(daisy_L)
7.    blist ≔ Ø
8.    cert ≔ NMSignBL_nmState(sid,t,w,targer,blist)
9.    svrState ≔ (sid,macKey_NS,blist,cert,Ø,Ø,Ø,t)
10.   return svrState
```

InsvrState, $macKey_{NS}$ is a key shared between the NM and the server for verifying the authenticity of nimble tickets, $time_{lastUpd}$ indicates the time period when the blacklist was last updated, which is initialized tot_{now}, the current time period at registration.

3.3 User Registration

A user with identity uid must registers with the PM once in each link ability window. To do so, the user initiates a type-Basic channel to the PM, followed by the User Registration protocol described below.

- The PM checks if the user is allowed to register. In our current implementation, the PM infers the registering user's IP address from the communication channel, and makes sure that the IP address does not belong to a known Tor exit node. If this is not the case, the PM terminates with failure.
- Otherwise, the PM reads the current link ability window as w_{now}, and runs

$Pnym:=PMCreatePseudonym_{pmState}(uid,w_{now})$

The PM then gives pnym to the user, and terminates with success.

- The user, on receiving pnym, sets her state *usrState* to (pnym, ϕ), and terminates with success.

3.4 Credential Acquisition

To establish a Nymble connection to a server, a user must provide a valid ticket, which is acquired as part of a credential from the NM. To acquire a credential for server Sid during the current link ability window, a registered user initiates a type Anonchannel to the NM, followed by the Credential Acquisition protocol below.

1. The user extracts pnym from usrState and sends the pair(pnym, Sid) to the NM.

2. The NM reads the current link ability window a w_{now}. It makes sure the user's pnymis valid: If

$NMVerifyPseunym_{nmState}(pnym,w_{now})$

Returns false, the NM terminates with failure, it proceeds otherwise.

3. The NM runs

$NMCreateCredential_{nmState}(pnym,sid,w_{now})$

Which returns a credentialcred? The NM sends cred to the user and terminates with success.

4. The user, on receiving cred, creates usrEntry:=(Sid, cred, false), appends it to its state usrState, and terminates with success.

3.5 Nymble Connection Establishment

To establish a connection to a serversid, the user initiates a type-Anonchannel to the server, followed by the Nymble connection establishment protocol described below.

3.5.1 Blacklist Validation

1. The server sends (blist, certi) to the user, where blist is its blacklist for the current time period and certis the certificate on blist.

2. For freshness and integrity, the user checks if verifies the blist is true

3.5.2 Privacy Check

Since multiple connection establishment attempts by a user to the same server within the same time period can be linkable, the user keeps track of whether she has already disclosed a ticket to the server in the current time period by maintaining a boolean variable ticket Disclosed for the server in her state. Furthermore, since a user who has been blacklisted by a server can have her connection establishment attempts linked to her past establishment, the user must make sure that she has not been blacklisted thus far. Consequently, if ticket DisclosedinusrEntries[Sid] in the user's usrState is true, or

$UserCheckIfBlacklisted_{usrState}(Sid, blist)=true$

then it is unsafe for the user to proceed and the user setssafe to false and terminates the protocol with failure.

3.5.3 Ticket Examination

1. The user setsticketDisclosed in *usrEntries[Sid]* in usrStatetotrue. She then sends(ticket) to the server, where ticket is ticket incred in *usrEntries[Sid]* in *usrState*. Note that the user discloses ticket for time period $t_{now}^{(u)}$ after verifying blist's freshness for $t_{now}^{(u)}$ This procedure avoids the situation in which the user verifies the current blacklist just before a time period ends, and then presents a newer ticket for the next time period.

2. On receiving ticket , the server reads the current time period and link ability window as $t^{(s)}_{now}$ now and $w^{(s)}_{now}$ respectively If any of the checks server link ticket is false, the server sends *(goodbye)* to the user and terminates with failure. Otherwise, it add sticket to slistin its state, sends hokayito the user, and terminates with success.

