Preventing UDP Flooding Amplification Attacks with Weak Authentication

Edoardo Biagioni
Department of Information and Computer Sciences
University of Hawai'i at Mānoa
esb@hawaii.edu

***Abstract*—An attacker wishing to flood a network with excess amounts of network traffic may send UDP packets with a spoofed IP source address corresponding to the target network. In many cases servers then amplify the attack by replying to the target network with more data than was sent by the attacker. This kind of attack has been successful in the past using both DNS and NTP servers.**

The AllNet protocol has been designed to deliver data over UDP as well as other media. Once an AllNet peer receives a suitable UDP packet, it records the sender's IP address and begins to forward AllNet data to that address. This is a legitimate form of traffic amplification, with one packet being used to request that a limited number of other packets (currently 100) be sent to this IP address.

To keep attackers from using AllNet peers for flooding amplification attacks, AllNet peers require potential contacts to return a bitstring that was sent to that specific IP address. In this way, legitimate contacts can receive and return the bitstring and start receiving their data. In contrast, attackers who spoof their IP address and do not receive the bitstring, are unable to direct amplified traffic to other networks.

This very weak form of authentication, conceptually related to TCP SYN cookies, only verifies that the packet comes from a system that is able to receive packets sent to that specific IP address. Such weak authentication is sufficient to prevent flooding amplification attacks.

***Index Terms*—Flooding attack, DoS, SYN cookies, UDP.**

I. Introduction and Related Work

In 2013 and 2014, a series of powerful Denial of Service (DoS) attacks hit target networks belonging to the firm CloudFlare. At its peak, the largest such attack was able to deliver over 400Gbps to the target network [1]. These attacks combined two major techniques:

- **IP source address spoofing**: the attackers sent packets with an IP source address that matched the target of their attack. Normally, IP source addresses record the sender of the IP packet. In these attacks the IP source addresses belonged not to the sender, but to the very target of the attack. This caused third-party servers, unrelated to either the attacker or the target, to send data to the target.
- **Traffic amplification**: the attackers sent the packets with the spoofed IP source address to servers that replied to small packets with larger amounts of data. This means a small amount of traffic from the attacker generated large amounts of traffic on the target network. Both DNS servers [2] and NTP servers [1] were used (in different attacks) to provide traffic amplification.

Because of the spoofed IP source address, the amplified data was sent directly to the target, rather than the attacker. This resulted in a denial of service to the target network.

This problem can be, and has been, solved by blocking either of the two techniques used by attackers. In regards to blocking spoofed IP source addresses, ISPs could in theory stop packets with spoofed IP source addresses from being sent outwards from their networks. This is called *egress filtering*. For a variety of reasons that include lack of incentive and the additional workload required, many ISPs do *not* perform egress filtering.

The second technique used by attackers can be stopped by designing servers and protocols that respond properly to legitimate UDP traffic but cannot be used for traffic amplification on other networks [3]. In the case of legacy NTP servers, for example, this is accomplished by disabling the `monlist` command, or by replacing the server software with up-to-date implementations of the NTP protocol [4].

We wish for AllNet peers to be unusable for flooding amplification attacks, that is, to respond properly to UDP packets from other legitimate AllNet programs but not support data amplification for spoofed source addresses.

AllNet [5] [6] is a protocol for interpersonal communication that is designed to work over a variety of networking technologies, including ad-hoc networks and UDP. Many such technologies, including ad-hoc networks, cannot be leveraged for the attacks described above, but UDP can. Instead, the AllNet protocol itself must be resistant to such attacks.

The core of our technique, which can be applied to other protocols besides AllNet, is to have each peer respond to an unauthenticated UDP packet P by sending a small packet K, a *keepalive*. K contains an unpredictable bitstring B which is computed from two inputs: a randomly-selected bitstring R known only to the sender, and the IP source address of the original packet. This keepalive K is sent back to the IP source address received as part of the original UDP packet P. If the IP source address was not spoofed, K will likely reach the sender of the original UDP packet, which can return B in a subsequent packet.

