Hanbot: S1mple Auth

Last change Sunday , 27. May, 2018

Version 1 Revision 1

S1mple

Contents

Intro

This white-paper serves as a documentation of the actual reference implementation, as a read-me for common errors and also as an in depth explanation about our design choices regarding this authentication (auth).

Goal

The goal of this auth is to provide an easy to use auth for both users and script developers, while maintaining a very high level of security and customizability. A normal user should never find an auth annoying to use, so we tried to simplify it as much as possible. Also a developer should have no issues implementing our auth, even if he never used it before.

For users

This white-paper mostly discusses the technical aspect of the auth. If you got trouble authenticating yourself, please contact me or the scripts author.

For developers

If you want to integrate this auth, take a look at the provided lua and php code. If you need any help with integrating it, please feel free to contact me. In case that you don't want to host the auth server yourself, please contact me, so i can help you integrating it on my server

Auth flow

Please checkout the graphic to find out what i am referring to.

Client

Load scripts The client just has to check the script in the ingame menu. If he has bought it on the website, he will be able to use it, if not he will be given a trial time (given that the author enables this feature)

Script

Load auth module The script loads the auth module and wait for it to return

Load script The script loads normally

Display Error The auth module has returned an error and the script is not supposed to load

Auth module

Verify integrity The auth module does some basic checks to make sure that the script was not tampered

Begin authentification The auth module assembles all necessary data (username, user key, script key, auth url) and then prepares an HTTPS request for the Auth API. It then checks if the HTTPS cert was not tampered and sends the request combined with an challenge.

End auth The Auth API has responded to the request. The auth module will now interpret the data and verifies that the challenge was responded correctly

Auth API

Is correct API call	The api verifies that the request was formed correctly
Return status	The api returns the status of the auth and the challenge response.
Create log entry	The api will call the SQL server to create a log entry of the auth event, containing the status, challenge, user and script
Generate key	Generates a new user key
Send email to buyer	Invokes php mail to send the key to the buyer

SQL Server

ck username and key	The SQL server checks if the username and key exist in the database
Can have trial	The SQL server checks if the user hasn't yet had a trial
Insert log entry	Insert an log entry into the DB
Select log	Selects the relevant log entries (Only for the current script author and other optional params)
Insert user	Inserts a new user into the DB
Modify ban	Modifies the ban status and time of a user
Modify script access	Modifies the users access to specific scripts
Insert script	Inserts a new script into the DB
Modify script	Modifies script name, status and optional parameters
Select scripts	Selects all scripts (or scripts of a author / champ)

Admin website

View log	Displays the log files of the script dev
Insert new user	Manually inserts a user into the DB
(Un)Ban User	Changes the ban status of a user for specific scripts or global [1]
Remove script access	Adds or removes a users access to a specific script
Create script	Create a new script with options and generate auth module
Modify scripts	Modify script options and regenerate auth module

Shop

Get paid scripts	Show all paid scripts or filter by author / champ
Check auth status	Allows a user to check what scripts he is authenticated for
Buy scripts	Invokes an external payment provider to allow a user to buy a script
Show status	Shows if the payment was successful

External payment provider

Bought External API that returns if the user has actually paid.

[1] global bans are only allowed, if you are an super admin or on request.

In depth view on the auth request

The challenges we face when writing an authentication are the following:

- The user should not be annoyed
- The auth has to be difficult to crack
- The hanbot api is limited
- Lua is slow

To solve this challenges we propose the following:
The client side auth module prepares a HTTPS post request. This request contains:

Username The Hanbot username of the client

User key This key gets send to the user whenever he buys the script

Script key This key is generated when you create the script in the admin dashboard. It is embedded into the encrypted auth module.

Challenge This is the most interesting part of the request. Both the client and the server have a shared secret. The client generates a challenge value for the server (a random number) attaches it to the request. The servers response will contain hash(challenge value + secret). This allows the client to check if his hash matches the server one. As hash method i propose SHA-256 for speed and security. Since there is no good way in the Hanbot api to check if the HTTPS cert if real, i propose on a quite large secret and challenge value. In the reference implementation the secret is 4096 bit long and the random value is 256 bit.

checksum A checksum of all fields

Disclaimer

All security guarantees that this auth is trying to make will no longer hold, if somebody reverse engineers hanbots script encryption.

License

The author can be contacted under scarjit@aol.com (Include "S1mple Auth" in your email topic).
If you make changes to the code or this paper i would be happy to be informed, but this is not required.