WEBIDENTITY

THE UNIVERSAL TOKEN FOR INTERNET SECURITY

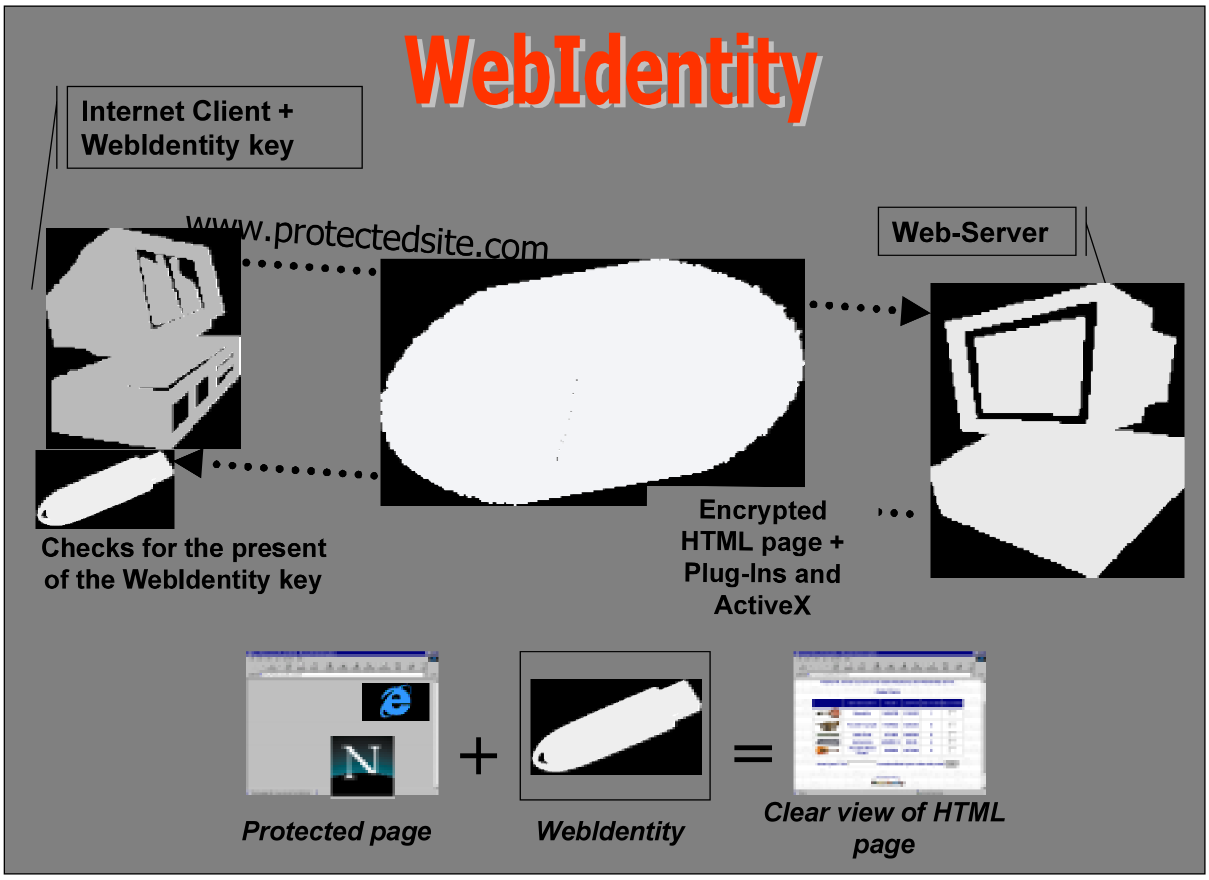
The Reader-less Smartcard

HOW WEBIDENTITY WORKS

Developed by

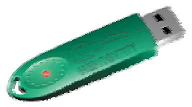
Eutron

Securing Internet & Software


WebIdentity
Internet Client + WebIdentity key
www.protectedsite.com
Web-Server
Checks for the present of the WebIdentity key
Encrypted HTML page + Plug-Ins and ActiveX
Protected page
+
WebIdentity
=
Clear view of HTML page

WEBIDENTITY

is a hardware device which simply connects to the USB port of a Personal Computer. When it is plugged in it permits secure and unambiguous identification of the user and the transaction of the data, which is encrypted, in all Web-based applications for Internet/Intranet/Extranet.



What is WebIdentity

WebIdentity is a hardware device that allows you to store your data inside it and encrypt it using the 3DES algorithm.

There are two versions of the device: USB and parallel port

Even though the two versions of the device have similar common features, which are based on different communications buses, they have different physical features:

USB
- Internal Read/Write memory of up to 10 Kbytes.
- Internal 3DES encryption chip.
- Powered directly through the USB bus.
- 16 byte identity LABEL which allows more than one device to be used simultaneously on the USB bus.
- Progressive serial number and client code inserted permanently into the device's memory during production. This ensures that every device is totally different from any other, making it impossible to clone them.

PARALLEL
- Internal Read/Write memory of up to 416 bytes
- Internal ASIC CHIP
- Powered directly from the parallel bus
- 16 byte identity LABEL which allows more than one device to be used simultaneously on the bus (the devices can be piggy-backed).
- Progressive serial number and client code inserted permanently into the device's memory during production. This ensures that every device is totally different from any other, making it impossible to clone them.

Eutron has developed a series of Client/Server libraries, which are supplied with the WebIdentity hardware device, to allow the use and integration with various WEB and software platforms. Thanks to the availability of the Netscape Plugin, Microsoft ActiveX and Java Class Library, programmers have a wide choice regarding both how to develop new WebIdentity Protected applications and integrating the use of WebIdentity into their own existing Internet applications.

