iSEC Partners, Inc. – Brad Hill, Principal Consultant

Weaknesses and Best Practices of Public Key Kerberos with Smart Cards

Kerberos V with smart card logon is the "gold standard" of network authentication for Windows Active Directory networks and interoperating systems. Smart cards and Public Key Kerberos are already widely deployed by governments and large enterprises to protect their networks and critical infrastructure, and decreasing prices and an increase in threats against password-based authentication are making these technologies attractive to even small and medium sized businesses.

Unfortunately, network architects, administrators and even experienced security professionals tend to place a sort of blind faith in smart cards and Kerberos. The low level details of the protocols and configuration options are little known. Myths and misunderstandings about the limits of these technologies are the norm.

This whitepaper will:

- Give a brief introduction to Kerberos and smart cards
- Dispel some common myths about smart cards
- Explore the certificate validation practices of common PKINIT implementations
- Discuss a practical elevation of privilege exploit possible in common configurations of Windows KDCs
- Provide step-by-step advice to network architects and administrators for securing their smart card deployments

While smart cards have definite advantages over passwords, they should be deployed with a realistic understanding of the actual protections they provide. Installations should take advantage of the latest configuration and hardening options available, administrators should continue to audit and work to eliminate outdated protocols like NTLM from their networks, and privileged users should always exercise caution when authenticating to low-integrity workstations, even with a smart card.

A Brief Introduction to Kerberos and Smart Cards

The Kerberos protocol is a network authentication protocol which allows nodes on a network to prove their identity to one another and exchange cryptographic key material in a secure manner. Kerberos uses symmetric cryptography primitives and a trusted third party, the Key Distribution Center (KDC), to provide mutual authentication, replay protection, confidentiality and integrity of communications between clients and servers.

For the purposes of this paper, a smart card is any small form-factor hardware device with an embedded integrated circuit providing cryptographic services. A smart card has the ability to store X.509 certificates and private keys in a tamper-resistant package and can do cryptographic operations using RSA and/or ECC algorithms. A card will typically require the user to enter a PIN code or biometrically authenticate in order to unlock cryptographic operations requiring use of a private key. Private keys can be imported onto a card, but are usually generated by an onboard cryptographic service provider. A private key cannot be exported or read from the card.

Smart cards offer many important advantages over passwords. They provide two-factor authentication as a user must both have possession of the physical card and know the PIN code to use it. A lost card can be deactivated and, until such time, is useless without the PIN. With proper policy, smart cards can prevent concurrent account usage. Unlike a password, a smart card can guarantee that authentication secrets are cryptographically strong and cannot be written down, lost, shared, "phished" or re-used in an insecure system. More generally, asymmetric cryptography can help eliminate the need for attackable, locally stored authenticators and server-side password databases.

Extensions allowing public key cryptography to be used for initial authentication (PKINIT) let smart cards replace user passwords in Kerberos, and the combination has proven compelling. Users can authenticate to and interoperate with the entire universe of existing Kerberos-capable services, across realms and platforms, and enterprises can roll out smart cards incrementally and transparently at a low deployment cost. Smart cards are a first class credential type for Microsoft Windows and Active Directory installations, allowing large enterprise networks to be operated entirely without passwords, and nearly all other major operating systems provide at least basic support for Public Key Kerberos with Smart Cards.

Smart card-initiated Kerberos logon has become the premier standard for network authentication at large enterprises in both the public and private sectors and is making rapid inroads into small and medium sized organizations as costs decrease and attacks targeting password-based authenticators increase. The ease of integration and promised benefits are so great that the combination of smart cards and Kerberos has come to be uncritically regard-

ed as an intranet authentication silver bullet by most network architects, administrators and security professionals.

While the combination of Kerberos and smart cards is conceptually simple, the full details of PKINIT's interaction with Active Directory, enterprise PKI, and client software packages are extremely complex and understood in their totality by only a handful of people.