3.6 Blacklist Update

Servers update their blacklists for the current time period for two purposes. First, as mentioned earlier, the serve needs to provide the user with its blacklist (and blacklist certificate) for the current time period during a Nymble connection establishment. Second, the server needs to be able to blacklist the misbehaving users by processing the newly filed complaints. The procedure for updating blacklists differs depending on whether complaints are involved. When there is no complaint blacklists stay unchanged, the certificates need only"light refreshment." When there are complaints, on the other hand, new entries are added to the blacklists and certificates need to be regenerated. Since these updates are certified for integrity and freshness at the granularity of time periods, multiple updates within a single time period are disallowed (otherwise, servers could send users stale blacklists). Our current implementation employs "lazy" update: the server updates its blacklist upon its first Nymble connection establishment request in a time period.

4. PERFORMANCE EVALUATIONS

In this section, we evaluate the effectiveness and efficiency of the proposed entropy variation based on IP trace back mechanism. Our first task is to show that the flow entropy variation is stable for non-attack cases, and find out the fluctuations for normal situations, the second task is to demonstrate the relationship between the drop of flow entropy variation and the increase of attack strength, so that we can identify the threshold for identifying attack sources, we further simulate the whole attack tree for trace back, and evaluate the total trace back time. As mentioned in above Section, the network security community lacks suitable data sets of real large-scale Misbehaving User attacks, and it is even harder to find suitable data sets for our algorithms. Consequently, in order to evaluate our scheme, we have carefully conducted extensive simulations and real case observations. The simulation settings are arranged according to Fig. 1. We set the attack tree as a binary tree or three-branch tree, respectively, and zombies are distributed in the attack tree uniformly. We note that, our entropy variation trace back mechanism is independent from the topology of attack network and it is also independent from the network topology of victims. We use the essential Misbehaving User attack parameters as presented in [7] in our simulations, such as, 5-10 minutes attack duration, 10,000 packets per second of attack flows. The performance evaluation included two parts—the first one focused on the entropy variation monitoring at a local router, and the second part was to demonstrate the effectiveness of Misbehaving User attacker trace back and the overall trace back time.

First, we observe the stability of entropy variations at a local router during non attack periods. We examine two kinds of flows, the Poisson distribution flows and the Normal distribution flows. The Poisson distribution is treated as the pattern of Internet traffic by most researchers, and the combination of Normal distributions with different parameters can be used to approximate most distributions.
The attack strength is obvious from legitimate flows, for example, at least seven times stronger. Therefore, the proposed trace back method can deal with the majority of Misbehaving User attacks,

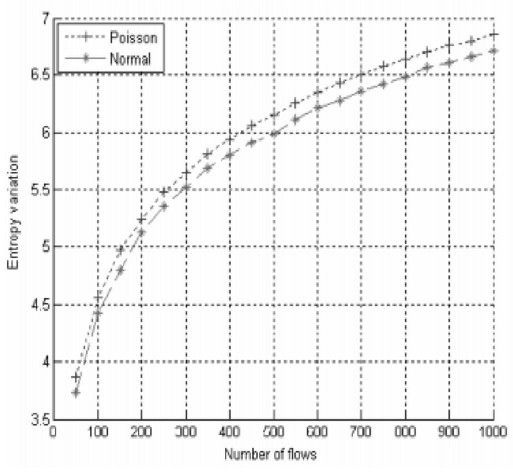

Figure 5(a): The entropy variation against number of Entropy variation against number of flows.

Based on these two convergence simulations, we can conclude that if a node in the attack tree possesses more child nodes, then the entropy variation converges faster. As a result, it is easier for us to conduct the trace back procedure. In order to estimate the overall trace back time, we assume the same number of zombies (N=1,024) and aforementioned parameters.

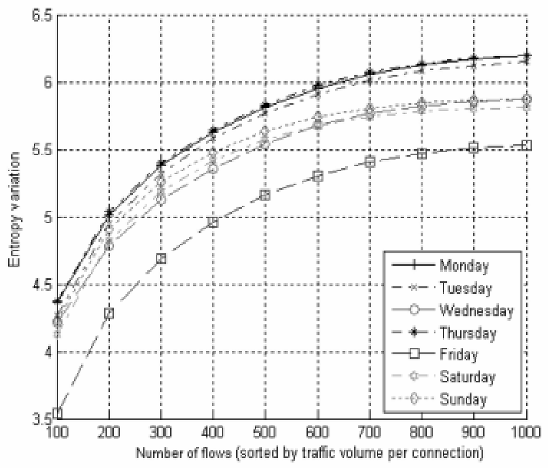

Figure5: (b) Flows for Poisson and normal distributions.

