TrustedSec
INFORMATION SECURITY MADE SIMPLE

SHIPS Installation Manual

Shared Host Integrated Password System (SHIPS) Tutorial Version 1

Prepared by: Geoff Walton - TrustedSec

Prepared for:
For Public Release

info@trustedsec.com ■ 11565 Pearl Rd. Suite 301 ■ Strongsville, OH 44136 ■ 877.550.4728

Table of Contents

1 SHIPS Summary

SHIPS is a solution to provide unique and rotated local super user or administrator passwords for environments where it is not possible or not appropriate to disable these local accounts. Clients may be configured to rotate passwords automatically. Stored passwords can be retrieved by desktop support personnel as required, or updated when a password has to be manually changed in the course of system maintenance. By having unique passwords on each machine and logging of password retrievals, security can be improved by making networks more resistant to lateral movement by attackers and enhancing the ability to attribute actions to individual persons.

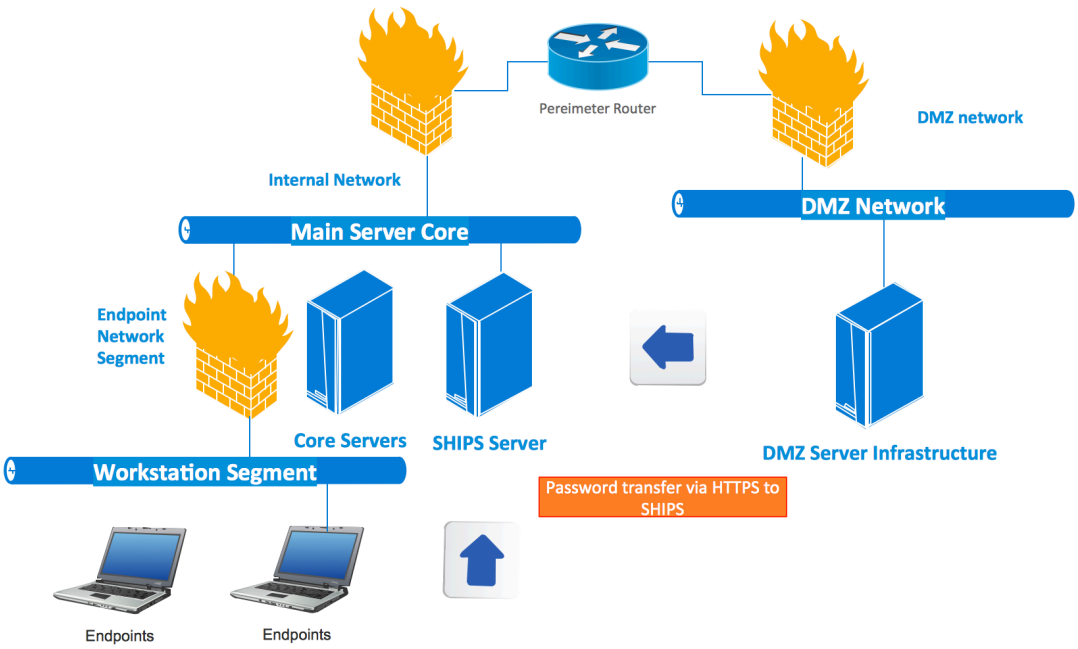
When performing penetration tests, our common attack vector is through compromising one host and pivoting to other systems with the information obtained. It is common to see large-scale breaches utilizing this method and that is where SHIPS comes into play.

SHIPS is designed to make post-exploitation more difficult and minimize what systems attackers gain access to. Once SHIPS is set up, there isn't much else that is needed and it's simple to integrate into existing business processes.

1.1 Project Goals:

- A complete solution packaged as a single application which can be deployed on a variety of platforms
- Deployments should be simple to move or relocate (this may be required in disaster recovery situations)
- Immediately useable with little or no training for support personnel
- Low resource consumption on server and clients
- Low impact on WANs
- Support a wide variety of clients
- Simple client protocol so various operating systems and devices can be integrated with the server through shell scripts and utilities such as cURL
- Simple to integrate with external directories or asset management tools
- Ability to easily script interaction with the server in order to facilitate system deployment processes, or integrate with other support tools

1.2 How it works:



A script is deployed to the endpoints, servers, and any other systems through group policy or similar deployment tools. The script is run on a determined timeframe from the organization. The script makes a password request to the server, which generates unique password string that it stores and transmits to the client. The client script than applies the new password on the client. TrustedSec recommends deploying SHIPS to servers, workstations, or any other windows-based systems. The passwords will now be unique per individual server and workstation.