Obtaining the full picture of how Kerberos authentication with smart cards works on a modern Windows Active Directory network requires synthesizing information contained in half a dozen RFCs, several IETF working drafts, a dozen or more documents from the Microsoft Communications Protocol Program (MCPP) and Work Group Server Protocol Program (WSPP), and a similarly large number of Microsoft Knowledge Base articles, whitepapers and guides. While a meticulous, expert reading of all of these documents in context reveals that the risks this paper will discuss have been known and acknowledged since at least 2006, their full implications and the possibilities for practical exploitation and elevation of privilege have never been frankly and openly discussed. As a result, all Active Directory networks using smart card login are vulnerable in their default configuration and few if any administrators are aware of the risks or the proper procedures and practices to mitigate them.

Before discussing the attacks in detail, it is useful to first dispel some common myths about smart cards and show some of their practical limitations.

Dispelling Common Myths about Smart Cards

Smart cards are commonly thought of as an inherently safe form of credential. Sophisticated administrators of Active Directory networks are aware of the risks of password based authentication and password-based protocols like NTLM, and deploy smart cards with the goal of reducing or eliminating such risks. However, they often assume too much about what guarantees a smart card can provide.

The two most common misunderstandings about smart card credentials are that:

- Deploying smart card only-logins will guarantee Kerberos is used for Integrated Windows Authentication, eliminating the risks associated with the older NTLM protocol.
- Private keys cannot be exported, so it is safe (or at least, safer) to authenticate to a potentially compromised workstation with a smart card because no long-lived credential material exists that can be used by an attacker after the card is removed.

Smart Cards and NTLM

The myth that use of smart cards prevents use of NTLM probably arises from the fact that the NTLM protocol is password-based, and smart card users do not enter (or may not have) a standard password. Although it is true that the initial Active Directory domain logon with a smart card is guaranteed to use Kerberos and that asymmetric credentials cannot be used for NTLM, it is not true that users who authenticate with a smart card will never use NTLM to access network resources.

Until the release of Windows 7 and Windows Server 2008 R2, it was not possible to disable the use of NTLM on a Windows network. The inability to perform NTLM authentication would make many common tasks in an Active Directory network impossible, so the system has to provide a way to use NTLM, even for smart card users. Most password-based authentication protocols in Windows are not based directly on the password, but on a hash of the password. There are two versions of this hash, the LM and the NTLM OWF (one-way function). The Active Directory stores a copy of these hashes[1] and uses it to verify standard Kerberos and NTLM authentication traffic. Smart card only users do not have a password, but they still have an OWF. Instead of being based on a password it is simply a randomly generated 128 bit value. When a user authenticates using PKINIT, to support NTLM authentication, the user's OWF is sent to the client in the privilege attribute certificate (PAC) PAC_CREDENTIAL_INFO buffer[2], part of the authorization data extensions in Microsoft's implementation of Kerberos.[3]

[1] Some systems are configured to only store the stronger NTLM OWF.
[2] See [MS-PKCA] http://msdn.microsoft.com/en-us/library/cc238455(PROT.13).aspx,

After initial logon, both a Kerberos TGT and the OWF are available, and the behavior of Windows Integrated Authentication using SPNEGO[4] is no different for smart card-originated logons than for password-originated logons. Password and smart card based logons are subject to identical risks from the use of the NTLM protocol with regard to credential forwarding and lack of server authentication.

The only advantage offered by smart cards is that the random OWF generated for a smart card only login will be significantly more resistant to dictionary or rainbow table attacks against NTLM protocol messages, especially NTLMv1, than will a user-selected password.

[MS-PAC] http://msdn.microsoft.com/en-us/library/cc237917(v=PROT.10).aspx and http://msdn.microsoft.com/en-us/library/cc237949(v=PROT.10).aspx, and [MS-NLMP] http://msdn.microsoft.com/en-us/library/cc236621(PROT.13).aspx for additional information.

[3] [MS-KILE] http://msdn.microsoft.com/en-us/library/cc233855(PROT.13).aspx

[4] Simple and Protected GSS-API Negotiation (SPNEGO) http://msdn.microsoft.com/en-us/library/ms818975.aspx

Recommendations:

Make accounts smart card only to increase the resistance of their NTLM OWF to dictionary and brute-force attacks.

