PHP krb5 extension Manual

Authentication Services

Kerberos V / KADM5 / GSSAPI

Introduction

This package allows you to authenticate against kerberos realms, use the GSSAPI library and perform administrative tasks on a kerberos realm.

More information about Kerberos can be found at » http://web.mit.edu/kerberos/www/.

For the GSSAPI the best available reference can be found in RFC2744.

If you have comments, bugfixes, enhancements or want to help in developing this you can send me a mail at » mbechler@eenterphace.org.

Installing/Configuring

Requirements

This extensions requires the presence of a MIT or Heimdal kerberos library in a recent version. For KADM5 functionality the MIT distribution is required.

Installation

KADM5 support is disabled by default, enable it passing *--with-krb5kadm* to *configure*. KADM5 support uses some interal header files of the MIT krb5 distribution. These headers are bundled but you can specify another source (which needs to be a unpacked krb5 distribution with *configure* and *make* run) giving a path to the switch.

When compiling this extension as shared module:

- run *phpize* in the extension directory
- run*./configure* (optionally add *--with-krb5kadm* if you need this functionality)
- run make && make install
- optional: enable your new extension in you php.ini by adding *extension=krb5.so* to it.

When compiling statically into your php binary:

- move this extension into your php distributions *ext/* folder
- make sure that the directory is named "krb5"
- run ./buildconf --force in the root directory of you php distribution
- *./configure* php with your common flags and add *--with-krb5* and optionally *--with-krb5kadm=<path>* where <path> is the path to your mit-krb5 distribution.
- make && make install

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Constants for GSSAPI

These constants are flags for GSSAPI contexts and used by the methods GSSAPIContext::initSecContext() GSSAPIContext::acceptSecContext(). More information is available in RFC2744.

Context Flags

constant	
GSS_C_DELEG_FLAG	Request credential delegation when initiating context, only available when used credentials are forwardable.
GSS_C_MUTUAL_FLAG	Request mutual authentication.
GSS_C_REPLAY_FLAG	Request replay detection.
GSS_C_SEQUENCE_FLAG	
GSS_C_CONF_FLAG	
GSS_C_INTEG_FLAG	
GSS_C_ANON_FLAG	
GSS_C_PROT_READY_FLAG	
GSS_C_TRANS_FLAG	

These constants are flags for GSSAPI credentials and used by the method GSSAPIContext::acquireCredentials().

Credential Flags

constant	
GSS_C_BOTH	Request credentials to be usable for initiating and accepting contexts.
GSS_C_INITIATE	Request credentials to be usable for

	initiating contexts.
GSS_C_ACCEPT	Request credentials to be usable for accepting contexts.

Examples

More examples can be found in the *examples* directory of the distribution.

This simple example shows how to obtain a TGT for a given credential using a password.

Example #1 - Initializing a credential cache

```
<?php
 $ccache = new KRB5CCache();
 $flags = array('tkt_lifetime' => 3600);
 $ccache->initPassword('principal@realm', 'password', $flags);
?>
```

The KRB5CCache class

Introduction

Objects of this class represent credential caches. All credential caches are stored in unique memory caches but may be imported/exported to other ccache formats.

Class synopsis

KRB5CCache

KRB5CCache {

/* Methods */

public array **KRB5CCache::getEntries** (void)

public string **KRB5CCache::getName** (void)

public void **KRB5CCache::initKeytab** (string `$principal`, string `$keytab` [, array `$flags`])

public void **KRB5CCache::initPassword** (string `$principal`, string `$password` [, array `$flags`])

public bool **KRB5CCache::isValid** ([int `$seconds`])

public void **KRB5CCache::open** (string `$source`)

public void **KRB5CCache::save** (string `$destination`)

public void **KRB5CCache::setConfig** (string `$file`)

}

KRB5CCache::getEntries

KRB5CCache::getEntries -- Gets the SPNs for which the ccache contains tickets

Description

public array **KRB5CCache::getEntries** (void)

Gets the SPNs for which the ccache contains tickets.