If the source IP address I of this latest packet can be combined with R to give B, then the sender is authenticated as being able to receive packets addressed to I. This weak authentication shows that the sender is reachable at the source address I, and the IP source address can now added to

the internal data structures of the AllNet peer as a suitable recipient for AllNet messages.

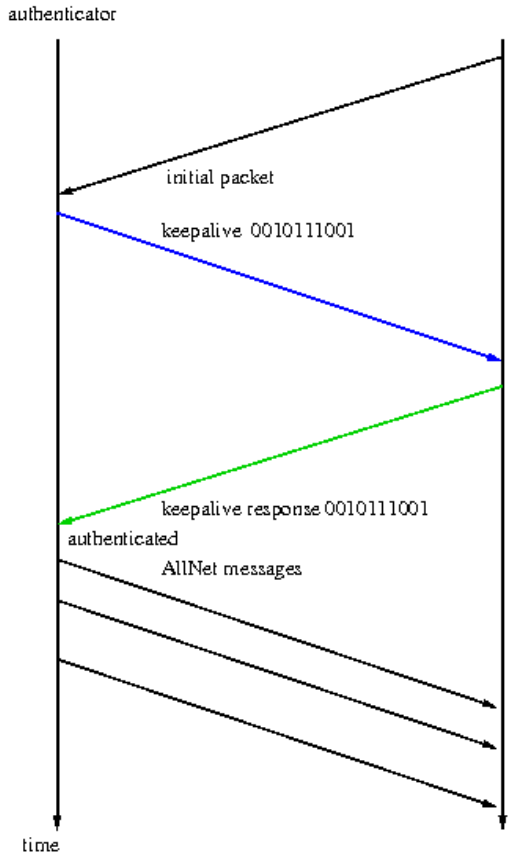

Fig. 1. The basic exchange leading to weak authentication.

Once the weak authentication has completed, the AllNet peer starts forwarding messages to the newly authenticated peer, as shown in Figure 1.

In contrast, Figure 2 shows that an initial packet with a spoofed IP source address leaves the attacker unable to complete the weak authentication process.

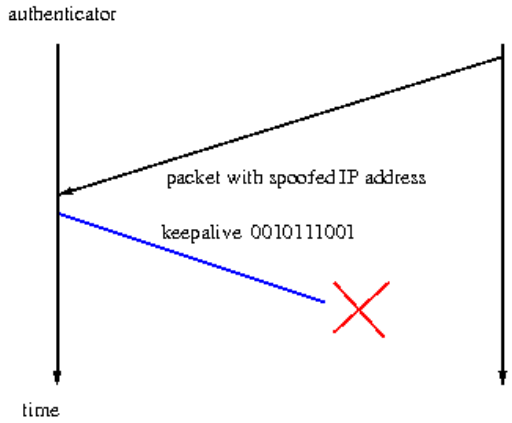

Fig. 2. The response to a packet with a spoofed IP source address is not delivered to the sender, which is then unable to issue the corresponding keepalive response.

The contributions of this paper include:

- an efficient and generalizable method for only responding to receivers that are able to confirm that they can receive packets sent to a specific IP address.
- cryptographic techniques analogous to TCP Syn Cookies to be able to properly respond to an arbitrary number of spoofed packets, while requiring only a fixed amount of state.

The remainder of this section continues discussion of related work. Section II then describes several variants of this weak authentication protocol. Section III considers the performance of the weak authentication protocol, and SectionIV provides concluding remarks.

A. SYN Cookies

SYN Cookies [7] are a technique introduced to blunt the impact of syn-flooding Denial of Service attacks on TCP-based servers. As in the flooding amplification attacks, syn-flooding attacks use spoofed IP source addresses, but here the focus of the attack is on getting the server to allocate large amounts of memory for new connections that are never completed and will never be useful. Such incomplete connections get in the way of legitimate connections to the server. With SYN Cookies, a server maintains only a fixed amount of total state for any number of incomplete connections. This is accomplished by encrypting the state of the incomplete connection in the initial sequence number (ISN) sent in response to the TCP SYN packet. In particular, the ISN carries part of a cryptographically secure hash of the IP source address of the SYN packet and some local state on the server. When the third packet in the TCP three-way exchange (the ACK) is received by the server, corresponding state is only allocated if the hash of the IP address and the local state matches the received sequence number.