For organizations where client and server support roles are segregated to different groups of employees, multiple instances of the SHIPS server can be run on a single host. Simply change the listing port on one of server instances and configure each to authorize the appropriate users. TrustedSec recommends using the alternate listening port for the instance supporting server infrastructure. In most cases accommodating requests on the alternate port from servers will be easier than frequently more mobile clients, with regard to firewalls or proxies.

When users with permission to access account passwords wish to retrieve them, they simply log into the SHIPS admin server and do a lookup of the machine name. The web application will display the current password associated with the device. SHIPS authorization can be tied to external systems such as LDAP.

2 Architecture

Client Password Server is implemented as a standalone Ruby application. The application is web based, however, http handling is provided by WEBrick, an in-process webserver included in the standard library. This application has minimal external dependences, gems sqlite and optionally ruby-ldap. The ruby-ldap dependency is not required if installation of ruby-ldap is a problem on your platform, LDAP integration will not be available if omitted. Keeping external dependencies minimized allows the application to be installed on a host that may have existing web services running and alongside applications that may use large frameworks such as RAILS without the need for a separate Ruby environment. It also helps ensure the application can be stood up quickly in a disaster recovery scenario.

Clients perform password updates using a single https request. The request and response are kept to a minimal size to facilitate large numbers of clients across WAN links while keeping impact and data utilization low. Remote clients can be supported as well as this request may be handled securely over the Internet and be passed by proxies if the client script implementation such as the VBS client provided for Windows clients supports it.

Support personnel access the information using a web interface. Functions such as creating, removing, and clearing passwords on computer objects may also be accessed via a web service call that does not require sessions or cookies for transactions with scripts.

2.1 Security considerations:

- The server can be configured to run as an un-privileged user.
- It is necessary to ensure the directory permissions of the server files are correct, for example, do not allow access to the database file by users who should not have access to the local account passwords. Guidance on file permissions is provided in the installation section.

- The server will validate the computer name exists in the database or external directory before issuing a password and storing the response. This provides basic protection from a DOS attack where the attacker would attempt to fill the password database.

- The server issues a nonce value to each machine along with the password, and requires the correct value be submitted with future password requests,

preventing attempts to corrupt the password database. This weakly authenticates the client. It may still be possible for an attacker to brute force the nonce making a large number of requests; this method was selected to keep client scripts simple while still providing some protection.

- When deploying the client script, the directory in which the script resides and the directory where the history file (stores suggested next update time, and nonce) is stored should be writeable only by groups who would otherwise have access to change the administrator password on the machine, Administrators. If file permissions allow a non-administrative user to alter the client script, this could create a local privilege escalation path; use care deploying client scripts.

- From a recovery standpoint, if the password database is ever lost or damaged it should automatically rebuild correctly as clients check in to update their passwords. The server will see the machine as a new client but valid in the directory and will issue a new password. The database can be otherwise be backed up by simply copying the SQLite file.

3 Installation

1) Install basic platform requirements
    a. Obtain Ruby 1.9.3 or later packages from your operation system vendor. Many platforms will already have a compatible Ruby installation.
    `ruby –v`
    will report the current version and exit if Ruby is installed.
    b. Install ruby sqlite3
    On most Linux and Unix platforms this can be accomplished with:
    `gem install sqlite3`
    Some documentation for installing SQLite3 and the Ruby gem for Windows is available here:
    http://stackoverflow.com/questions/15480381/how-do-i-install-sqlite3-for-ruby-on-windows
    c. Install Ruby ldap
    `gem install ruby-ldap`
    d. Install Ruby USA
    `gem install usa-0.8.?.gem`
    (You should have download a USA gem with the project)