Always use Group Policy to control the use of NTLM on your network, whether smart cards are deployed or not. For all Windows systems, iSEC Partners recommends setting the following Group Policy options under: **Computer Configuration\ Policies \Windows Settings\Security Settings\Local Policies\Security Options**

Network security: LAN Manager authentication level: Send NTLMv2 response only. Refuse LM & NTLM

Network security: Minimum session security for NTLM SSP based (including secure RPC) clients: and

Network security: Minimum session security for NTLM SSP based (including secure RPC) servers:

Require NTLMv2 session security

Require 128-bit encryption

On Windows 7 and Server 2008 R2 and above systems, iSEC Partners recommends the following additional settings:

Network security: Allow Local System to use computer identity for NTLM: Enabled

Network security: Allow LocalSystem NULL session fallback: Disabled

iSEC Partners also recommends auditing and, where possible, disabling NTLM use on Windows 7 and Server 2008 R2 networks. For more information on auditing NTLM usage, see:

NTLM Blocking and You: Application Analysis and Auditing Methodologies in Windows 7

http://blogs.technet.com/askds/archive/2009/10/08/ntlm-blocking-and-you-application-analysis-and-auditing-methodologies-in-windows-7.aspx

and the following resources on TechNet: (with step-by-step guides still yet to be released)

Introducing the Restriction of NTLM Authentication

http://technet.microsoft.com/en-us/library/dd560653(WS.10).aspx

Smart Cards, Long-Term Authenticators and Potentially Malicious Workstations

The previous section's discussion should have already dispelled the myth that smart card users do not have long-term authenticators subject to theft by a malicious workstation. If NTLM is allowed on your network, the OWF sent as part of the Kerberos PAC is a long-term password equivalent. It can be used to perform network authentication with NTLM, as the user, without requiring the presence of the smart card. If the user is not configured for smart card only logon, the OWF is also a password equivalent for Kerberos initial authentication.

Even if NTLM is completely disabled on the network and a user is configured for smart card only logon, a user's TGT is valid for 10 hours and renewable up to 7 days by default.

It is not safe for a highly privileged user to logon to a low integrity workstation, even with a smart card, unless the account used has no rights for Network Logon elsewhere in the forest.

Recommendations:

Make accounts smart card only to prevent a captured OWF from being used to acquire Kerberos credentials indefinitely.

Reduce the lifetime of user and service tickets for smart card users with the Group Policy options available under **Computer Configuration -> Policies -> Windows Settings -> Security Settings -> Account Policies -> Kerberos Policy**

Because Kerberos authentication with smart cards requires computationally expensive asymmetric cryptography, take caution when setting this policy. Setting a very low expiration time for a large user base can place a large load on your KDC and may cause a denial of service for your entire network. Adjust this setting gradually and observe server performance. You may require additional KDCs, processor resources or cryptographic accelerators to support a large user base with a short ticket lifetime.

Provide Domain Administrators with two user accounts and two smart cards. One account should be given interactive logon rights only, the other network logon rights only. One smart card should contain the credentials for the interactive logon only, the other card should contain both sets of credentials. If the NT OWF of an account valid only for interactive logon is captured by a compromised workstation, it will not allow additional elevation of privilege on the network. Train administrators to use the card with credentials valid for network logon exclusively on trustworthy systems, and use the **/smartcard** option of the **runas*** command to perform network operations with that identity.

* http://technet.microsoft.com/en-us/library/bb490994.aspx

Certificate Trust in Public Key Kerberos Implementations

Mutual authentication is a critical security service provided by the Kerberos protocol. A client and the KDC can each verify the identity of the other, as can a client and a service. In standard Kerberos, this is accomplished by using shared symmetric keys. The KDC has a unique, long-term, shared key for every principal in the realm, and the KDC can allow clients and services to be mutually authenticated by creating a new, ephemeral key, and encrypting a copy of it with each of their keys.

Public Key Kerberos leaves most of the basic mechanisms unchanged, but modifies the initial authentication step between the client and KDC to use public key cryptography – typically an X.509 certificate-based PKI.

