**设计思路**

如果应用程序将秘密值添加到字符串中，使用易受攻击的算法对其进行哈希处理，然后将字符串和哈希值（而不是秘密）委托给攻击者，则该应用程序很容易受到哈希长度扩展攻击。然后，服务器依靠机密来确定以后返回的数据是否与原始数据相同。

即使攻击者不知道前置密码的值，他仍然可以为生成有效的哈希值{secret || data || attacker\_controlled\_data}。这可以通过简单地选择哈希算法保留的位置来完成；事实证明，继续进行散列所需的状态的100％在大多数散列算法的输出中！我们只需将该状态加载到适当的哈希结构中，然后继续哈希即可。

当知道secret的长度、message和hash(secret|message)时，可以进行哈希扩展攻击。

**运行结果**

$ echo -n secretdata |gmssl sm3

(stdin)= 3bceb5ecbd2555b0c5214ec568fa96c766e41ad2567f0f8e8726d68cf244f008

$ ./hash\_extender --data data --secret 6 --append append --signature 3bceb5ecbd2555b0c5214ec568fa96c766e41ad2567f0f8e8726d68cf244f008 --format sm3

Type: sm3

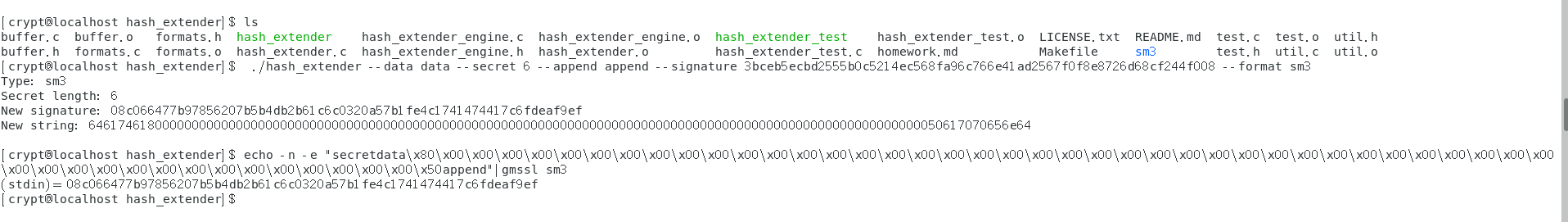
Secret length: 6

New signature: 08c066477b97856207b5b4db2b61c6c0320a57b1fe4c1741474417c6fdeaf9ef

New string: 64617461800000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000050617070656e64

$ echo -n -e "secretdata\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x50append"|gmssl sm3

(stdin)= 08c066477b97856207b5b4db2b61c6c0320a57b1fe4c1741474417c6fdeaf9ef

****