

DNABolt System Synopsis

Introduction.

Humans have DNA that are universally unique and are used to identify a certain individual.

Computers like humans can have DNAs that are universally unique and can be used to identify a single computer.

DNABolt is an authentication system based on a computer DNA.

The computer DNA.

DNABolt creates a computer DNA based on a set of hardware characteristics such as number of processors, processor model and speed, memory size and banks, video card model, bios information, hard disk information, network adapter cards etc.

A total of 10 different hardware characteristics are used to form a computer DNA. To preserve user privacy, each of the 10 DNA components are acquired and converted into a hash.

DNABolt components.

The DNABolt authentication system has three main components: a DNA database, an authentication server that we call the Vault, and a client code.

Computer DNAs are kept in a database. DNAs stored in the Vault are associated to a UserID. One UserID can have several associated DNAs. The Vault can hold DNAs belonging to different organizations in different groups called Realms.

The DNABolt Vault is responsible for DNA authentication and storage. It receives a Realm/UserID and DNA from the DNABolt client and compares it against the DNAs associated to that Realm/UserID stored in the database. If a matching DNA is found it returns a positive authentication else authentication fails.

The authentication server has provisions to accommodate hardware upgrades. It has a set of rules that allows for changes on a subset of the 10 fingerprint hashes that identifies a

machine. Optionally the system can be configured to require 100% match to authenticate a DNA.

The DNABolt authentication server was developed in Java as a set of Java Servlets and runs under a servlet container such as Apache Tomcat and BEA WebLogic. It is distributed as a set of “.war” files.

The DNABolt client is responsible for acquiring a machines DNA and its transmission to the authentication server. The client code must be installed on each machine that will use the DNABolt authentication system.

Currently, DNABolt clients are available for Microsoft Windows (95, 98, Me, NT4, XP, Vista/7 and 2003/2008 server), Windows Mobile (2003 SE and 2005) and Apple MacOSX (10.3.9 and 10.4 and later Intel and PPC) and Linux. It can be used by web based applications or native applications.

The following installation options for the DNABolt client are available:

- Windows

1. Firefox Plug-in.Extension
2. ActiveX control installed by IE.
3. ActiveX control installed by an InstallShield based MSI installer.

- MacOSX

1. Internet-enabled disk image.

- Linux

1. Firefox Plug-in Extension

The DNABolt clients were developed in C/C++ with a small portion of the code written in assembler.

DNA security.

DNA data transmitted over the net to the authentication server is protected against playback. Protection is achieved by data encryption using a different encryption key on each transmission. DNA components are also shuffled prior to encryption.

Although it is impossible to fully protect any software from being hacked, the DNABolt client code is protected against various forms of hacking.

Data transmission.

DNABolt client sends data to the authentication server using HTTP or HTTPS. DNA data is encrypted and transmitted in binary format.

Authentication server protection.

The authentication server can be protected against unauthorized access by a combination of the following mechanisms:

1. Client IP address access control. A set of single IP addresses and/or a set of IP address ranges can be configured in an IP address permission list. Only clients whose IP addresses are included in the list are granted Vault access.
2. SSL client certificate access control. SSL works by providing three security services, each of which use public-key encryption techniques.
 - a. After an initial handshake, all transmissions are encrypted to prevent such transmissions from being eavesdropped
 - b. The SSL protocol ensures that messages between the sender and receiver have not been tampered with in any way between the two points thus maintaining session integrity.
 - c. Mutual authentication. This is the process by which the client and server can convince each other of its identity through the exchange of X.509 certificates.

An X 509 certificate has several pieces of information such as:

- a. Issued by
- b. Validity
- c. Public key
- d. Issued to (distinguished name - DN)

The Vault certificate access control can perform the following client certificate validations:

- a. General certificate validation (validity and CA signature)
- b. Any DN field such as Common Name (CN), Organization(O), Organization Unit(OU) can be matched against a set of specified names.
- c. Reverse DNS on the client IP address can be matched against the client certificate Common Name (CN)

DNABolt authentication flow.

Web browser application.

1. End user opens a web page on a HTTP server and provides his/her userid and password.
2. HTTP server authenticates the user using its own authentication mechanism.
3. HTTP server opens a session with the DNABolt Vault and receives a sessionID and a seed number.
4. HTTP server responds to the end user with a page that has calls to invoke the DNABolt client The call includes the following parameters: a. URL where DNA data should be sent to b. Realm and UserID c. The randomly generated seed number received on step 3 d. The sessionID received on step 3
5. DNABolt client creates the DNA data and sends it to the authentication server
6. HTTP server queries the authentication server to authenticate the DNA. The sessionID obtained in 3 is passed to the authentication server as a parameter.
7. If access is granted, user receives a web page to proceed with the application

The authentication flow can be seen on the page following:
(Figure 1 – DNA authentication flow)