While this change may seem trivial on the surface, it actually introduces considerable complexity. In 2005, initial drafts of the protocol were found by Andre Scedrov, et al. to be vulnerable to a man-in-the-middle attack which allowed impersonation of the KDC to a client,[5] leading to Microsoft Security Bulletin MS05-042.[6] (CVE-2005-1982[7]) More recently, a variety of attacks the X.509 public key infrastructure against as it is used for HTTPS[8] have further highlighted the importance of correct issuance policies by certification authorities and acceptance policies for protocol participants. What are these policies, and how is trust established, for common implementations of PKINIT Kerberos?

For the purposes of this paper, we will assume that the issuance policies and mechanisms of the certification authorities involved are correct. That is, that the authentication and authorization mechanisms around certificate issuance work as designed and all certificate signing/enrollment requests are well-formed and handled correctly. What we are concerned with are the policies by which KDCs and clients establish trust in the certificates used for PKINIT Kerberos.

Trust decisions around certificates in a PKI[9] are typically made using a few criteria. The first decision is whether to trust the certificate itself as valid. This is done by chaining the certificate to a trust anchor.

Trusted Root Certification Authorities for Public Key Kerberos

For many applications of PKI, operating systems, client programs (e.g. Firefox) or application platforms (e.g. Java) will ship with a pre-configured set of trusted root authorities. The PKI used for HTTPS on the Web, for Authenticode signatures, or signed Java applets/applications, are all examples of this model. All root certification authorities are usu-

---

[5] *Breaking and fixing public-key Kerberos*, Cervesato, Jaggard, Scedrov, Tsay and Walstad, http://portal.acm.org/citation.cfm?id=1350254

[6] http://www.microsoft.com/technet/security/Bulletin/ms05-042.mspx

[7] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CAN-2005-1982

[8] E.g. http://www.win.tue.nl/hashclash/rogue-ca/, http://www.thoughtcrime.org/papers/null-prefix-attacks.pdf, and http://files.cloudprivacy.net/ssl-mitm.pdf

[9] That is, excluding self-signed certificates or those trusted on the basis of a known thumbprint, etc.

ally trusted equally, there are often many, and they may be of varying quality in their issuance policies and due diligence. Thankfully, no major implementation of PKINIT Kerberos uses a default set of authorities for its trust root(s).

In the MIT[10] and Heimdal[11] implementations of PKINIT Kerberos, the set of trust anchors is specified explicitly. A configuration file, typically located at /etc/krb5.conf, specifies a path to a file containing the certificates of the trusted enterprise certification authorities.

```
# from /etc/krb5.conf

[kdc]
# server trust root
     pkinit_anchors = FILE:/path/to/trust-anchors.pem

[libdefaults]
# client trust root
     pkinit_anchors = FILE:/path/to/trust-anchors.pem
```

Obviously, this file must have proper ACLs to prevent its modification by unauthorized users and the insertion or substitution of new root certificates.

In a Microsoft Windows[12] Active Directory network, certificates trusted for PKINIT are found in the NTAuth store. These are cached in the registry of systems joined to the domain at:

HKEY_LOCAL_MACHINE\Software\Microsoft\EnterpriseCertificates\NTAuth\Certificates

and in the Active Directory at:

LDAP://dc1.domain.com/CN=NTAuthCertificates,CN=Public Key Services, CN=Services, CN=Configuration, DC=name, DC=com

Once a certificate has been chained to a trusted anchor, verified according to its validity period, checked for revocation against the CRL or OCSP location, and its signature verified, an application can proceed to make a trust decision using the additional information contained in the certificate.

Certificate Acceptance Policies for KDCs

When a KDC is presented with a client certificate, it will require the following to accept the certificate as valid for PKINIT.

---

[10] http://web.mit.edu/Kerberos/
[11] http://www.h5l.org/
[12] http://www.microsoft.com/windows/, and http://msdn.microsoft.com/en-us/library/aa378747(VS.85).aspx

For Windows Active Directory KDCs[13]:

- The certificate must have a Subject Alternative Name extension of type "Other-Name" set to the UPN of the user account stored in Active Directory
- The certificate must have the Client Authentication KU and Microsoft Smart Card Authentication Enhanced Key Usages.
- The Subject of the certificate should be the Distinguished Name of the user in Active Directory

For Heimdal and MIT KDCs, the requirements are more flexible and depend on the options specified in the configuration file. The "Smart Card Authentication" or id-pkekuoid (1.3.6.1.5.2.3.4) OIDs may be required as an EKU in the certificate. A KDC may be configured to retrieve the principal from the certificate Subject or Subject Alternative Name, or it may have a file that contains an explicit mapping of subject DNs to Kerberos principals.