Parameters

This function has no parameters.

Return Values

Returns an array of SPNs for which tickets exist.

Examples

Example #1 - KRB5CCache::getEntries() example

```
<?php
$ccache = new KRB5CCache();
$ccache->initPassword("test","test");
$entries = $ccache->getEntries();
var_dump($entries);
?>
```

The above example will output something similar to:

```
array(1) {
 [0]=>
 string(32) "krbtgt/MYREALM@MYREALM"
}
```

KRB5CCache::getName

KRB5CCache::getName -- Returns the credential cache identifier for the ccache

Description

public string **KRB5CCache::getName** (void)

Returns the credential cache identifier for the ccache

Parameters

This function has no parameters.

Return Values

Returns the credential cache identifier for the ccache of the form *TYPE:identifier*

Examples

Example #1 - KRB5CCache::getName() example

```
<?php
echo $ccache->getName();
?>
```

The above example will output something similar to:

```
MEMORY:5fds34asd
```

KRB5CCache::initKeytab

KRB5CCache::initKeytab -- Gets a TGT using a key given in a keytable file

Description

public void **KRB5CCache::initKeytab** (string `$principal`, string `$keytab` [, array `$flags`])

Gets a TGT using a key given in a keytable file

Parameters

`principal`
: Name of principal to get TGT for.

`keytab`
: Path to keytable file which contains a suitable key for *$principal*

`flags`
: Associative array of Ticket flags.

Usable ticket flags

flag	type	description
forwardable	bool	Try to get a forwardable TGT
proxiable	bool	Try to get a proxiable TGT
tkt_life	integer	Lifetime of TGT in seconds
renew_life	integer	Renewable lifetime of TGT in seconds

Examples

Example #1 - KRB5CCache::initPassword() example

```
<?php
$ccache = new KRB5CCache();
$flags = array(
 "forwardable" => true,
 "tkt_life" => 60 * 60
);
```

```
$ccache->initKeytab("test@MYREALM", "/path/to/test-myrealm.keytab", $flags);
// if everything worked $ccache will now contain forwardable a TGT
// for test@MYREALM with lifetime of one hour
?>
```

KRB5CCache::initPassword

KRB5CCache::initPassword -- Gets a TGT using a given password

Description

public void **KRB5CCache::initPassword** (string `$principal`, string `$password` [, array `$flags`])

Gets a TGT using a given password. Please note that using passwords is not suitable for services needing to accept GSSAPI contexts as the service key is required but then not available to GSSAPI.

Parameters

`principal`
Name of principal to get TGT for.

`password`
Password to authenticate.

`flags`
Associative array of Ticket flags.

Usable ticket flags

flag	**type**	**description**
forwardable	bool	Try to get a forwardable TGT
proxiable	bool	Try to get a proxiable TGT
tkt_life	integer	Lifetime of TGT in seconds
renew_life	integer	Renewable lifetime of TGT in seconds

Examples

Example #1 - KRB5CCache::initPassword() example

```
<?php
$ccache = new KRB5CCache();
$flags = array(
 "forwardable" => true,
 "tkt_life" => 60 * 60
);
$ccache->initPassword("test@MYREALM", "test", $flags);
// if everything worked $ccache will now contain forwardable a TGT
// for test@MYREALM with lifetime of one hour
?>
```

KRB5CCache::isValid

KRB5CCache::isValid -- Checks whether the credentials contained are still valid

Description

public bool **KRB5CCache::isValid** ([int `$seconds`])

Checks whether the credentials contained are still valid or will be still valid after a given amount of time. Make sure to add a reasonable amount of time to compensate for clock skew.

Parameters

seconds
  Number of seconds for which the credentials should stay valid.

Return Values

Returns *true* if the credentials are valid and *false* otherwise.