As pointed out in the RFC, SYN cookies are an imperfect mechanism. This is mostly because TCP has options, only some of which can be captured with a SYN cookie. RFC 6013 [8], although now of only historical value, suggests ways to accomodate arbitrary TCP options, but at the cost of impacting other customary uses of TCP.

Similar to conventional TCP SYN Cookies, in AllNet a keepalive only needs to be a secure hash of the IP source address of a sender.

B. Security Model

We assume attackers wish to flood, i.e. send large amounts of useless data, to a target network. For the attackers, it is generally preferable if the data appears to originate from computers that are not directly under the control of the attackers.

We assume that it is undesirable for AllNet peers, and other UDP servers, to participate in such an attack.

We also assume that IP packets with an IP destination address in the target network are delivered only to that IP destination address, and specifically not delivered to the attackers. If attackers are able to snoop traffic addressed to the

target network, the authentication technique described in this paper will not be effective. Fortunately, attackers who resort to Denial of Service attack generally only do so if they are unable to directly penetrate the target network.

Without knowing whether the IP source address is spoofed, a server may confirm whether a packet is legitimate by sending a response, then waiting for a confirmation that the response was correctly received. Such a confirmation is evidence of weak authentication demonstrating that the IP source address can be used to reach a sender that is participating in the protocol, but without giving any stronger information about the sender of the packet.

II. Design

This section describes a family of protocols to keep a server or a peer from amplifying the effect of received UDP packets with spoofed IP source addresses.

A. Basic Design

The first and simplest such protocol is to respond to an incoming UDP packet from a new IP source address and port number, by sending a keepalive packet to that same address and port. The keepalive contains a cryptographic hash of the IP source address and source port number, concatenated with an internal random secret R known only to the server.

At this point, the new address is not yet (weakly) authenticated. Since the address may have been spoofed, the server does not allocate any state at this time, and specifically, records neither the source address of the incoming packet, nor having sent the keepalive.

A legitimate peer will respond with a keepalive response containing the hash it was sent. When this is received and the IP source address hashes to the same value as was received in the keepalive response, the server can trust that the sender (of the keepalive response) was able to receive the keepalive, and therefore that the IP source address was not spoofed. The server can then allocate state and begin to send data to the address.

An attacker, on the other hand, will never see the keepalive packet, and will therefore never be able to respond properly. This is sufficient to prevent this server being used for flooding amplification attacks.

Because UDP is unreliable, a program desiring to receive data from a server may have to transmit the original packet repeatedly, and may likewise have to retransmit the keepalive response. The server, on the other hand, simply responds to each incoming packet, and until the weak authentication succeeds, keeps no information about the remote peer. This means keepalives are idempotent, i.e. can be sent once or multiple times with the same effect, and servers are stateless until the peer is authenticated.

B. Variations on the Basic Design

If desired, the keepalives can be sent again after a certain length of time, to confirm that packets sent to the given IP address are still reaching a device participating in this protocol. Since, unlike TCP, UDP does not reliably detect errors, repeating the weak authentication protocol can confirm that the remote peer still wishes to receive data from this sender.

With or without such repeated authentication, the cryptographic hash may include, as well as the IP source address and the random internal secret, an internal counter that is incremented periodically. Such a counter, in combination with re-generating the random secret R on every reboot (and on every overflow of the internal counter), prevents replay attacks.

This process of computing the bitstring B is illustrated in Figure 3.

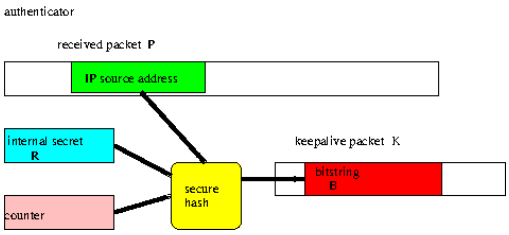

Fig. 3. Computing the secret bitstring that is sent in the keepalive packet.

III. Discussion and Implementation

The weak authentication presented in this paper is very lightweight, requiring a single hash of readily available values for every incoming packet from an unknown IP address and UDP port. Since all hashes are computed on the server, the hash algorithm need not be standardized and can be changed at will.

