



Sign in

staaldraad / go-ntlm Public

Notifications

Fork 3

Star 4

forked from [bigkraig/go-ntlm](#)

Code

Pull requests

Actions

Projects

Security

Insights

go-ntlm / ntlm / ntlmv1.go



473 lines (397 loc) · 14.5 KB

Code

Blame

Raw



```
1 //Copyright 2013 Thomson Reuters Global Resources. BSD License please see License file for more info
2
3 package ntlm
4
5 import (
6     "bytes"
7     rc4P "crypto/rc4"
8     "errors"
9     "log"
10    "strings"
11 )
12
13 /*****
14  Shared Session Data and Methods
15  *****/
16
17 type V1Session struct {
18     SessionData
19 }
20
21 func (n *V1Session) SetUserInfo(username string, password string, domain string) {
22     n.user = username
23     n.password = password
24     n.userDomain = domain
25 }
```

```
26
27 func (n *V1Session) GetUserInfo() (string, string, string) {
28     return n.user, n.password, n.userDomain
29 }
30
31 func (n *V1Session) SetMode(mode Mode) {
32     n.mode = mode
33 }
34
35 func (n *V1Session) SetTarget(target string) {
36     n.target = target
37 }
38
39 func (n *V1Session) Version() int {
40     return 1
41 }
42
43 func (n *V1Session) SetNTHash(nthash []byte) {
44     //fmt.Printf("Passed: %x\n", nthash)
45     n.responseKeyNT = nthash
46 }
47
48 ✓ func (n *V1Session) fetchResponseKeys() (err error) {
49     //check if we have set the LM from the command line (pass the hash)
50     if len(n.responseKeyNT) > 0 {
51         return
52     }
53     n.responseKeyLM, err = lmowfv1(n.password)
54     if err != nil {
55         return err
56     }
57     n.responseKeyNT = ntowfv1(n.password)
58     return
59 }
60
61 ✓ func (n *V1Session) computeExpectedResponses() (err error) {
62
63     if NTLMSSP_NEGOTIATE_EXTENDED_SESSIONSECURITY.IsSet(n.NegotiateFlags) {
64         n.ntChallengeResponse, err = desL(n.responseKeyNT, md5(concat(n.serverChallenge, n.
65         if err != nil {
66             return err
67         }
68         n.lmChallengeResponse = concat(n.clientChallenge, make([]byte, 16))
69     } else {
70         n.ntChallengeResponse, err = desL(n.responseKeyNT, n.serverChallenge)
71         if err != nil {
```

```
72         return err
73     }
74     // NoLMResponseNTLMv1: A Boolean setting that controls using the NTLM response for
75     // response to the server challenge when NTLMv1 authentication is used.<30>
76     // <30> Section 3.1.1.1: The default value of this state variable is TRUE. Windows
77     // does not support providing NTLM instead of LM responses.
78     noLmResponseNtlmV1 := false
79     if noLmResponseNtlmV1 {
80         n.lmChallengeResponse = n.ntChallengeResponse
81     } else {
82         n.lmChallengeResponse, err = desl(n.responseKeyLM, n.serverChallenge)
83         if err != nil {
84             return err
85         }
86     }
87 }
88
89 return nil
90 }
91
92 func (n *V1Session) computeSessionBaseKey() (err error) {
93     n.sessionBaseKey = md4(n.responseKeyNT)
94     return
95 }
96
97 ✓ func (n *V1Session) computeKeyExchangeKey() (err error) {
98     if NTLMSSP_NEGOTIATE_EXTENDED_SESSIONSECURITY.IsSet(n.NegotiateFlags) {
99         n.keyExchangeKey = hmacMd5(n.sessionBaseKey, concat(n.serverChallenge, n.lmChallengeResponse))
100     } else {
101         n.keyExchangeKey, err = kxKey(n.NegotiateFlags, n.sessionBaseKey, n.lmChallengeResponse)
102     }
103     return
104 }
105
106 ✓ func (n *V1Session) calculateKeys(ntlmRevisionCurrent uint8) (err error) {
107     // This lovely piece of code comes courtesy of an the excellent Open Document support system
108     // In order to calculate the keys correctly when the client has set the NTLMRevisionCurrent