Examples

Example #1 - KRB5CCache::isValid() example

```
<?php
$ccache->isValid(); // checks whether the credential cache is still valid
$ccache->isValid(3600); // checks whether the credential cache is still
valid in an hour
?>
```

KRB5CCache::open

KRB5CCache::open -- Copies the credentials contained in some credential cache to this credential cache

Description

public void **KRB5CCache::open** (string `$source`)

Copies the credentials contained in some credential cache to this credential cache.

Parameters

`destination`
    Identifier of the destination credential cache

KRB5CCache::save

KRB5CCache::save -- Copies the contents of the credential cache to another one

Description

public void **KRB5CCache::save** (string `$destination`)

Copies the contents of the credential cache to another one. Please note that only the credentials currently available will be stored, so you will have to save the credential cache *after* service credentials are obtained through e.g. GSSAPI.

Parameters

destination
: Identifier of the destination credential cache (e.g. *FILE:/some/path/to/ccache*)

Examples

Example #1 - KRB5CCache::open() example

```php
<?php
$ccache = new KRB5CCache();
$ccache->initPassword('test@MYREALM', 'test');
$ccache->save('FILE:/tmp/my.ccache');

$ccache2 = new KRB5CCache();
$ccache2->open('FILE:/tmp/my.ccache');
// $ccache2 will now contain the TGT for test@MYREALM
?>
```

KRB5CCache::setConfig

KRB5CCache::setConfig -- Sets the kerberos confiuration file to use

Description

public void **KRB5CCache::setConfig** (string `$file`)

Sets the kerberos confiuration file to use

Parameters

`file`
  Path to configuration file

}

The GSSAPIContext class

Introduction

Objects of this class represent GSSAPI security contexts. It is suitable for both initiating and accepting security contexts. It is designed to work with the kerberos mechanism.

Class synopsis

GSSAPIContext

GSSAPIContext {

/* Methods */

public bool **GSSAPIContext::acceptSecContext** (string `$input_token` [, string `&$output_token` [, string `&$src_name` [, int `&$ret_flags` [, int `&$time_rec` [, KRB5CCache `$deleg`]]]]])

public void **GSSAPIContext::acquireCredentials** (KRB5CCache `$ccache` [, string `$principal` [, int `$type`]])

public string **GSSAPIContext::getMic** (string `$message`)

public bool **GSSAPIContext::initSecContext** (string `$target` [, string `$input_token` [, int `$reqflags` [, int `$timereq` [, string `&$output_token` [, string `&$ret_flags` [, string `&$time_rec`]]]]]])

public void **GSSAPIContext::inquireCredentials** (void)

public void **GSSAPIContext::registerAcceptorIdentity** (string `$keytab`)

public bool **GSSAPIContext::unwrap** (string `$input`, string `&$output`)

public boolean **GSSAPIContext::verifyMic** (string `$message`, string `$mic`)

public bool **GSSAPIContext::wrap** (string `$input`, string `&$output` [, boolean `$encrypt`])

}

GSSAPIContext::acceptSecContext

GSSAPIContext::acceptSecContext -- Accepts a GSSAPI context initiated by a remote party

Description

public bool **GSSAPIContext::acceptSecContext** (string `$input_token` [, string `&$output_token` [, string `&$src_name` [, int `&$ret_flags` [, int `&$time_rec` [, KRB5CCache `$deleg`]]]]])

Accepts a GSSAPI context initiated by a remote party. **GSSAPIContext::acquireCredentials** should first be used to select the correct credentials, otherwise the default credential cache and keytab will be used.

Parameters

`input_token`
: Token passed by the remote party.

`output_token`
: Token to pass to the remote party.

`src_name`
: Principal name of the authenticated remote party

`ret_flags`
: Flags of the established GSSAPI context.

`time_rec`
: Time in seconds for which the context will stay valid.

`deleg`
: The given credential cache will be reinitalized and filled using delegated credentials, if available.

Return Values

Will return *true* if the context was established, *false* otherwise.