While, in principle, a KDC configured for PKINIT could simply operate and issue tickets for any presented certificate without maintaining a database of users, in practice every KDC implementation maintains an accounts database, and must be able to discover an account with an appropriate UPN or account mapping. This database and pre-existence of a user account is necessary for the KDC to look up user rights, group memberships and other information frequently conveyed to services as part of a Kerberos authentication, as well as to support user-to-user authentication.

Certificate Acceptance Policies for Kerberos Clients

To complete the trust relationship between users and their Kerberos domain, the KDC must also be identified by the user. In traditional Kerberos this authentication is accomplished by means of the secret shared between the KDC and the client. In Public Key Kerberos, a certificate acceptance policy is the means by which a client establishes trust in the KDC. The behavior and configuration options of the three most widely deployed Kerberos clients and servers show considerable variation in this area, and the vulnerabilities discussed in the remaining sections of the paper arise from these variations and deviations from the standard.

The most familiar example of an acceptance policy for X.509 certificates is the one employed by SSL and TLS. To be valid for HTTPS, a certificate for a web server must have a Server Authentication EKU extension, and the CNAME field of the Subject, or a SubjectAltName, must match the DNS name of the host the client was attempting to reach.

In PKINIT, SSL-style name matching is not used because a client may not know the name of a valid KDC. A Windows server or workstation joined to a domain has to perform a discovery process to identify the nearest and most available KDC. This process traditional-

[13] These are the default requirements. Beginning in Windows Server 2008 these requirements are all configurable to enable support for cards that do not meet these requirements, such as smart card-based national ID cards.

ly uses insecure protocols, such as local NetBIOS broadcasts or DNS SRV record lookups. Because the name is not discovered securely, matching the name to a presented certificate cannot provide meaningful assurance in the general case. MIT and Heimdal clients often have the KDC(s) specified by a configuration file, but the Kerberos RFCs do not specify that a server host name be present in a KDC certificate, so even these clients are not typically configured to validate names in a KDC certificate. There is an optional configuration parameter for the MIT client, pkinit_kdc_hostname, which enforces matching with the KDC certificate's DNS Subject Alternative Name. It is not enabled by default.

If name verification is not generally used, what verification mechanism does the protocol specify, and what have clients implemented?

RFC 4556 specifies that the Digital Signature Key Usage and id-pkinit-PKPKdc Enhanced Key Usage should be present in certificates issued to a KDC.

RFC 4556 also specifies that, "Unless the client can otherwise verify that the public key used to verify the KDC's signature is bound to the KDC of the target realm, KDC's X.509 certificate MUST contain a Subject Alternative Name extension [RFC3280] carrying an AnotherName whose type-id is id-pkinit-san (as defined in Section 3.2.2) and whose value is a KRB5PrincipalName that matches the name of the TGS of the target realm (as defined in Section 7.3 of [RFC4120])." The Heimdal client provides a configuration option to make this check, but Windows and MIT clients appear to ignore this protocol option.

Where the specifications are not clear, examining the certificates issued to typical KDCs can help clarify client verification policies. The certificates issued by common tools for MIT and Heimdal KDCs contain the id-pkinit-PKKdc EKU, and testing verifies that the clients check for that EKU in their default configuration.

Questions arise when the certificates issued to Windows Active Directory KDCs are examined. The KDC of a Windows Server uses the same certificate issued for all Domain Controller tasks. The details of this certificate are described in the Microsoft KB article 291010, "Requirements for Domain Controller Certificates from a Third-Party CA"[14].