2) Unpack the Application Archive
    a. Change to a directory where you wish to store the application, such as /opt
    b. `tar –xvf /path/to/archive.tar`
3) Create service and groups accounts or select an existing account such as "httpd" for the application. The service accounts primary group should be the group account you selected for use. Both are assumed to be "httpd" in this document. Consult your operating system documentation for account setup.
4) Set permissions
    a. On most Linux/Unix platforms appropriate permissions should be:
        i. root.root 755 for directories
        ii. root.httpd 640 for files
    b. The exceptions to this are:
        i. Passwordserver.rb root.root 750
        ii. ./var/log httpd.root 770
        iii. ./var/data httpd.root 770
    c. Note: if you change the location of the database or log files you may need to modify the permissions on the new location accordingly
5) Generate and obtain an SSL certificate
    a. The application expects these files in pem format with no password. Your clients must trust this certificate.

b. To create a new certificate:
openssl genrsa –aes256 -out ssl.key.orig 2048

openssl req -new -key server.key.orig -out ssl.csr
Complete the prompts for the certificate fields.

openssl rsa -in ssl.key.org -out ssl.key

Present the file ssl.csr to a Certificate authority for signing, save the response as ssl.pem (You may need to convert to pem format if the response is delivered differently)

When finished, the files ssl.pem and ssl.key should be in the ./etc directory. You may remove the other files.

6) Minimal configuration
This will provide a basic server using local database authentication and no external computer validation directory; this is good starting configuration for environments where no external directory integration will be used, and it will allow clients to register themselves. Leaving the server in this configuration is not advised, as it is vulnerable to a denial of service attack.
   a. Open the file ./etc/conf - The start of this file contains a skeletal config commented out, along with a short description of the configuration options. The configuration file uses YAML grammar the # indicates a comment. Lines starting with # may be removed if desired.
   b. In the web: section
      i. Locate the line Port: 443 and modify the port number if required; if you are already running an TLS/SSL enabled web server, most likely you may need to use a different port
      ii. Locate the line daemonUser: and set the value to the user you selected in step 2
      iii. Locate the line serverName: Set this to the DNS name of the server as clients understand it; this must match the CN name set on the server certificate or one of the SAN names if you used an existing certificate.
7) Arrange for your platform to start the application, for example, by adding it to /etc/rc.d/rc.local on many Linux systems. The application should be started as the system super user. This is necessary so that it may bind ports and perform a call to su.

   For many deployments, authoring a shutdown script may not be required as it is common for many Linux platforms to send a TERM signal to all processes as part of the shutdown procedure.

If you do author a shutdown of the script, it should simply send a TERM signal to the process. The process will not terminate immediately but may take several seconds to ensure all changes are committed to the database file.

8) If you have used identSQLite, for authentication, a default user account “admin” with password “admin” will have been created. The password should be changed as soon as possible.

4 Server Configuration

The configuration file is located at ./etc/conf in the application root. This file uses YAML grammar, it has multiple sections, and each is in the form a hash, starting in the left most column. This file is case sensitive. Values are required unless otherwise noted.

The **web** section – defines a number of WWW server level properties. It has the following members:

1) port: A numeric value expressing TCP port number, usually 443
2) daemonUser: a string value naming the user account the service should become after binding ports
3) Log: a string value giving the path to the log file the application should use. This may be omitted and defaults to ./var/log/passwordserver.log
4) SSLCertificate: a string expressing the path to the SSL certificate; this may be omitted, and defaults to ./etc/ssl.pem
5) SSLPrivateKey: a string expressing the path to the SSL private key in pem format; this may be omitted defaulting to ./etc/ssl.key
6) serverName: This is the host name for the webserver. This does not need to be the system host name but must be the name clients use when connecting to the host; the host should be able to resolve this name to itself. This value is a string.
7) sessionTimeout: a numeric value specifying the number of seconds a client session may be idle before it is logged off.

The **app** section – defines most application and service settings. It has following the members.

1) forground: a boolean value, when true, prevents the application from detaching the console; this is useful for debugging with Ruby debugging tools such as pry. This may be omitted and defaults to false
2) datapath: a string value expressing the path to the SQLite datafile, this may be omitted and defaults to ./var/data/passwordserver.sqlite
3) owner_uid: a string specifying the UID of the “Super User”, this user has special permissions within the application. When using the SQL identity back end, the super user alone may manage the accounts of other users. This user may also perform other operations like manually adding or deleting computers or exporting the database.
4) owner_ident: a string specifying the ident class the super user may sign in with. **Note:** if this user’s UID exists in multiple ident classes, they will only be treated as the superuser when logging on with this method.

