

Table of Contents

<u>Sample Keepalive Script to Connect/Disconnect to an SSL Web Server Running with a Non-Encrypted Handshake</u>	1
<u>Document ID: 47386</u>	1
<u>Introduction</u>	1
<u>Prerequisites</u>	1
<u>Requirements</u>	1
<u>Components Used</u>	1
<u>Sample Script</u>	2
<u>Related Information</u>	3

Sample Keepalive Script to Connect/Disconnect to an SSL Web Server Running with a Non-Encrypted Handshake

Document ID: 47386

Introduction
Prerequisites
Requirements
Components Used
Sample Script
Related Information

Introduction

This script will connect to an Secure Socket Layer (SSL) Web server running SSL version 3.0. Connect to the server, do the non-encrypted handshake, and disconnect. This document also addresses implementation of scripted keepalives. This method of scripting is most closely related to functionality, which is present in Remote Access Server (RAS) dialup clients, terminal programs, and general scripting utilities. This feature utilizes WebNS's rich scripting language.

Complete with a simple socket Application Program Interface (API) (connect/disconnect/send/receive), a scripted keepalive will give the user the ability to tailor their own protocol, or write their own sequence of steps to provide a reliable ALIVE or DOWN state of a service. Without the scripted keepalive functionality, you are currently limited to FTP, HTTP, ICMP, and TCP. With scripted keepalives, however, you can remain on top of the current protocols by writing your own scripts. For example, you can develop a script specifically toned to connect to a POP3 server without requiring WebNS to build a keepalive type POP3. This feature allows customers to create their own custom keepalives to suit their specific requirements. Although this is a component of the Content Services Switch (CSS), custom scripts are not supported by the Cisco Technical Assistance Center (Cisco TAC).

The scripted keepalives below are not officially supported by TAC, but have been tested, and are available for use at your own discretion.

Prerequisites

Requirements

Familiarity with WebNS rich scripting language.

Components Used

The information in this document is based on the software and hardware versions:

- WebNS versions 3.x and higher
- CSS 11x00 Series

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Sample Script

The script below can be used to connect and disconnect to an SSL Web server running with a non-encrypted handshake.

```
!--- No echo.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- Filename: ap-kal-ssl-port
!--- Parameters: HostName
!
!--- Description:
!--- This script will connect to an SSL Web server running SSL
!--- version 3.0. Connect to the server, do the non-encrypted
!--- handshake, and disconnect.
!
!--- Parameters:
!--- SSL-IP: Address of the SSL Accelerator
!--- SSL-Port: Port for the SSL Accelerator
!
!--- Failure Upon:
!--- 1. Not establishing a connection with the host.
!--- 2. Not receiving a positive authentication.
!
!--- Author: KGS
!--- Last Tested: 9/27/01
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "==" "0"
  echo "Usage: ap-kal-ssl-port \'SSL-IP [SSL-Port]\'"
  exit script 1
endbranch

!--- Defines.

set HostName "${ARGS}[1]"
set SSL-Port "443"
if ${ARGS}[#] "GT" "1"
  set SSL-Port "${ARGS}[2]"
endbranch

!--- Connect to the remote host.

set EXIT_MSG "Connection Failure for ${HostName}:${SSL-Port}"
socket connect host ${HostName} port ${SSL-Port} tcp 2000

!--- Send the GET request for the Web page.

set EXIT_MSG "Send: Failed"
```

```
!--- Send over the hex for the fields:
!--- [Handshake: 0x16] [Version: 0x03 0x00] [Length: 0x00 0x59]
!--- [Client Hello: 0x01] [Length: 0x00 0x00 0x55] [Version: 0x03 0x00]
!--- [Random (32bit) #: 0x39 -> 0xff] [Session Length: 0x20]
!--- [Session ID (32bit): 0x3a -> 0x5d] [Cipher Length: 0x00 0x0e]
!--- [Cipher Suite: 0x00 -> 0x00 (Last Byte in stream)]

!--- Break the request into two send requests, as there is a 128 byte
!--- max on quoted text parameters.

socket send ${SOCKET} "1603000059010000550300392ae5530da35d89041b4beaa42891470e49
351c3bfef7631296139928dd7fff203a" raw

socket send ${SOCKET} "9a0ed92a4e4f66d75ecce24c3a361efc26ab86310c4b9e7271a1317d9
7635d000e0004ffe0000a00640062000300060100" raw

!--- Wait for a good status code.

set EXIT_MSG "Waitfor: Failed"

!--- Wait for a handshake message (0x16), paired with the version
!--- of SSL (0x03 0x00).

socket waitfor ${SOCKET} "160300" 2000 raw

!--- Wait for the specific server hello (0x02).

socket waitfor ${SOCKET} "02" 2000 raw

!--- Wait for the version again (it appears twice: 0x03 0x00).

socket waitfor ${SOCKET} "0300" 2000 raw

no set EXIT_MSG
socket disconnect ${SOCKET}

exit script 0
```

Related Information

- [CSS 11000 Series Content Services Switches Hardware Support](#)
- [CSS 11500 Series Content Services Switches Hardware Support](#)
- [Software Download for CSS11500](#)

All contents are Copyright © 1992–2006 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

Updated: Jan 20, 2006

Document ID: 47386
