

[UNIX] Security Flaws Found in Tinc

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-01/0039.html>

From: support@securiteam.com

Date: 01/11/02

From: support@securiteam.com

To: list@securiteam.com

Date: Fri, 11 Jan 2002 10:55:10 +0100 (CET)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> -- Know that you're safe.

Security Flaws Found in Tinc

SUMMARY

<<http://www.nl.linux.org/pub/linux/tinc/>> Tinc is a Virtual Private Network daemon that uses tunneling and encryption to create a secure, private network between hosts on the Internet.

An analysis of Tinc revealed several cryptographic weaknesses in the VPN protection.

DETAILS

Background:

This section describes how Tinc secures forwarded packets. The outgoing packet begins with a 'salt' of 2 bytes containing a cryptographically strong random value. It plays the role of an IV according to the manual "2 bytes of salt (random data) are added in front of the actual VPN packet, so that two VPN packets with (almost) the same content do not seem to be the same for eavesdroppers." The forwarded packet is appended. The couple salt and forwarded is padded to be 64bit aligned (blowfish's block size). The whole (salt, forwarded packet and padding) is encrypted with blowfish in CBC.

Vulnerabilities:

No packet authentication:

The aim of encryption is to make the data unreadable for anybody who does

Securiteam: [UNIX] Security Flaws Found in Tinc

not know the key. It does not prevent an attacker from modifying the data. People assume that an attacker will not do it because the attacker would not be able to choose the resulting clear text. However, this section shows that the attacker can choose the resulting clear text to some extent and that modifying the cipher text data may be interesting even if the attacker ignores the result.

Inserting random data:

If the attacker modifies the cipher text without choosing the resulting clear text, it will likely produce random data. The legitimate user will not detect the modification and will use them as if they were valid. As they likely appear random, it will result in a Denial of Service (DoS).

Inserting chosen data:

The encryption mode is CBC [RFC 1752, sec 5.3]. CBC allows "cut and paste" attacks – i.e. the attacker can cut encrypted data from one part of a packet and paste them in another location. As both data sections have been encrypted by the same key, the clear text will not be completely random data.

This lack of authentication is not a CBC flaw. Authentication is not considered an aim of the encryption mode, so most modes (e.g. ECB, CFB, and OFB) do not authenticate the data. To use another mode would be flawed in the same way except if they explicitly protect against forgery. Recently some modes including authentication popped up to speed up the encryption / authentication couple but as far as we know, they are all patented.

In short, encrypting with CBC is $C_n = \text{Enc}(C_{n-1} \text{ xor } P_n)$ where $\text{Enc}(x)$ is encrypting x , P_n is the n th block of plain text and C_n the n th block of cipher text. For the first block, C_{n-1} is an Initial vector (a.k.a IV) which may be public and must be unique for a given key. The decryption is $P_n = \text{Dec}(C_n) \text{ xor } C_{n-1}$. See, "National Institute of Standards and Technology implementing and using the NBS data encryption standard. Federal information processing standards fips74, April 1981", for a longer description of CBC.

If the attacker copies s blocks from the location m to n (a.k.a $[C_n, \dots, C_{n+s-1}] == [C_m, \dots, C_{m+s-1}]$), P_{n+1} up to P_{n+s-1} will be the same as P_{m+1} to P_{m+s-1} and P_n will likely appear random. C_n (i.e. C_m) will be decrypted as $P_n = \text{Dec}(C_n) \text{ xor } C_{n-1}$ but C_{m-1} and C_{n-1} are different so P_n will likely appear random. Nevertheless $P_{n+1} = \text{Dec}(C_{n+1}) \text{ xor } C_n = \text{Dec}(C_{m+1}) \text{ xor } C_m = P_{m+1}$, so $P_{n+1} = P_{m+1}$. Therefore, if the attacker has an idea of the content of a group of blocks in a packet, he can copy them to the N th block, thus it can choose the content of it without being detected.

As usual packets are not designed to appear random its content may be predictable to some extent (e.g. IP header). The attacker may use such information to guess the contents and do a knowledgeable cut/paste.

Securiteam: [UNIX] Security Flaws Found in Tinc

Insecure IV:

The aim of an IV is to hide the repetitive patterns inside the encrypted plain text, so it must be unique for a given key. Tinc's IV, called salt in the source, is random so they are not guaranteed to be unique and are vulnerable to the birthday paradox. Moreover, the IV is only 16bit long, so as a rule of thumb if Tinc forwards 255 packets, there is a probability of 50% to have at least 2 packets with the same IV.

No replay protection:

Tinc does not include any protection against packet's replay, so an attacker who can eavesdrop the encrypted packets can successfully replay them later and the destination will consider them as legitimate.

The manual section 6.3.2 claims "There is no extra provision against replay attacks or alteration of packets. However, the VPN packets, normally UDP or TCP packets themselves, contain checksums and sequence numbers. Since those checksums and sequence numbers are encrypted, they automatically become cryptographically secure. The kernel will handle any checksum errors and duplicate packets."

We believe it is risky to base the security on assumption on the forwarded packets. Moreover, in this case, the assumptions are incorrect. UDP does not have any sequence number. TCP do have sequence numbers but, for example, an attacker can replay a TCP SYN packet to perform a SYN flood attack on a server behind the Tinc peer.

ADDITIONAL INFORMATION

The information has been provided by <<mailto:jme@off.net>> Jerome Etienne.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

-
- **Previous message:** support@securiteam.com: "[NEWS] Netscape Publishing wp-force-auth Command"
 - **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)