The zombies are evenly distributed in 10 groups in terms of hops away from the victim. Each of the groups can be anywhere from the victim: from 1 hop away to 31 hops away. In the worst case,

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

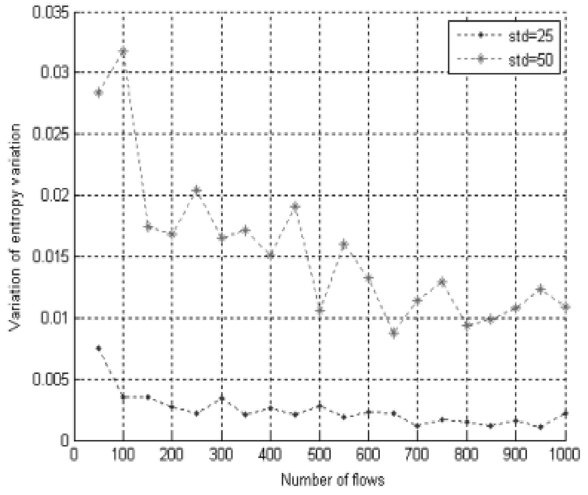

Figure 5(c): The standard variation of entropy variation against number of flows with different standard variations. The zombies are located evenly at the far end on the attack tree,

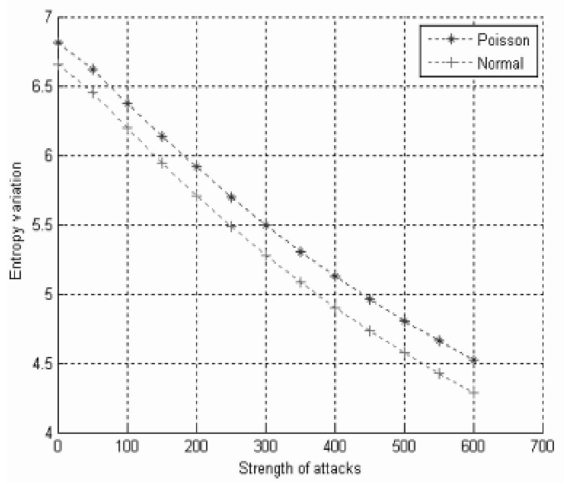

Fig. 5(d) the changes of entropy variation against strength of attacks. In other words, the 10 groups of zombies are located from 21 to 30 hops away from the victim. We simulated each case for the binary attack tree and the three-branch attack tree,

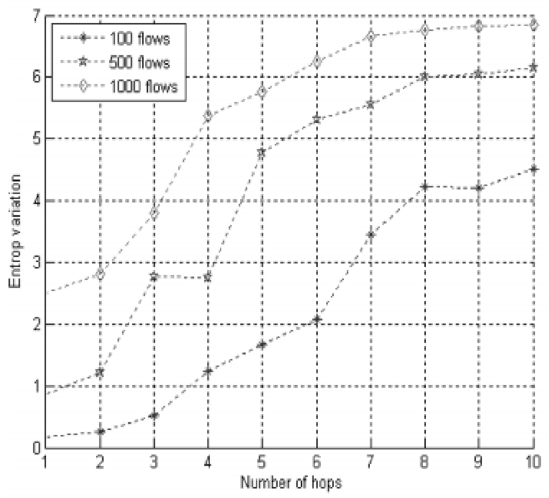

Figure 5 (e): The convergence of the entropy variation on a Binary attack tree.

This simulation demonstrates that our method is better than the previous trace back method in terms of overall trace back time. Moreover, it is shown in that the average duration for Misbehaving User attacks is 5-10 minutes, so that our method cans trace back to the most far away zombie effectively before it disappears from the attacking scene.

4.1 Experimental Results

Fig. 6 shows the size of the various data structures. The X-axis represents the number of entries in each data structure—complaints in the blacklist update request, tickets in the credential nymbles in the blacklist, tokens and seeds in the blacklist update response, and nymbles in the blacklist. For example, a link ability window of one day with five minute time periods equates to L=288 The size of a credential in this case is about 59 KB. The size of a blacklist update request with 50 complaints is roughly 11 KB, whereas the size of a blacklist update response for 50 complaints is only about 4 KB. The size of a blacklist (downloaded by users before each connection) with 500 nymbles is 17 KB.