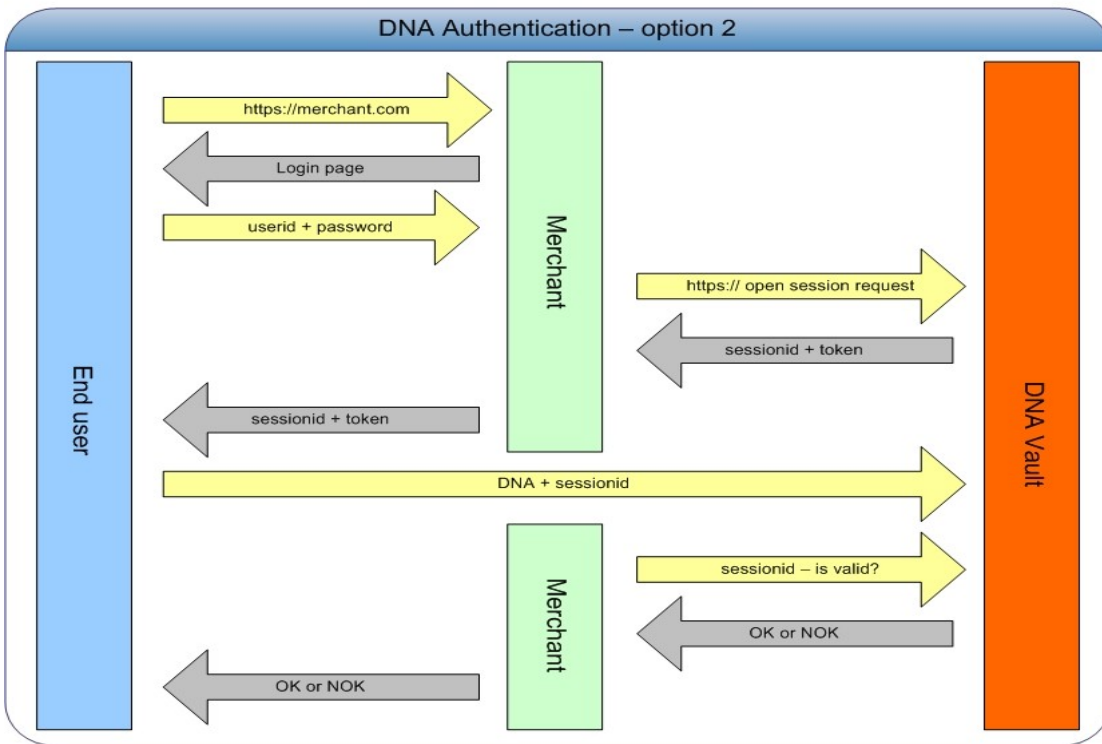
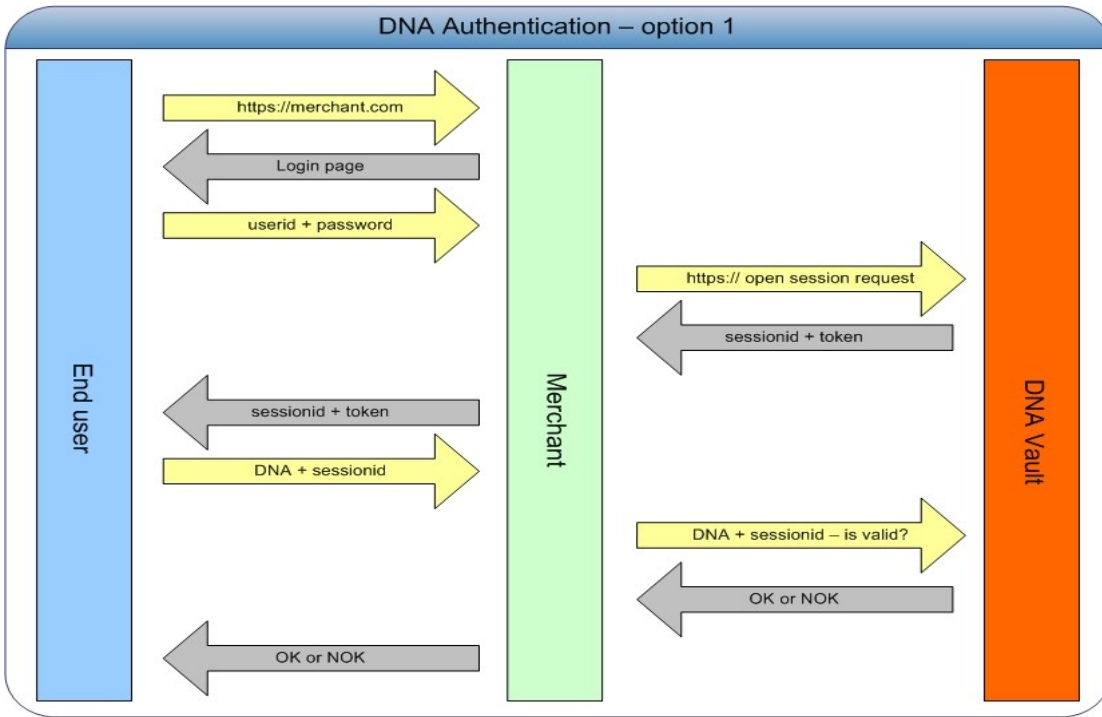


