CS427 Project

SOCat Cryptography Analisis

CS427 Project

Winter 2016

Student:
Sam Quinn
Quinnsa@Oregonstat.edu

Professor:
Michael Rosulek
rosulekm@eecs.oregonstate.edu

March 13, 2016

Contents

1 Introduction **2**

2 Problem Defined **2**

3 Why the problem is bad **2**

3.1 Backdoor Possibilitties 2

4 How they fixed the problem **3**

5 Conclusion **3**

5.1 What I learned 3

1 Introduction

Within Unix operating systems everything is considered a file, socat is a means of outputting or concatenating output to a socket. Socat can relay data in a bidirectional manner between two independent data channels. Socat can transmit to a file, pipe, device, or as socket (IP4, IP6, UDP, TCP) all encrypted over SSL. SSL is mandatory here since anytime you send data any eavesdroppers can view your data in plaintext, unless however it is encrypted.
Socat can be used as a TCP port forwarder essentially socks proxying TCP traffic to another remote system in a secure manner. There for all of your outbound traffic will pass through the secure socket tunnel to the remote system and from there fulfill the desired request.
As you can see the importance of security and encrypted data transfers in this application. As I mentioned earlier all network data traffic is susceptible to eavesdroppers, the protection comes from the data transmitted being encrypted where the adversary would not be able to understand the data that is being transferred. The entire security of the data transmitted over the wire is based on the encryption model that the developers of the application have implemented. Oddly enough many developers do not specialize in cryptography and many so called secure applications are inherently very insecure. This was the fact for Socat as well.
Socat secures network transmission with the extremely popular OpenSSL. OpenSSL is infact still considered very secure, Socats problems were a product of their own misuse of the secure libraries within OpenSSL as defined in the following section.

2 Problem Defined

Socats implemented a hybrid encryption scheme to securely transfer data among peers. The symmetric keys are obtained using Diffie Hellman key exchange. Diffie Hellman protocol needs to publicly share two variables g and p to function correctly. The variable p is very large, at least 512 bits in length. The p value will be used as the modulus for the combined shared key after the two parties raise the g to their private keys. When generating p and g you want to choose good numbers but most importantly both these numbers MUST be prime, which is where Socat went wrong. Socat for a full year had a p value that was not prime.
Their exact value was 114356381100738840153121389513746326020580806750705217075797030883704465976720674522290245
isprime(114356381100738840153121389513746326020580806750705217075797030883704465976720674522290245668347324490
143319364394905942617148968085785991039146683740268996579566827015580969124702493833109074343879894586653465
277888932760697247965045556755976589005956167697737270632318753141568853613791001332648041847107894071285740
So this is where it gets a little fishy for me, as we can see above that the first smallest value is not prime.
Then on January 4 2015 Zhigang Wang, an Oracle Solution Specialist reported the bug and a submitted the patch. This was mentioned in the commit message made by Gerhard Rieger the maintainer and lead developer.

3 Why the problem is bad

supa bad!

3.1 Backdoor Possibilitties

Oh yeah

4 How they fixed the problem

better key

5 Conclusion

The end

5.1 What I learned

A lot