109     // We must treat the flags as if NTLMSSP_NEGOTIATE_LM_KEY is set.
110     // This information is not contained (at least currently, until they correct it) in the MS-
111     if ntlmRevisionCurrent == 15 {
112         n.NegotiateFlags = NTLMSSP_NEGOTIATE_LM_KEY.Set(n.NegotiateFlags)
113     }
114
115     n.ClientSigningKey = signKey(n.NegotiateFlags, n.exportedSessionKey, "Client")
116     n.ServerSigningKey = signKey(n.NegotiateFlags, n.exportedSessionKey, "Server")
117     n.ClientSealingKey = sealKey(n.NegotiateFlags, n.exportedSessionKey, "Client")
118     n.ServerSealingKey = sealKey(n.NegotiateFlags, n.exportedSessionKey, "Server")
119 }
```

```
417         m.ClientSessionKey = sessionKey(m.NegotiateFlags, m.ExportSessionKey, Client)
```



```
400         }
401
402         err = n.calculateKeys(cm.Version.NTLMRevisionCurrent)
403         if err != nil {
404             return err
405         }
406
407         n.clientHandle, err = rc4Init(n.ClientSealingKey)
408         if err != nil {
409             return err
410         }
411         n.serverHandle, err = rc4Init(n.ServerSealingKey)
412         if err != nil {
413             return err
414         }
415
416         return nil
417     }
418
419     func (n *V1ClientSession) GenerateAuthenticateMessage() (am *AuthenticateMessage, err error) {
420         am = new(AuthenticateMessage)
421         am.Signature = []byte("NTLMSSP\x00")
422         am.MessageType = uint32(3)
423         am.LmChallengeResponse, _ = CreateBytePayload(n.lmChallengeResponse)
424         am.NtChallengeResponseFields, _ = CreateBytePayload(n.ntChallengeResponse)
425         am.DomainName, _ = CreateStringPayload(n.userDomain)
426         am.UserName, _ = CreateStringPayload(n.user)
427         am.Workstation, _ = CreateStringPayload("RULER")
428         am.EncryptedRandomSessionKey, _ = CreateBytePayload(n.encryptedRandomSessionKey)
429         am.NegotiateFlags = n.NegotiateFlags
430         am.Version = &VersionStruct{ProductMajorVersion: uint8(5), ProductMinorVersion: uint8(1), P
431         return am, nil
432     }
433     func (n *V1ClientSession) GenerateAuthenticateMessageAV() (am *AuthenticateMessage, err error) {
434         return nil, nil
435     }
436
437     func (n *V1ClientSession) computeEncryptedSessionKey() (err error) {
```

```
438         if NTLMSSP_NEGOTIATE_KEY_EXCH.IsSet(n.NegotiateFlags) {
439             n.exportedSessionKey = randomBytes(16)
440             n.encryptedRandomSessionKey, err = rc4K(n.keyExchangeKey, n.exportedSessionKey)
441             if err != nil {
442                 return err
443             }
444         } else {
445             n.encryptedRandomSessionKey = n.keyExchangeKey
446         }
447         return nil
448     }
449
450     /*****
451     NTLM V1 Password hash functions
452     *****/
453
454     func ntowfv1(passwd string) []byte {
455         return md4(utf16FromString(passwd))
456     }
457
458     // ConcatenationOf( DES( UpperCase( Passwd)[0..6], "KGS!@#$$" ), DES( UpperCase( Passwd)[7..13],
459     ✓ func lmowfv1(passwd string) ([]byte, error) {
460         asciiPassword := []byte(strings.ToUpper(passwd))
461         keyBytes := zeroPaddedBytes(asciiPassword, 0, 14)
462
463         first, err := des(keyBytes[0:7], []byte("KGS!@#$$"))
464         if err != nil {
465             return nil, err
466         }
467         second, err := des(keyBytes[7:14], []byte("KGS!@#$$"))
468         if err != nil {
469             return nil, err
470         }
471
472         return append(first, second...), nil
473     }
```