

 Etienne Stalmans SEAL and UNSEAL (basically same thing, should... 0cbbff5 · 8 years ago History

```
1 //Copyright 2013 Thomson Reuters Global Resources. BSD License please see License file
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 }
```

```
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.ntChallengeResponse)))
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
75             // response to the server challenge when NTLMv1 authentication is used.
```



```
307     *****/
308
309     type V1ClientSession struct {
310         V1Session
311     }
312
313     /*
314     NTLMSSP_NEGOTIATE_56: true
315     NTLMSSP_NEGOTIATE_KEY_EXCH: true
316     NTLMSSP_NEGOTIATE_128: true
317     NTLMSSP_NEGOTIATE_VERSION: true
318     NTLMSSP_NEGOTIATE_EXTENDED_SESSIONSECURITY: true
319     NTLMSSP_NEGOTIATE_ALWAYS_SIGN: true
320     NTLMSSP_NEGOTIATE_NTLM: true
321     NTLMSSP_REQUEST_TARGET: true
322     NTLM_NEGOTIATE_OEM: true
323     NTLMSSP_NEGOTIATE_UNICODE: true
324     */
325
326     func (n *V1ClientSession) GenerateNegotiateMessage() (nm *NegotiateMessage, err error) {
327         flags := uint32(0)
328         flags = NTLMSSP_NEGOTIATE_KEY_EXCH.Set(flags)
329         flags = NTLMSSP_NEGOTIATE_56.Set(flags)
330         flags = NTLMSSP_NEGOTIATE_128.Set(flags)
331         flags = NTLMSSP_NEGOTIATE_VERSION.Set(flags)
332         flags = NTLMSSP_NEGOTIATE_EXTENDED_SESSIONSECURITY.Set(flags)
333         flags = NTLMSSP_NEGOTIATE_ALWAYS_SIGN.Set(flags)
334         flags = NTLMSSP_NEGOTIATE_NTLM.Set(flags)
335         flags = NTLMSSP_REQUEST_TARGET.Set(flags)
336         flags = NTLM_NEGOTIATE_OEM.Set(flags)
337         flags = NTLMSSP_NEGOTIATE_UNICODE.Set(flags)
338
339         neg := new(NegotiateMessage)
340         neg.Signature = []byte("NTLMSSP\x00")
341         neg.MessageType = 1
342         neg.NegotiateFlags = flags
343
344         //if NTLMSSP_NEGOTIATE_OEM_DOMAIN_SUPPLIED
345         neg.DomainNameFields = new(PayloadStruct)
346         //if NTLMSSP_NEGOTIATE_OEM_WORKSTATION_SUPPLIED
347         neg.WorkstationFields = new(PayloadStruct)
348
349         neg.Version = &VersionStruct{ProductMajorVersion: uint8(5), ProductMinorVersion
350
351         return neg, nil
352     }
353
354     func (n *V1ClientSession) ProcessChallengeMessage(cm *ChallengeMessage) (err error) {
```



Files



cd032d4



Go to file

ntlm



md4



av_pairs.go



challenge_responses.go



crypto.go



crypto_test.go



helpers.go



helpers_test.go



keys.go

```
355         n.challengeMessage = cm
356         n.serverChallenge = cm.ServerChallenge
357         n.clientChallenge = randomBytes(8)
358
359         // Set up the default flags for processing the response. These are the flags th
360         // in the authenticate message
361         flags := uint32(0)
362         flags = NTLMSSP_NEGOTIATE_KEY_EXCH.Set(flags)
363         // NOTE: Unsetting this flag in order to get the server to generate the signatu
364         flags = NTLMSSP_NEGOTIATE_VERSION.Set(flags)
365         flags = NTLMSSP_NEGOTIATE_EXTENDED_SESSIONSECURITY.Set(flags)
366         flags = NTLMSSP_NEGOTIATE_TARGET_INFO.Set(flags)
367         flags = NTLMSSP_NEGOTIATE_IDENTIFY.Set(flags)
368         flags = NTLMSSP_NEGOTIATE_ALWAYS_SIGN.Set(flags)
369         flags = NTLMSSP_NEGOTIATE_NTLM.Set(flags)
370         flags = NTLMSSP_NEGOTIATE_DATAGRAM.Set(flags)
371         flags = NTLMSSP_NEGOTIATE_SIGN.Set(flags)
372         flags = NTLMSSP_REQUEST_TARGET.Set(flags)
373         flags = NTLMSSP_NEGOTIATE_UNICODE.Set(flags)
374
375         n.NegotiateFlags = flags
376
377         err = n.fetchResponseKeys()
378         if err != nil {
379             return err
380         }
381
382         err = n.computeExpectedResponses()
383         if err != nil {
384             return err
385         }
386
387         err = n.computeSessionBaseKey()
388         if err != nil {
389             return err
390         }
391
392         err = n.computeKeyExchangeKey()
393         if err != nil {
394             return err
395         }
396
397         err = n.computeEncryptedSessionKey()
398         if err != nil {
399             return err
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)
```

go-ntlm / ntlm / ntlmv1.go

↑ Top

Code

Blame

473 lines (397 loc) · 14.5 KB

Raw



```
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 = NegotiateFlags
```

- message_authenticate.go
- message_authenticate_test.go
- message_challenge.go
- message_challenge_test.go
- message_negotiate.go
- negotiate_flags.go
- negotiate_flags_test.go
- ntlm.go
- ntlmv1.go
- ntlmv1_test.go
- ntlmv2.go
- ntlmv2_test.go
- payload.go
- signature.go
- signature_test.go
- version.go
- .gitignore
- License
- README.md
- go.mod

```
429         am.NegotiateFlags = n.NegotiateFlags
430         am.Version = &VersionStruct{ProductMajorVersion: uint8(5), ProductMinorVersion: 0}
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( Password)[0..6], "KGS!@#$$%"), DES( UpperCase( Password)[7..14], "KGS!@#$$%"))
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     }
```