Figure 1 – DNA authentication flow

A JavaScript example of the authentication process

```
<html>
<head>

<title>.: DNABolt :. Your Digital DNA</title>
<script language="JavaScript">

    // Global XMLHttpRequest variable
    var content = '';

    //-----
    // This function loads the specified URL and inserts the result // (or an error
message) into the "content" variable.
    // It uses a synchronous XMLHttpRequest to do so.
    // Returns false if an error occurs loading the URL.
    //-----
    function loadURL(url)
    {
        var request = null;
        var retvalue = false;
        content = '';

        // Create an XMLHttpRequest object or ActiveX control
        if (window.XMLHttpRequest) {
            request = new XMLHttpRequest();
        } else if (window.ActiveXObject) {
            request = new ActiveXObject("Microsoft.XMLHTTP");
        }

        // If XMLHttpRequest is supported
        if (request) {

            // Set up synchronous request
            request.open("GET", url, false);

            // Send synchronous request
            request.send(null);

            // Check the status
            if (request.status == 200) {
                // Success
                content = request.responseText;
                retvalue = true;
            } else {
                // Error
                content = 'Error: ' + request.status + ' ' + request.statusText;
                alert (content)
            }
        }

        return retvalue;
    }

    //-----
    //-----
    //-----
    //-----
    function Authenticate() {

        var f = document.form;
        if (f.realm.value == "") {
            alert('Fill in your realm name');
            f.realm.focus();
            return false;
        }
        if (f.user.value == "") {
            alert('Fill in your user name');
            f.user.focus();
            return false;
        }

        // open a session with DNAvault for the realm specified by the user
        if (loadURL("/dnavault/dispatch?op=open&realm="+f.realm.value)) {

            // show the vault response
```

```

alert ("open session result:\n"+content)

var sessionid = '';
var token = '';

// parse the results to obtain sessionid and token
var words=content.split("\n")
for (i=0; i<words.length; i++) {
    if (words[i].indexOf("result=") == 0) {
        if (words[i].charAt(7) != '0') {
            return false;
        }
    } else if (words[i].indexOf("id=") == 0) {
        sessionid = words[i].substr(3,words[i].length - 4);
    } else if (words[i].indexOf("token=") == 0) {
        token = words[i].substr(6,words[i].length - 7);
    }
}

var baseurl = location.protocol+"//"+location.hostname+": "+location.port
// invoke the plug-in to calculate the DNA and send it to the DNAVault
dnacli.SendFPEX(baseurl+"/dnavault/dispatch", token.toString(),
f.realm.value, f.user.value, "", sessionid.toString());
if (dnacli.LastErrorCode != 0) {
    alert("DNA plugin error: "+ dnacli.LastErrorMsg)
    return false;
}

var opercode = "validate";
if (f.radiob[1].checked) {
    opercode = "insert";
}

// call DNAVault to complete the desired operation (validate or insert)
if (loadURL("/dnavault/dispatch?op="+opercode+"&id="+sessionid)) {

    // show the vault response
    alert (opercode+" result:\n"+content)

    // terminate the session
    if (loadURL("/dnavault/dispatch?op=close&id="+sessionid)) {
        // show the vault response
        alert ("close session result:\n"+content)
    }

} else {
    alert(opercode+" failed !")
}

} else {
    alert("Open session failed!")
}
}
</script>
</head>

```

```

<body>
<script language="JavaScript">
<!--hide script from old browsers
var myUserAgent = navigator.userAgent.toLowerCase();
if (myUserAgent.indexOf("msie") > 0) {
    document.write("<object id=\"dnacli\" classid=\"clsid:7820FDBF-8009-4191-8B02-63409964A5C4\"
codebase=\"https://DNABolt.com:8443/dnaplugin/axdnacli.cab#version=1,2,4,0\"
width=\"0\" height=\"0\"></object>");
} else {
    // works with Netscape and Safari
    document.write("<embed id=\"dnacli\" type=\"application/x-dnacli-plugin\"
hidden=\"true\"></embed>");
}
// end hiding -->
</script>

```


This sample shows a typical authentication/machine enrollment process.

The required steps are:

-
- Open a session with the vault
- Invoke the plug-in to calculate the DNA and send it to the vault
- Call the vault to validate the DNA or to enroll the machine
- Close the session

Note: usually steps 1, 3 and 4 take place between the web server and the DNABolt vault

step 2 is executed at the end user machine.


```
<form name="form" onSubmit="return Authenticate()">  
  Realm:&nbsp;<input tabindex="1" type="text" name="realm" value=""  
class="input"><br><br>  
  User:&nbsp;&nbsp;&nbsp;&nbsp;<input tabindex="2" type="text" name="user" value=""  
class="input"><br><br>  
  <input tabindex="3" type="radio" name="radiob" checked>Authenticate DNA<br>  
  <input tabindex="4" type="radio" name="radiob" >Enroll this machine<br><br>  
  <input tabindex="5" type="submit" value="Submit">  
</form>
```

</body>

</html>