Examples

Example #1 - GSSAPIContext::acceptSecContext() example

```
<?php
```

```
// assume $client is a KRB5CCache containing credentials for client@MYREALM
// assume $server is a KRB5CCache containing credentials for server@MYREALM
(initialized using keytab)

$cgssapi = new GSSAPIContext();
$cgssapi->acquireCredentials($client);

$sgssapi = new GSSAPIContext();
$sgssapi->acquireCredentials($server);

$token = '';
$cgssapi->initSecContext("server@MYREALM", null, null, null, $token);

$token2 = '';
$remote = '';
$sgssapi->acceptSecContext($token, $token2, $remote);

echo $remote;
?>
```

The above example will output something similar to:

```
client@MYREALM
```

See Also

- **GSSAPIContext::initSecContext**
- **GSSAPIContext::acquireCredentials**
- **KRB5CCache::initKeytab**

GSSAPIContext::acquireCredentials

GSSAPIContext::acquireCredentials -- Obtains credentials for establishing a GSSAPI context

Description

public void **GSSAPIContext::acquireCredentials** (KRB5CCache `$ccache` [, string `$principal` [, int `$type`]])

Obtains credentials for establishing a GSSAPI context. If the credentials shall be used for accepting a GSSAPI context, the given credential cache must be initialized with a keytab, as access to the service key ist required.

Parameters

`ccache`
: Credential cache to fetch credentials from.

`principal`
: Principal to acquire credentials for

`type`
: Type of credentials to acquire (GSS_C_BOTH, GSS_C_ACCEPT, GSS_C_INIT)

Examples

Example #1 - GSSAPIContext::acquireCredentials() example

```
<?php
$ccache = new KRB5CCache();
$ccache->initPassword('test@MYREALM', 'test');

$gssapi = new GSSAPIContext();
$gssapi->acquireCredentials($ccache);

$token = '';
$gssapi->initSecContext('server@MYREALM', null, null, null, $token);
// the context will be initiated as test@MYREALM
?>
```

See Also

- **GSSAPIContext::inquireCredentials**

GSSAPIContext::getMic

GSSAPIContext::getMic -- Calculate a MIC on a given message

Description

public string **GSSAPIContext::getMic** (string `$message`)

Calculate a MIC on a given message. For remote verification both the message and the returned MIC have to be passed to the remote party.

Parameters

`message`
: Message to calculate MIC on.

Return Values

Returns the MIC valid for the current GSSAPI context.

Examples

Example #1 - GSSAPIContext::getMic() example

```
<?php
// assume $cgssapi is the initiator of some context
// and $sgssapi is the acceptor of the context

$message = 'foo';

$mic = $cgssapi->getMic($message);

$sgssapi->verifyMic($message, $mic); // will return true if valid
?>
```

See Also

- **GSSAPIContext::verifyMic**

GSSAPIContext::initSecContext

GSSAPIContext::initSecContext -- Initiates a GSSAPI security context

Description

public bool **GSSAPIContext::initSecContext** (string `$target` [, string `$input_token` [, int `$reqflags` [, int `$timereq` [, string `&$output_token` [, string `&$ret_flags` [, string `&$time_rec`]]]]]])

This method initiates a GSSAPI context. **GSSAPIContext::acquireCredentials** should first be used to select the correct credentials, otherwise the default credential cache will be used.

Parameters

`target`
SPN of principal to establish security context with.

`input_token`
GSSAPI Token passed by the context acceptor (for the kerberos mechanism only required if mutual authentication is performed).

`reqflags`
GSSAPI Context flags (see the Constants section, defaults to no flags)

`timereq`
Time in seconds which the context should stay valid (defaults to 0, which means the context will stay valid as long as possible)

`output_token`
Token to pass to the context acceptor for authentication

`ret_flags`
Flags of the (possibly not yet established) security context

`time_rec`
Time in seconds which the context will stay valid

Return Values

Returns *true* if the context is fully established and *false* otherwise.