5) login_idents: an array of strings, naming one or more identity (ident) classes that may be used to authenticate users to the application. In most cases single Ident class is all that is required, however, there are cases where multiple Ident backends may be useful; for example, you wish to have users authenticated against active directory using identLDAP, but you also want the root user on the local system to be able to authenticate with their password from the local /etc/shadow file in the event that the LDAP server is unavailable using identETC.

   The following Ident classes are built in
   a. IdentETC (Linux only) uses /etc/password and /etc/shadow
   b. IdentSQLite provides authentication against a table of users in the applications SQLite file; these passwords are stored using a salted SHA hash.
   c. IdentLDAP provides authentication against a remote LDAP server, allows specifying an LDAP group membership requirement for authentication to succeed.

   It is possible to provide your own ident class, passed as an additional module for Ruby to load when executing the server. It would be specified here as well.
6) defaultIdent: a string naming the Ident class to use when a user accesses a page that requires a session and they do not have a session, or when any other resource requires a login method and one is not provided in the request. This must be one of the specified login_idents
7) ComputerValidators: an array of zero of more strings specifying the validator classes to check computer names against when a computer name is not already present in the database. ValidateAny is a built-in that matches all names; this can be used to allow clients to initially populate a database when not using an external directory source for names. ValidateLDAP is also provided which looks for names within an LDAP directory. As with Ident a custom validator class you are preloading may be specified here well.
8) styleSheet: a string specifying a path to a CSS document to use for interactive web pages. This may be omitted and ./etc/style.css will be used as a default.
9) syslog: a boolean, when true, messages are written to syslog when passwords are stored or retrieved. This may be simply omitted to disable syslog. It is not available on Windows platforms.

The **rules** section defines rules for computers.

1) age: a numeric value specifying the suggested password rotate time in days
2) length: a numeric value specifying the number of characters for generated passwords; do not set a value less that of 7.

The **moduleOptions** section is a hash of options to be passed to Ident classes and Validator classes when they are instantiated; this may be used to pass parameters to a custom Ident or Validator classes as well. The ValidateLDAP and IdentLDAP classes use the following values (these may be omitted if these classes are not in use):

ldapGroupDN: a string for the distinguished name of a group whose members are permitted to login.

ldapUserOU: a string for the DN of an OU to subtree search for users who may log in.

ldapComputerOU: a string for the DN of an OU to subtree seach for authorized computer names.

ldapServer: a string value naming the LDAP server or specifying its IP address
ldapPort: a numeric port to connect to LDAP on usually 389
ldapUserDN: a string for the DN naming the LDAP user to bind the directory with when searching.
ldapPassword: a string containing the password belonging to the user in ldapUserDN.

5 Application Quick Start

Most functions are reached from /admin. This page allows you to look up computer information or create new computer accounts if you are the super user.

After looking up a computer, users are presented with the current and previous three passwords, which may be useful in cases where the machine failed to adjust its password after issuing the server request. The current password may be modified, by overtyping it and then clicking the save button. If the machine is out of sync with the server, the nonce value may be returned to the "open" state by pressing clear. This will permit the computer to request a password update with any nonce value on the next request. The delete button can be used to remove a computer that is no longer in the environment from the database.

If a computer needs to set a password but cannot be validated, or if you are not using an external validator, the super user may manually enter computer names and add the computer using the new button. The computer is not stored until the save button is pressed. The super user may also export the complete listing of computers and passwords as tab separated values using the link on this page.

/users
Is accessible to the super user and allows creating, deleting, and password management of other users when IdentSQLlite is used as an Identity class.

/adminws
Provides a non-session method by which commands may be sent to the server. This allows for clients that need to transact using single requests and do not wish to store cookies. This interface is mainly designed to be used by scripts, but will function from a web browser. Use cases are, for example, a Microsoft Deployment Manager script adding a newly deployed computer to the database, or an asset management tool removing a retired computer from the database.