- It must contain a valid CRL distribution point.
- It should contain the directory path of the server object in the Subject field
- The specified Key Usages must include Digital Signature and Key Encipherment
- The specified Enhanced Key Usages must include Client Authentication and Server Authentication
- The Subject Alternative Name section must contain:
    - A type OtherName containing the GUID of the domain controller in the Active Directory
    - A type DNSName containing the full name of the server (e.g. dc1.testdom.com)
- Must contain an extension of type Certificate Template (1.3.6.1.4.1.311.21.7) with the BMP value of "DomainController"

[14] http://support.microsoft.com/kb/291010

Beginning with Windows Server 2003, Domain Controller Authentication certificates additionally contain the Microsoft Smart Card Authentication EKU (1.3.6.1.4.1.311.20.2.2, identical to the extension required in smart card client authentication certificates).

Note that the id-pkinitPKKdc EKU is *not* specified in this list.

How can clients verify a Windows Server KDC without the PKINIT KDC EKU? For interoperability, Heimdal and MIT provide several flags to disable or change their verification behavior when interoperating with Active Directory.

Heimdal provides a "pkinit_require_eku" configuration option. This must be set to "false" when using a Windows KDC with a standard certificate. MIT provides a "pkinit_eku_checking" option. The default value, "kpKDC" requires the EKU. Other options include "kpServerAuth" to check only for the Server Authentication EKU, and "none" to disable all checking. (This option is explicitly not recommended in documentation.)

Windows client behavior was not documented for some time. Beginning in 2008 (with Windows Vista SP1 and Windows Server 2008) a new group policy option appeared: "Require Strict KDC Validation."[15] In the description for this policy setting, it states: "If you disable or do not configure this policy setting, the Kerberos client requires only that the KDC certificate contain the Server Authentication purpose object identifier in the EKU extensions."

Is this enough to distinguish a KDC from other systems in the domain? In a 2006 whitepaper[16], Microsoft cautions that it is not:

"When a domain-joined client computer performs a PKINIT with a server, the client needs to be able to verify that the other computer has a valid certificate and that it actually is also a Domain Controller. The domain controller and the Domain Controller Authentication certificate add the domain controller's fully qualified domain name (FQDN) to the certificate. However, with this information, a client is not able to truly verify whether the machine is a valid domain controller because a client does not have an authoritative list of all valid domain controllers for a domain. Therefore, the Kerberos Authentication certificate template adds the domain name instead of the domain controller's FQDN to the certificate."

If clients only verify the Server Authentication EKU, what other systems have certificates that meet the verification criteria for a Windows KDC certificate?

The Web Server template supplied by Microsoft Active Directory Certificate Services contains this EKU. There may be many dozens of such certificates in a large enterprise, and internal web applications are notorious for their security weakness. Web Server certificates

---

[15]" Windows Server 2008 Group Policy settings for interoperability with non-Microsoft Kerberos realms," http://support.microsoft.com/kb/947706

[16] Active Directory Certificate Server Enhancements in Windows Server Code Name "Longhorn", http://www.microsoft.com/downloads/details.aspx?FamilyID=9bf17231-d832-4ff9-8fb8-0539ba21ab95&displaylang=en

from the enterprise authority are also commonly issued to non-Windows systems, creating a still wider set of possible exploitation vectors. Even in an environment in which Network Access Protection is deployed, at least one server with this kind of certificate, the remediation server, will be accessible even to non-compliant clients in the quarantine zone.

More troubling than web servers, the "Computer" certificate template also contains the Server Authentication EKU. When Active Directory Certificate Services are installed in a domain, the Computer template is enabled by default, and every computer account in the domain has permission to enroll by default. This means that every single Windows workstation in the domain can acquire a credential which might be used to impersonate the KDC.

Practical Exploitation and Elevation of Privilege

Is this really true? Can any certificate with a server auth EKU be presented as a KDC certificate and be accepted by a client? iSEC Partners' testing showed that this was indeed the case. Heimdal and MIT clients configured for Windows interop, and Windows clients up to and including Windows 7, will accept a PKINIT AS-REP signed by any certificate issued with the "Web Server" and "Computer" templates by the enterprise authority.