Examples

Example #1 - GSSAPIContext::initSecContext() example

```
<?php
// assume $client is a KRB5CCache containing credentials for client@MYREALM
// assume $server is a KRB5CCache containing credentials for server@MYREALM
(initialized using keytab)

$cgssapi = new GSSAPIContext();
$cgssapi->acquireCredentials($client);

$sgssapi = new GSSAPIContext();
$sgssapi->acquireCredentials($server);

$token = '';
$cgssapi->initSecContext("server@MYREALM", null, null, null, $token);

$token2 = '';
$remote = '';
$sgssapi->acceptSecContext($token, $token2, $remote);

echo $remote;
?>
```

The above example will output something similar to:

```
client@MYREALM
```

See Also

- **GSSAPIContext::acceptSecContext**
- **GSSAPIContext::acquireCredentials**

GSSAPIContext::inquireCredentials

GSSAPIContext::inquireCredentials -- Gets information about the credentials used for context establishment

Description

public void **GSSAPIContext::inquireCredentials** (void)

Gets information about the credentials associated with the GSSAPIContext.

Parameters

This function has no parameters.

Return Values

Will return an associative array of

Returned information

key	type	description
name	string	Prinicipal (local)
lifetime_remain	int	Remaining time in seconds for which the credentials are valid
cred_usage	integer	Possible credential usage (GSS_C_BOTH, GSS_C_INITIATE, GSS_C_ACCEPT)
mechs	array of strings	OIDs of usable mechanisms

See Also

- **GSSAPIContext::acquireCredentials**

GSSAPIContext::registerAcceptorIdentity

GSSAPIContext::registerAcceptorIdentity -- Register credentials for context establishment (acceptor side)

Description

public void **GSSAPIContext::registerAcceptorIdentity** (string `$keytab`)

Registers a keytab with the GSSAPI context from which the service credentials will be extracted.

Parameters

`keytab`
Path to keytab from which the service key will be extracted when accepting a context.

Return Values

Returns nothing.

See Also

- **GSSAPIContext::acceptSecurityContext**

GSSAPIContext::unwrap

GSSAPIContext::unwrap -- Unwraps a previously wrapped message

Description

public bool **GSSAPIContext::unwrap** (string `$input`, string `&$output`)

Unwraps a previously wrapped message, which means that the embedded MIC will be verified and the message possibly be decrypted.

Parameters

`input`
: Input token

`output`
: Message (possibly decrypted)

Return Values

Returns *true* if successful and *false* otherwise.

Examples

Example #1 - GSSAPIContext::unwrap() example

```php
<?php
/* ... */

$message = 'test';
$decoded = '';
$token = '';


$sgssapi->wrap($message, $token, true); // adds a MIC and encrypts $message

$cgssapi->unwrap($token, $decoded);
// returns true when the embedded MIC is valid
// $decoded will contain test
?>
```

See Also

- **GSSAPIContext::wrap**

GSSAPIContext::verifyMic

GSSAPIContext::verifyMic -- Verifies a MIC calculated by the remote party

Description

public boolean **GSSAPIContext::verifyMic** (string `$message`, string `$mic`)

Verifies a MIC calculated by the remote party

Parameters

`message`
: Message to verify MIC

`mic`
: MIC

Return Values

true if the passed MIC is valid for the message and *false* otherwise.

Examples

Example #1 - GSSAPIContext::verifyMic() example

```
<?php
// assume $cgssapi is the initiator of some context
// and $sgssapi is the acceptor of the context

$message = 'foo';

$mic = $cgssapi->getMic($message);

$sgssapi->verifyMic($message, $mic); // will return true if valid
?>
```

See Also

- **GSSAPIContext::getMic**

GSSAPIContext::wrap

GSSAPIContext::wrap -- Wraps a message

Description

public bool **GSSAPIContext::wrap** (string `$input`, string `&$output` [, boolean `$encrypt`])

Wraps a message, which means that the message and a MIC will be combined into a token which may be passed to a remote party. Optionally the message can be also encrypted.

