THREAT MODELLING FOR ACTIVE DIRECTORY

David Chadwick
ISI, University of Salford, Salford, M5 4WT, England.

Abstract: This paper analyses the security threats that can arise against an Active Directory server when it is included in a Web application. The approach is based on the STRIDE classification methodology. The paper also provides outline descriptions of countermeasures that can be deployed to protect against the different threats and vulnerabilities identified here.

Key words: security, LDAP, active directory, threats, vulnerabilities

1. INTRODUCTION

Active Directory (AD) is Microsoft's LDAP product offering, first introduced with Windows 2000 servers. Whilst being reasonably conformant to many of the LDAP set of standards e.g. [1,2,3], nevertheless it is non-conformant in some aspects. For example, it does not support some standardized features, such as multi-valued relative distinguished names (RDNs) or country based naming, but it does support many proprietary features, such as a tight coupling with the operating system and Microsoft's DNS server. It has also replaced several standardized features with its own proprietary ones. For these reasons customers using Microsoft's operating systems are well advised not to try to replace Active Directory with an alternative more standards' conformant LDAP product such as OpenLDAP.

Active Directory is a core service holding user and server account details and security information. For example, Windows authentication uses credentials stored in the Active Directory. Active Directory is therefore fundamental to the correct operation of a Microsoft domain. For this reason most Microsoft based Web applications will need to access Active Directory either directly or indirectly at some point during their business processing, often during the authentication and/or authorization phases, but also at different stages of the business process.

Access to Active Directory therefore needs to be well controlled and protected, otherwise an attacker could severely impair the correct functioning of both the Web application and the back office by successfully launching an attack on Active Directory. Web application builders need to understand the vulnerabilities of Active Directory and the threats that can potentially exploit these vulnerabilities. In common with the other papers in this series [4, 5, 6, 7] we use the STRIDE approach [8] to categorize the most frequent or damaging threats that can arise against Active Directory when deployed as part of the generic model for Web applications described in [9]. Finally we describe the countermeasures that can be used to prevent these threats or to mitigate against the damages subsequent to a successful attack.

2. ASSUMPTIONS

The guidelines discussed in this paper will be effective only if the Active Directory is properly installed, configured and patched with the latest updates and service packs as released by Microsoft.

Correct configuration requires that that the Access Control Lists (ACLs), that are used to control access to objects in the Active Directory, are set up to give minimum privileges to the users (and to the Application Server acting as a user or a proxy for the users).

The assumption is that the Application Server will communicate with the Active Directory server by RPC messages generated by the Active Directory Service Interfaces (ADSI), using one of the various scripting or programming languages that it supports e.g. C++, Visual Basic or Java.

It is assumed that there is only a limited amount of trust between the Active Directory Server and the Application Server, and between the Application Server and the Web Server. By this, we mean that the Active Directory Server will not let the Application Server have unrestricted access to its resources, but will impose some controls on what the Application Server can do. For example, the AD server may have an administrative limit on the number of LDAP entries that can be returned to any ADSI request; the AD server may have controls on the complexity and number of filter items that can be included in a Search filter; and the AD server will have properly configured access controls that limit which directory entries and which operations the Application Server (and its users) are allowed to access. Likewise, the Application Server will have some controls on the messages originating from the Web Server and will validate and restrict their contents. From a security perspective, the less trust that there is between the AD server and the Application Server, and between the latter and the Web Server, the better, as more controls will be imposed by the AD server on what the Application Server is allowed to do, and by the Application Server on what the Web Server can do. For example, at one extreme the AD server may forbid any modification operations to originate from the Application Server. The more trust that there is, the more careful the application developer will need to be to ensure that this trust is not abused by an application server that may become compromised, or that is just badly programmed.

The final assumption is that there is no (or very little) trust between the Web Server and the client, or between the Web Server and the network over which the client's http messages are transported. Thus eavesdropping of messages on the network is possible, and in extreme cases, message modifications. Furthermore, the Web Server must expect the client to try to circumvent whatever client controls are placed on the messages that it sends. Consequently the Web Server and all subsequent servers that receive client messages, for example SOAP messages that are relayed through the Web Server, must be designed to protect against threats emanating from modified or badly formed client messages, by rigorously validating their contents.

3. SECURITY REQUIREMENTS

We can look at security requirements from two perspectives: the security requirements placed on the design of the web application because it has little or no trust in the client and the external network, and the security requirements placed on the Active Directory because it only has limited trust in the web application.