As far as performance is concerned, if AllNet is implemented without such weak authentication, sending a single packet to a running AllNet peer is sufficient to start the flow of AllNet data to the given IP source address. With weak authentication, the time to start the flow of Allnet data incurs additional latency of at least one round-trip time. AllNet peers connect to the Internet on a scale of seconds to years, and AllNet is fundamentally designed for interpersonal communications, so one additional round-trip time for weak authentication does not noticeably affect performance.

It is interesting to consider the consequences of repeated authentication. In particular, suppose we limit the number of data packets we send to each destination to a fixed number, say 10. Then, each receiver should send a keepalive response to the sender at least once every 10 data packets, and since UDP packets are not delivered reliably, keepalives should be sent more often. If the keepalive counter is incremented for every 10 data packets sent, then the sender also has to send new keepalives before 10 data packets are sent. In cases where the traffic may be high, such as a new AllNet peer joining the network, the round-trip latency might be more than the time between updates of the keepalive counter. At this point, both the sender and the receiver might have several keepalives and keepalive responses in transit at the same time, as shown in Figure 4. To support large amounts of traffic

without interruption in the data flow, an AllNet peer therefore must be able to accept as correct older keepalive responses. The algorithm discussed above, using counters, is suitable for accepting delayed keepalive responses as long as we keep a copy of the old random secret R_{-1} and a record of the last counter for which we have received a response.

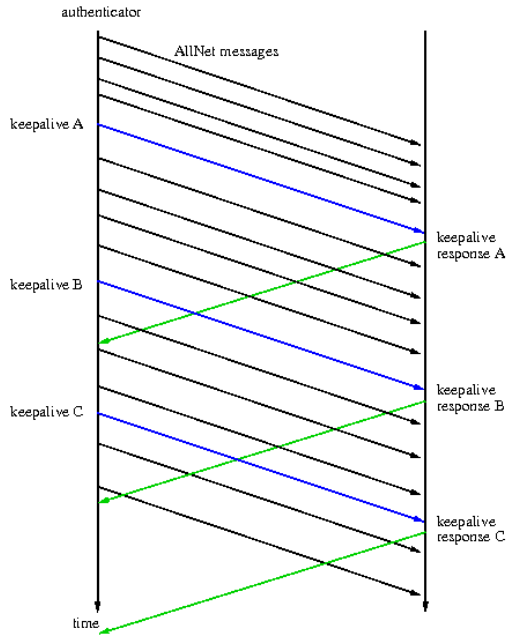

Fig. 4. Timeline showing when different packets are sent and received under conditions of high traffic load and high latency.

We have completed an initial implementation in the AllNet network of one of the protocols in Section II, specifically the protocol which requires repeated authentication (currently once every 10-100 packets – we are still evaluating the implementation under different circumstances), but does not keep a counter.

Since the generation of the bitstring B is a purely local computation, selecting a new algorithm in the future, e.g. to use a counter, can be done without changing the protocol. Such repeated authentication with different values of the bitstring B would have the advantage of limiting the amplification available to an attacker that is able to capture one or a few, but not all, of the keepalive packets.

In addition, since this protocol is very lightweight and since in AllNet communication is between symmetric peers (AllNet peers cannot be split into clients and servers), the protocol can be used in both directions simultaneously. AllNet keepalive packets therefore carry both a secret bitstring to be sent back by the receiver of the keepalive packet, and also a second secret bitstring to be used to authenticate the sender of the keepalive packet. The presentation in Section II, which distinguished keepalives from keepalive responses is useful for exposition, but in practice combining the secret bitstring from the sender in the same packet with the authenticating response to the receiver reduces the overall number of packets sent and the overhead of sending keepalives.

A. Evaluation

The preliminary implementation of this algorithm has been used to gather statistics on the performance of this weak authentication protocol under normal conditions, i.e. not under attack.

In one case, $7,238$ AllNet messages were captured in 500 seconds, for a rate of about 14.5 messages per second. Of these messages, 163 or 2.3% were authenticating keepalives with both sender and receiver authentication. Of these, 124 were sent by the host under test (100 on IPv4 and 24 on IPv6) and 39 were received (9 on IPv4 and 30 on IPv6).