When issued a GET request, this resource will respond with simple a web form that provides information on how to create various POST based transactions. Only the POST method transacts; GET can be sent with a single query string parameter "method" specifying a desired Ident class. If the method is not specified, forms with fields corresponding to the authentication parameters that belong to the default Ident method are generated.

/password
This has been designed for maximum compatibility with proxies, for clients that may

need to set a password via the Internet. **If you have reverse proxy or next-gen firewall in your environment, consider using it to restrict access to only GET requests to this resource from external clients.** Clients do not need to make any other requests to update their passwords.
This is the resource clients use to request password updates. The client sends a simple request via the GET method with the following parameters:

nonce or nouonce – followed by any 32bit unsigned number as a decimal string if the computer has not previously set a password. Otherwise this is the value from the servers response in a previous request.

name – The unique name of the computer.

The server will respond with a minimalistic HTML page, for example:

```
<!DOCTYPE
html><html><body>true,KnBzSmYmWHg2P3ZWO0VnVyZxNDw=,1988738762,2014-11-27
17:59:32</body></html>
```

The response body consists of comma-separated fields:

a. Success – true if the request was accepted. The client should **not** modify the local password if this value is false, and should try the request again later.

b. Message – the base64-encoded password if the request was accepted, otherwise a failure message as a plain string.

c. nonce – the nonce to use with the next request

d. Suggested change time – the server's proposed time for the next update request to be sent. It is up to the client to respect this value, and to retry as required.

6 Deploying the Windows Client Script

Most environments will want to use some type of deployment management solution to propagate the client script to PCs. The following steps will allow manual deployment of the script. The VBS script provided should work on most Windows Clients win2k and above.

0) Create a directory such as C:\password on the Windows client. Ensure permissions on this directory allow only Administrators to have write or modify access.
1) Copy the file setAdminPass.vbs from the ./clients directory on the server to the directory on the client PC.
2) Open the SetAdminPass.vbs file and update two constants at the top of the script.
   QUERYSTRING="https://myserver.example.com/password?"
   HISTORYFILE="c:\password\password_history.txt"
3) Open a command prompt. This must be done with the right-click "run as administrator" method on Windows 7 or Windows Server 2008 or later versions.
4) Run the command:
   schtasks.exe /rl HIGHEST /ru "SYSTEM" /Create /SC HOURLY /TN "Update Administrator Password" /TR "c:\windows\system32\cscript c:\password\SetAdminpass.vbs"

This will call the client script every hour. The first thing the script does on startup is read the history file. If the change time has not been reached, the script exists silently. The file is not updated if the script fails to reach the server. This will enable the client to retry every hour until successful. There will be WSH entries in the Application event log that will indicate if the password was updated or where in the process the update failed.

7 Using IdentETC

The IdentETC identity class is available on Linux platforms only. It offers limited functionality, but may be useful for simple deployments or temporary measures. This Ident class authenticates users against the password stored in /etc/shadow. Because of the permission model used by SHIPS, in most cases this will require a dedicated server with few accounts present, as all user accounts will have permissions to authenticate and retrieve passwords.

Most service accounts such as httpd, main, cron, etc., do not have a password set and will therefore not be able to authenticate to the SHIPS server. You should be aware, however, that if a password has been set for these accounts, they would be able to access the system as well.

0) Change to the directory where your platform stores gems
1) Determine if you need to recompile the shadowtest binary or if the existing file can be used. You will need to be root.
   bash #start a new shell
   uset HISTORYFILE #disable command history
   cd usa-?.?.?/usa/bin
   ./shadowtest –p {root password} –u root
   If "`Authentication successful`" is printed skip to step three (3)
2) To recompile the shadowtest binary for your platform
   cd usa
   gcc –l crypt showdowtest.c –o ../shadowtest
   cd ..
3) Set permissions on the binary
   chown root.shadow shadowtest
   chmod 4755 shadowtest

8 Implementing Validators or Ident Classes

If you wish to implement your own Validator or Identity integration, see the two files identsample.rb, and validatesample.rb in the ./doc directory. These contain skeleton classes with the public methods that must be implemented along with a few comments. If you do not wish to alter the program source files, it is possible to utilize custom classes simply by invoking the server as:

```
ruby –r MyCustomIdent.rb –r MyCustomValidator.rb passwordserver.rb
```

Update your configuration file as appropriate.