The security requirements placed on the web application i.e. the web server and the Application Server, partially depend upon the type of application that is being built. At one extreme, we may have an application that is only retrieving public information from the AD server. At the other extreme we may have an application that is accessing highly confidential directory information and writing to the AD server by adding, modifying and deleting objects in the Directory Information Tree (DIT).

In the former case the web application may have very few security requirements placed on it, and may allow unauthenticated user access over unsecured http links. Example applications might be: one that accesses the contact information of people in the marketing and sales department, or one that retrieves certificate revocation lists (CRLs) for a PKI application. The main security function of the web application will be to validate the contents of the client requests (see below) and ensure that only a predefined limited set of Search requests, and no modification requests, are sent to the Active Directory.

In the latter case the web application will have very strict security requirements placed on it. An example application might be one that supports single sign on (SSO) and user authorisation by checking user credentials in and retrieving their privileges from the AD server, whilst simultaneously supporting dynamic provisioning and management of user rights. In such cases, the web application will demand strong authentication of the user to prevent masquerade, and will require all messages to be carried across encrypted links to protect against eavesdropping and message modification. The web application should never request nor accept user passwords passed in the clear from the client. This will facilitate password capture over an insecure network. The application should always require passwords to be sent over an encrypted link e.g. using SSL or IPsec, or use HMAC hashing which creates one-time passwords. When using SSL, the web server should check that the SSL cipher suite that has been negotiated with the client is a minimum of 128 bit encryption, and that it has not been negotiated down to plain text (no encryption) or weak encryption. The same holds true for the Application Server if it is using an insecure link to communicate with the Web Server. In addition rigorous checking and validation of all client provided fields and requests should take place as described next.

Preferably, and whenever possible, limit the choices that are available to the client by having picking lists of predefined values so that the client cannot create its own values (this is very important for attribute type names, matching rules, the distinguished names of subtree roots, the name of the AD server and its connection details, although the latter of these will usually be pre-configured into the Application Server and the client will not have any control over them). For fields where the client must usually have complete freedom of choice over the input values, for example, attribute values for Search filters, then the Application Server should perform rigorous validation of these values. Firstly determine the maximum length of each field and check that it has not been exceeded by the client. Reject client operations in which fields are too long. Secondly, treat each field as a literal and make sure that it is encoded as such, for example by enclosing the user's input in quotation marks. Consider the following: say that the client interface had separate input fields for attribute types and values when creating a Search filter. The code might put them together to create a filter such as (<user type>=<user value>). An attacker might place the following in the Attribute Type field

&(objectCategory=person)(!salary>=10000)(commonName

and the code would then create the following valid complex filter

(&(objectCategory=person)(!salary>=10000)(commonName=<user value>)

thereby allowing the attacker to create whatever Search filters they want to. Input field validation and checking is thus extremely important.

Because the Active Directory only has limited trust in the web application, the security requirements placed on the Active Directory and on the design of the web application are common, regardless of the type of application that is being built. Firstly the Active Directory should be configured so that the Application Server has no (or very limited) access privileges to data in the Active Directory. This will help to protect against elevation of privileges, whereby a user gains the access privileges of the application rather than his/her own. Secondly the Active Directory should limit the types of request from and the volume of data returned to the Application Server. Finally, the Application Server should Bind to the Active Directory using the client's user context rather than its own.

When an application Binds to an object in the Active Directory, the access privileges that the application has to that object are based on the user context specified during the Bind operation. For the ADSI binding functions and methods (IADsOpenDSObject::OpenDSObject, ADsOpenObject, ADsGetObject, GetObject) an application can implicitly use the credentials of the caller, or explicitly specify the credentials of a user account, or use an unauthenticated user context (Guest). The Application Server should never Bind to the AD server using a stronger form of authentication than that used by the client, nor should it use a user account that has higher privileges than the client's (for example, the LocalSystem account on a domain controller has complete access to Active Directory whereas a typical user has only limited access to some of the objects in the directory). Ideally, the Application Server should either use the credentials provided by the user, and validate them by passing them to the AD server either implicitly or explicitly, or should discard the user credentials altogether and use the Guest context. In the former case the Application Server is acting as a proxy for the client and will thus only have the same access rights to the directory data as if the client were binding directly. In the latter case, the Application Server will only gain minimum/public access rights to the directory data. For example, an application policy might say that local users who access the Active Directory when they are at a remote site should only have Guest access to public data in the directory, in which case their credentials would be discarded by the Application Server when they contact it from a remote location.