Parameters

input
: Message to wrap

output
: Wrapped token

encrypt
: Whether to encrypt the message

Return Values

Returns *true* if successful and *false* otherwise.

Examples

Example #1 - GSSAPIContext::wrap() example

```
<?php
/* ... */

$message = 'test';
$decoded = '';
$token = '';


$sgssapi->wrap($message, $token, true); // adds a MIC and encrypts $message

$cgssapi->unwrap($token, $decoded);
// returns true when the embedded MIC is valid
// $decoded will contain test
?>
```

See Also

- **GSSAPIContext::unwrap**

The KRB5NegotiateAuth class

Introduction

This class provides a simple interface for doing HTTP Negotiate/SPNEGO authentication. This interface might be obsolete, as this functionality might be easily implemented using the GSSAPI interface. Please note that this is only possible for SAPIs which have access to the *Authorization* header (e.g. *not* using the CGI SAPI).

Class synopsis

KRB5NegotiateAuth

KRB5NegotiateAuth {

/* Methods */

KRB5NegotiateAuth::__construct (string `$keytab`)

public bool **KRB5NegotiateAuth::doAuthentication** (void)

public string **KRB5NegotiateAuth::getAuthenticatedUser** (void)

public void **KRB5NegotiateAuth::getDelegatedCredentials** (KRB5CCache `$ccache`)

}

KRB5NegotiateAuth::__construct

KRB5NegotiateAuth::__construct -- Initialize the negotiate auth handler

Description

KRB5NegotiateAuth::__construct (string `$keytab`)

Initialize the negotiate auth handler

Parameters

`keytab`
: Path to the keytab containing the service key for the principal *HTTP/server.f.q.d.n@MYREALM*

See Also

- **KRB5NegotiateAuth::doAuthentication**

KRB5NegotiateAuth::doAuthentication

KRB5NegotiateAuth::doAuthentication -- Performs HTTP Negotiate authentication

Description

public bool **KRB5NegotiateAuth::doAuthentication** (void)

This method performs HTTP Negotiate authentication. It will fetch the provided credentials from the request headers and set the response headers appropriately to perform authentication.

Parameters

This function has no parameters.

Return Values

Returns *true* when authentication was successful, *false* otherwise.

Examples

Example #1 - KRB5NegotiateAuth::doAuthentication() example

```
<?php
$auth = new KRB5NegotiateAuth('/etc/krb5.keytab');


if($auth->doAuthentication()) {
       echo 'Success - authenticated as ' . $auth->getAuthenticatedUser();

       try {
               $cc = new KRB5CCache();
               $auth->getDelegatedTicket($cc);
       } catch (Exception $error) {
               echo 'No delegated credentials available';
       }
} else {
       if(!empty($_SERVER['PHP_AUTH_USER'])) {
               header('HTTP/1.1 401 Unauthorized');
               header('WWW-Authenticate: Basic', false);
       } else {
               // verify basic authentication data
               echo 'authenticated using BASIC method<br />';
       }
}

?>
```

See Also

- **Classname::Method**

KRB5NegotiateAuth::getAuthenticatedUser

KRB5NegotiateAuth::getAuthenticatedUser -- Returns the principal name of the authenticated user

Description

public string **KRB5NegotiateAuth::getAuthenticatedUser** (void)

Returns the principal name of the authenticated user

Parameters

This function has no parameters.

Return Values

Returns the principal name of the authenticated user

See Also

- **Krb5NegotiateAuth::doAuthentication**

KRB5NegotiateAuth::getDelegatedCredentials

KRB5NegotiateAuth::getDelegatedCredentials -- Gets the possibly delegated credentials of the authenticated user

Description

public void **KRB5NegotiateAuth::getDelegatedCredentials** (KRB5CCache `$ccache`)

Gets the possibly delegated credentials of the authenticated user

Parameters

ccache
: Credential cache that will be initialized using the delegated credentials.

See Also

- **Krb5NegotiateAuth::doAuthentication**