5. CONCLUSION

We proposed an effective and efficient IP traceback scheme against misbehaving user attacks. Our nymble system can pursue the source even if an IP address is forged, and have demonstrated the effectiveness of the trace and blocking processing. The main features of our proposed method are trace and blocking method. It is to blocks the IP address of misbehaving users and identify there IP addresses based on entropy variations between normal and Misbehaving User attack traffic.

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

REFERENCES

[1]. Shui Yu,Member, IEEE, Wanlei Zhou, Senior Member, IEEE, Robin Doss,Member, IEEE, and Weijia Jia, Senior Member, IEEE "Traceback of DDoS Attacks Usin Entropy Variations" Manuscript received 27 Oct. 2008; revised 20 July 2009; accepted 21 Oct. 2009; published online 30 Apr. 2010.
[2] Patrick P. Tsang, Apu Kapadia,Member, IEEE, Cory Cornelius, and Sean W. Smith "Nymble: Blocking Misbehaving Users in Anonymizing Networks" Manuscript received 7 Dec.2008; revised 29 Apr. 2009; accepted 24 June 2009 published online 1 Sept. 2009.
[12] Y. Kim et al., "PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks," IEEE Trans. Dependable and Secure Computing,vol. 3, no. 2, pp. 141-155, Apr.-June 2006.
[3] H. Wang, C. Jin, and K.G. Shin, "Defense against Spoofed IP Traffic Using Hop-Count Filtering,"IEEE/ACM Trans. Networking, vol. 15, no. 1, pp. 40-53, Feb. 2007.
[4] Y. Chen and K. Hwang, "Collaborative Detection and Filtering of Shrew DDoS Attacks Using Spectral Analysis," J. Parallel and Distributed Computing,vol. 66, pp. 1137-1151, 2006.
[5] A. Yaar, A. Perrig, and D. Song, "StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense," IEEE J. Selected Areas Comm., vol. 24, no. 10, pp. 1853-1863, Oct. 2006.
[6] A. El-Atawy et al., "Adaptive Early Packet Filtering for Protecting Firewalls against DoS Attacks,"Proc. IEEE INFOCOM,2009.
[7] D. Moore et al., "Inferring Internet Denial-of-Service Activity," ACM Trans. Computer Systems,vol. 24, no. 2, pp. 115-139, May 2006.
[8] J.R. Douceur, "The Sybil Attack," Proc. Int'l Workshop on Peer-toPeer Systems (IPTPS),Springer, pp. 251-260, 2002.
[9]. T. Nakanishi and N. Funabiki, "Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps,"Proc. Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT),Springer, pp. 533-548, 2005.
[10] C. Patrikakis, M. Masikos, and O. Zouraraki, "Distributed Denial of Service Attacks,"The Internet Protocol J.,vol. 7, no. 4, pp. 13-35,
2004.
[11] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems,"ACM Computing Surveys,vol. 39, no. 1, p. 3, 2007.
[12] P.P. Tsang, M.H. Au, A. Kapadia, and S.W. Smith, "Blacklistable Anonymous Credentials: Blocking Misbehaving Users without TTPs,"Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07),pp. 72-81, 2007.
[13] P.P. Tsang, M.H. Au, A. Kapadia, and S.W. Smith, "PEREA: Towards Practical TTP-Free Revocation in Anonymous Authentication,"Proc. ACM Conf. Computer and Comm. Security,pp. 333-344, 2008.
[14] C. Cornelius, A. Kapadia, P.P. Tsang, and S.W. Smith, "Nymble: Blocking Misbehaving Users in Anonymizing Networks," Technical Report TR2008-637, Dartmouth College, Computer Science, Dec. 2008.
[15] B.N. Levine, C. Shields, and N.B. Margolin, "A Survey of Solutions to the Sybil Attack," Technical Report 2006-052, Univ. of Massachusetts, Oct. 2006.
[16] B. Al-Duwairi and M. Govindarasu, "Novel Hybrid Schemes Employing Packet Marking and Logging for IP Traceback,"IEEE Trans. Parallel and Distributed Systems,vol. 17, no. 5, pp. 403-418, May 2006.
[17] M.T. Goodrich, "Probabilistic Packet Marking for Large-Scale IP Traceback,"IEEE/ACM Trans. Networking,vol. 16, no. 1, pp. 15-24, Feb. 2008.
[18] A. Belenky and N. Ansari, "IP Traceback with Deterministic Packet Marking,"IEEE Comm. Letters, vol. 7, no. 4, pp. 162-164, Apr. 2003.