4. ACTIVE DIRECTORY THREATS AND COUNTERMEASURES

4.1 Spoofing

Spoofing can take one of two forms. Either an attacker attempts to spoof a user or an attacker attempts to spoof the Active Directory. In the former case the attacker captures or guesses a user's credentials and then masquerades as the user when accessing the Active Directory. In the latter case the attacker tricks a client into believing that information came from the Active Directory when it did not, or tricks the client into sending confidential data to it that should have been sent to the Active Directory. Spoofing results from vulnerabilities in the client or in the network. Spoofing the directory could be achieved by social engineering (e.g. sending a wrong URL to users), misdirecting operations or modifying data in transit. The use of SSL links will counteract the latter two, and user education will help to protect against social engineering, although this is notoriously difficult to fully protect against.

Spoofing a user can be aided by vulnerabilities in the network, vulnerabilities in the Active Directory Information Base and vulnerabilities in the Application Server. An attacker can sniff the network to obtain user account names and passwords, or access the Active Directory to retrieve valid user account names and then find the password by either a dictionary attack or modifying the password attribute in the directory. Since the Active Directory is often designed to return user account names, it may be difficult to stop attackers from gaining this information, but if clients generally don't need to know user account names, then these should not be returned to the client interface. The use of encrypted connections such as SSL or IPsec will stop network sniffing, as will the use of HMAC [10] or Kerberos authentication. Dictionary attacks can be prevented by having the Application Server count the number of failed login attempts per user account name, writing them to audit trails, and then disabling the account when a threshold number is exceeded. Modification of password attributes can be prevented by the Application Server not providing a modification capability to the client, but if this is essential, then the server should carefully validate all modification operations and trap ones that try to modify the password attribute.

4.2 Tampering

With this threat, an attacker tries to modify directory data either in transit to the client, or whilst it is stored in the AD server. This can be due to vulnerabilities in the network, vulnerabilities in the Active Directory Information Base or vulnerabilities in the Application Server. Threats to the AD server can arise from masquerade, poorly configured access controls and the injection of modification operations via the Application Server. Countermeasures include protecting data in transit by using either SSL or IPsec. Masquerade has been dealt with in Section 4.1. The Application Server should be configured to reject all Modification operations, or if this is not possible, to very carefully validate all user input fields and to reject operations with “invalid” arguments. The Application Server should Bind to the AD server using the user provided credentials so that the user does not inherit the possibly higher privileges of the Application Server process.

4.3 Repudiation

Repudiation is when users deny that they have performed specific actions or transactions. Keeping adequate audit trails will provide evidence of who did what and will help to counteract this type of threat. Auditing should be performed by both the AD server and the Application Server, and in this way insider attacks directed straight to the AD server will be more easy to identify. Requiring relatively strong client authentication will minimize the chances that an attacker can perform actions on behalf of a client which will subsequently be repudiated.

4.4 Information Disclosure

Information disclosure occurs when a user gains read access to information that (s)he is not supposed to have access to. This can be due to vulnerabilities in the network, vulnerabilities in the Active Directory or vulnerabilities in the Application Server. Vulnerabilities in the AD software, other than those caused by badly configured access controls, are outside the scope of this document. An attacker may sniff the network, masquerade as another user, or generate valid or invalid search or modify requests. Network sniffing and masquerade have already been dealt with in section 4.1. Generating invalid search or modify requests may return useful error diagnostic messages, which can provide the attacker with valuable information. The countermeasure to this is for the Application Server to scrub useful information from error messages and to return bland generic error messages to the client, whilst writing the full error message to its audit trail.

The Application Server should exert control over the Search requests that clients can perform. In general only specific limited Searches should be allowed by clients, otherwise attackers may generate very broad searches that trawl the entire directory. All user input should be validated, and only a fixed subset of ADSI arguments should to be allowed. Searches with "invalid" arguments should be rejected. However, a determined attacker may even circumvent this by generating multiple valid Search requests that only return snippets of information each time. If this is done a sufficient number of times, the sum total of information gained by the attacker may be more than the application designers ever intended to be revealed. For example, retrieving details of individual users in each Search request, may enable an attacker to retrieve details of the entire organizational workforce. Such attacks are very difficult to stop. Even building an audit trial and refusing access after a set number of searches might not stop the problem if the attacker has access to multiple user accounts.

Similarly, the Application Server should exert tight control over Modification operations. Ideally, it should refuse to allow any Modification operation through the interface, or if this is not possible, it should ensure that only authenticated users can perform modification operations, whilst simultaneously very carefully validating all user input fields and rejecting those with "invalid" arguments. For example, an attacker may try to modify the heuristic status of attributes, by setting bit 1 (which will make the attribute visible to unauthenticated users) or unsetting bit 3 (which removes operational attribute status).