Because this is a preliminary implementation, it is likely these numbers will change as further analysis drives improvements in implementation. Once the software is fully functional, we expect to test simulated attacks and verify that the weak authentication prevents amplification of UDP flooding attacks from spoofed IP source addresses.

B. Future Work: Denial of Service Attack on the AllNet network

If an attacker wished to do a denial of service attack on the AllNet network, the attacker could send a false authentication keepalive to an AllNet host A. This fake keepalive would be sent with the spoofed IP source address of a different AllNet host B, carrying a random secret bitstring. This AllNet host A will then use this secret bitstring to attempt to authenticate itself with the AllNet host B at the spoofed source address. Since the attacker's secret bitstring will not authenticate A to B, B may blacklist A and ignore its communication.

We have not yet studied this attack in detail, but our analysis so far suggests that it would have little likelihood of success. This is because host A gets keepalive packets both from the attacker and from host B, and will respond correctly to any keepalive packet sent by B, authenticating itself with such responses. To really succeed, the attacker would have to prevent the delivery of keepalive packets from B to A, which is much harder than simply sending a few packets with spoofed source address.

IV. Conclusions

We have presented a technique for preventing AllNet peers, and similar servers that respond to incoming UDP packets, from participating in flooding amplification attacks.

The technique is inspired by TCP SYN Cookies, but is simpler because it doesn't need to deal with the complexity of the TCP connection establishment. Instead, the received IP source address and an internal secret (and perhaps a counter) are hashed to give a secret identifier which is sent to the new peer. Further data is then transmitted only after the same secret is returned – this returned secret indicates that a collaborating peer indeed wishes to receive the data transmission at the given IP address. The secret can be verified with the same hash computation as was used to generate it in the first place, so that only a fixed amount of state is required until the authentication is complete, no matter how many actual or spoofed peers may be trying to authenticate.

Because the AllNet layer does not rely on conventional network addresses in delivering data [9], packets are forwarded to a number of other AllNet peers in hopes that one of them is the final destination, or will deliver the packet closer to its final destination. This makes AllNet an ideal target for a flooding amplification attack, since a single UDP packet may request delivery of many AllNet packets to the spoofed IP source address. The weak authentication protocol described here is designed to prevent the use of AllNet peers for such flooding amplification attacks.

As a final note, the algorithm is independent of the details of the IP address, and can be used equally well with IPv4 and IPv6 packets.

ACKNOWLEDGMENTS

The author wishes to thank the many individuals who have contributed to the AllNet project over the years, and in particular the recent and ongoing work by Tiago do Couto and Henry Eck.

REFERENCES

[1] S. Khandelwal, "Largest Ever 400Gbps DDoS attack hits Europe uses NTP Amplification", The Hacker News (thehackernews.com), February 11, 2014.

[2] M. Kumar, "World's biggest DDoS attack that Almost Broke the Internet", The Hacker News (thehackernews.com), March 28, 2013.

[3] Acunetix, "Preventing NTP Reflection DDOS Attacks Based on CVE-2013-5211", acunetix.com, September 20th, 2014.

[4] "NTP: DoS in monlist feature of ntpd (CVE-2013-5211)", rapid7.com, January 2, 2014.

[5] E. Biagioni, "Ubiquitous Interpersonal Communication over Ad-Hoc Networks and the Internet, at the 47th HICSS (Hawaii International Conference on Systems Sciences), January 2014.

[6] E. Biagioni, "A Ubiquitous, Infrastructure-Free Network for Interpersonal Communications", Edoardo Biagioni, presented at the fourth International Conference on Ubiquitous and Future Networks (ICUFN 2012), July 4-6, 2012, Phuket, Thailand.

[7] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.

[8] W. Simpson, "TCP Cookie Transactions", RFC 6013, January 2011.

[9] E. Biagioni, "Mobility and Address Freedom in AllNet", Ninth International Conference on Ubiquitous and Future Networks (ICUFN 2017), July 4-7, 2017, Milan, Italy.