The services provided by WebIdentity: client identification, data encryption, read/write to and from remote devices are based on the concurrent use of the client and server libraries. Using the methods exposed by the supplied class libraries, you can set up secure communications between the client and the server using the native communications protocol of the application you wish to protect: in the most common case of a Web application, you could use just the HTTP protocol.

For a better understanding of how WebIdentity works, let's take an imaginary case and analyze the steps and methodologies used to provide protection. Let's consider our Web application, which for the moment consists of a single source of dynamic information that we will call the Content Provider (CP). This source could be, for example, an ASP page, a JSP page, a Servlet, an application server or any other component that can carry out processing on the server side, maintain the notion of the state of the connection and interact with a client browser or through a Web

Server. It could also reply independently using the HTTPS protocol to any coupled address: ports belonging to an Internet server.

Main features of a Content Provider:

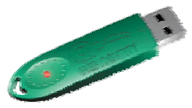
- Able to carry out processing.
- Able to maintain the state of the connection.
- Able to interact with the client
- Support for and capability to interact with ActiveX or Java classes

When a client connects, the CP will have to check if the current browser connection is the first one for the current session or if it is another one.

In the case of a user which has not yet been identified (first connection), the CP will have to send the client everything necessary in order to identify it. The "necessary" will be an HTML page which contains the following "objects":

- A tag object/embed in order to be able to use the ActiveX or Plugin within the browser.
  e.g.:
  ```
  <object classid="clsid:878A0D61-48D2-11D3-A75D-00A0245382DE" id="WIDrvCli"
  codebase="software/WICli.cab#version=3,0,1,2"  VIEWASTEXT>
    <embed type="application/x-wicli-plugin" name="WIDrvCli" hidden="true"
  width="1200" height="300">
    <noembed>
      You need a browser that supports Netscape plugins or ActiveX controls to view this
  page.
    </noembed>
  </object>
  ```

- A form with the fields needed to send information to the CP using the POST method of the HTTP protocol (you can also use other methods to send information to the server: GET, cookies, ...)
  e.g.:
  ```
  <form action="index.asp" method="POST" name="WebIdData" >
     <input TYPE="hidden" NAME="PIN">
     <input TYPE="hidden" NAME="KEYFOUND" value="true">
  </form>
  ```



- A JavaScript which reads the content of the key and sends it to the server using the form.
  e.g.:

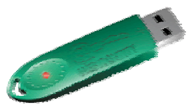
```
<script language="JavaScript">
<!--
   document.WIDrvCli.RndSessionString = "Tq88k9tMlb!LINloaa!Ox0"
   document.WIDrvCli.Label = "WEBIDENTITY"

   function SendPIN()
   {
         // Send PIN to server
         document.WebIdData.PIN.value = document.WIDrvCli.ReadPin();
               document.WebIdData.submit();
   }
//  -->
</script>
```

Analyzing the JavaScript, it becomes clear that there are some basic parameters which are indispensable for the protection to work:

- RndSessionString: a string which is used as part of the data encryption key to identify the client. This string is supplied by the CP and must be different for different sessions. This rules out any chance of re-using information which is travelling over the network by unauthorized persons.
- Label: required parameter which identifies the family of devices used by the current Internet application.
- PIN: it is read encrypted from the key using the ReadPin method. This is used to uniquely identify the client.

At this point it is worth while explaining the concept of the PIN and therefore understand how the recognition function of WebIdentity works.

1. **Initialization of the hardware device**: in order to uniquely represent a user, the device has to be linked to it during the initialization phase. At this moment, information that will enable the WebIdentity token to prove its "identity" to the server will be written to its memory. This is done by using an identifying string, user data and a service password. The password, which is kept secret, allows the server to identify the client and the client to let it be identified. At the same time an alphanumeric PIN number will be made available. The PIN number is needed to link a user to a token. It will therefore be linked in a database to the data regarding its owner.
2. **The creation by the client of an "identification pair "**: in order to be identified with certainty, the client uses the methodology, "being and proving". That is, a pair of strings are produced: IDENTITY (I affirm who I am) and PROOF (I prove that I am). This is done through the processing of data saved in the key during initialization. The processing is based on the joint use of the HD5 hashing algorithm and the 3DES encryption algorithm. So as to avoid unwanted re-use, this pair will be encrypted



    again, on a variable time basis, using the RndSessionString.

3. **Identification of the user by the server**: when the server knows the identification pair, by using the service password and the knowledge of the "Identity-Proof" algorithm, it can check that the client is really who it says it is. In this case the PIN created in phase 1 will be made available. The PIN, which is linked in the user database to the identity of its owner, will provide the CP with the real identity of the client.

Let's go back to our application. When the client gets the identification page, the browser will execute the JavaScript in it. An "identification pair" will be created by making a call to the ReadPin method of the Plugin or ActiveX. The pair will be sent to the CP further encrypted with session dependent code (RndSessionString) using the POST method of the HTTP protocol.
Through the use of the "identification pair", the CP can identify the client and reply as a result to current and future client requests for the duration of the session.

The scenario illustrated is the simplest form of implementation of WebIdentity protection. It has been left to the developer to extend at will the identification process, adding extra passwords, replicating the control of the token in multiple places etc.

Once the client has been identified, the WebIdentity libraries provide the developer with the ability to encrypt the flow of data between the client and the server as well as the ability to read and write to and from the memory of the remote hardware device. These operations are always based on the transmission to the client by the Content Provider of specific HTML pages containing JavaScript which, through the use of the methods provided in the Plugin and ActiveX, are able to interact with the token and carry out encryption.

For further information on the product and the use of the libraries, please refer to the SDK supplied with the product.