4.5 Denial of Service

In a denial of service attack, the attacker denies access to the AD server for normal users. This can be aided by vulnerabilities in the Active Directory server and vulnerabilities in the Application Server. Denial of Service attacks are typically very hard to protect against.

The attacker may try to crash the AD server, or more likely, consume excessive resources. The easiest way to consume excessive resources is to launch CPU or network intensive Search operations. The former can be started by creating Searches with inefficient and/or complex filters, or ones containing multiple ambiguous name resolution elements (i.e. those where the attribute type is set to anr) [11]. Network intensive Searchers are designed to return lots of entries – the entire AD contents if possible.

Countermeasures to the above are as follows. The Application Server should validate all filters input by the user and only allow a predefined subset of filters to get through. In addition, the AD server should be configured to reject complex filters, and to only return a pre-defined maximum number of entries for any Search request.

An attacker may try to open up multiple connections to the Application

Server and/or AD server, preferably using SSL which consumes more resources. Countermeasures include timing out inactive sessions, keeping a record of the usernames of each active session and only allowing a fixed number of sessions per user at any one time.

More sophisticated attacks, which would normally require administrator level privileges, include: switching off indexing which kills the performance of most search operations; starting replication between AD servers which again kills performance; or updating the schema which might actually crash the AD server. Careful validation of the allowed modification operations by the Application Server should trap operations such as these.

4.6 Elevation of Privileges

Elevation of privileges can occur when an attacker either masquerades as a user with higher privileges than his own, or modifies data in the directory, for example, by adding a user to a group, or modifying ACLs in directory objects. Masquerade has already been described in Section 4.1. Illegal modification of directory data can be prohibited by disallowing any Modification operation to originate from the Application Server, or if this is not possible, by very carefully validating all user input fields and rejecting operations with “invalid” arguments. Correctly configured Access Control Lists in the AD server, and Binding with minimum privileges are also essential.

5. CONCLUSIONS

Whilst many different vulnerabilities and threats exist, they can nearly all be protected against by a few common countermeasures:

- Encrypt and authenticate messages that pass over insecure networks by using either SSL or IPsec
- Always have the Application Server Bind to the Active Directory using the same or lower privileges than those possessed by the client
- Ensure that the Access Control Lists in the AD server are correctly configured to give minimum privileges to clients.
- Severely limit the number and scope of directory operations that the Application Server sends to the AD server on behalf of the client. Always try to restrict the range of parameters that can be set or chosen by the client, and validate all user input fields for their content. If possible, ensure that no Modification operations are ever sent.
- Restrict the error diagnostic messages that are returned to the client.

With these countermeasures in place, it will significantly reduce the risk that an attacker will be able to launch a successful STRIDE attack against an Active Directory server.

6. ACKNOWLEDGEMENTS

The work reported in this paper was developed as part of the *Designing Secure Applications* (DeSecA) project, funded by Microsoft. Partners within this project were the Universita' degli Studi di Milano, the Technical University of Ilmenau, the University of Salford, and the COSIC and DistriNet research groups of the Katholieke Universiteit Leuven.

7. REFERENCES

[1] M. Wahl, T. Howes, and S. Kille. Lightweight Directory Access Protocol (v3), RFC 2251, December 1997

[2] Wahl, M., Coulbeck, A., Howes, T., Kille, S. "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions". RFC 2252. December 1997.

[3] Kille, S et.al. "Using Domains in LDAP/X.500 Distinguished Names", RFC 2247, Jan 1998

[4] E. Bertino, D. Bruschi, S. Franzoni, I. Nai-Fovino, and S. Valtolina. Threat modelling for SQL Servers. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp189-201

[5] R. Grimm and H. Eichstädt. Threat modelling for ASP.NET – Designing Secure Applications. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp175-187

[6] D. De Cock, K. Wouters, D. Schellekens, D. Singelee, and B. Preneel. Threat modelling for security tokens in web applications. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp 213-223

[7] L. Desmet, B. Jacobs, F. Piessens, and W. Joosen. Threat modelling for web services based web applications. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp161-174

[8] MSDN Library - Improving web application security: Threats and Countermeasures http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/ThreatCounter.asp, 2003

[9] L. Desmet, B. Jacobs, F. Piessens, and W. Joosen. A generic architecture for web applications to support threat analysis of infrastructural components, Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp155-160

[10] Wahl, M., Alverstrand, H., Hodges, J., Morgan, R. "Authentication Methods for LDAP", RFC 2829, May 2000

[11] MSDN Library - Creating More Efficient Microsoft Active Directory-Enabled Applications. http://msdn.microsoft.com/library/en-us/dnactdir/html/efficientadapps.asp?frame=true