For the MIT and Heimdal clients, the 'kinit' command with a smart card from the command line allows an easy elevation of privilege path. A server account with a certificate of the appropriate type (either a web server or any Windows workstation in the domain) is compromised by the attacker. With the ability to actively influence the network traffic of the victim (by ARP spoofing, hijacking of name resolution or DHCP poisoning) the attacker inserts himself as the KDC on port 88 TCP and UDP, receives the user's AS-REQ, replies with his own AS-REP, signed by his captured certificate, and is trusted as the KDC. If the user makes any subsequent actions authenticated by Kerberos, the active network attacker can again insert himself into the traffic and spoof the remote service. For an SSH session, this might allow capture of a sudo password, or if Kerberos is used to authenticate a remote file server, it may be possible to supply a trojan executable and take full control of the user's session.

Creating an elevation of privilege path for a Windows client is somewhat more difficult. First, in addition to providing Kerberos services on port 88, the attacker must provide appropriate resolution services to identify himself as the KDC. This includes DNS or NetBIOS SRV records to identify and CLADP to impersonate the Domain Controller. Still, this is relatively easy to accomplish, as these bootstrap protocols are all unauthenticated.

The next step involves successfully completing a smart card logon. Unlike the MIT and Heimdal clients, which are typically invoked from the command line, Windows PKINIT happens only with a smart card, and typically only when a user is logging on or unlocking a workstation. First, the user contacts the KDC and makes an AS-REQ. The attacker can successfully man-in-the-middle this request and supply an AS-REP with his captured certificate. This will be trusted by the client but, immediately following this, the client makes

a TGS-REQ to obtain a service ticket to the workstation computer account. This is then used locally to authenticate the user to the workstation and complete the logon process.

At this point the naïve attack fails. The impostor KDC can convince the user it is genuine with only a captured certificate, but to formulate a valid service ticket for the workstation computer account, it must know the secret shared between the computer and the genuine KDC. Public Key Kerberos provides facilities only for initial authentication, not generation of service tickets. The impostor KDC cannot give the user a valid service ticket for the workstation, and so the logon attempt fails.

Perhaps this failure to complete authentication with a simple man-in-the-middle attack explains the less-than-emphatic nature of Microsoft's guidance on strict KDC validation. Nevertheless, it is possible to turn this break in the trust path into a viable elevation of privilege with a bit more work.

Domain join is one scenario in which no service ticket is required. Prior to joining a domain, the computer workstation account does not exist. The account and its password are created as part of the domain join process, which is entirely bootstrapped on user credentials and trust. If smart card credentials are used to perform the domain join process (this is likely in a smart card only environment), the impostor KDC can spoof the entire process and take full control of the machine.

More interesting than simply controlling the machine without joining it to the legitimate domain is to complete the join to the real domain, but maintain covert control of the system. This can be accomplished in two ways. If the attacker already controls the credentials of a user with rights to join a system to the domain, the impostor KDC can act as a man-in-the-middle between the workstation and the real KDC, recording and forwarding (authenticated as the collaborating user) the traffic stream from the workstation. At the end of such a process, the impostor KDC knows the machine account credentials, and can maintain control of it for the lifetime of that password, leaving no trace of its presence on the system. The only forensic indication of anything unusual would be that the domain join log at the DC (event ID 645) would show the collaborating user rather than the expected user.

Perhaps no collaborating user with appropriate access is available. If, on the reboot following domain join, the same, domain-join-privileged, user logs in again, the attacker can accomplish the same task. An executable to be run as Local System can be pushed via group policy mechanisms on the initial domain join reboot, and when the privileged user authenticates again, the executable can hijack the credentials to make the necessary calls to a legitimate DC, add the computer account with the same machine password, re-authenticate the system and user to the real KDC, then remove itself. Again, the attacker knows the password of the workstation and can control it until it is refreshed with no forensic trace.

To maintain longer term control, typical rootkit or other means could be employed as well, though at some greater risk of detection.

Recommendations:

Limit the user accounts which have the privilege to join a system to the domain.

Use offline domain join, (http://technet.microsoft.com/en-us/library/offline-domain-join-djoin-step-by-step(WS.10).aspx) or join systems to a domain on an isolated, trusted network.

Consider using an account with a strong password instead of a smart card for domain join.

After domain joining a system with a privileged account, do not logon to the system with that account again.

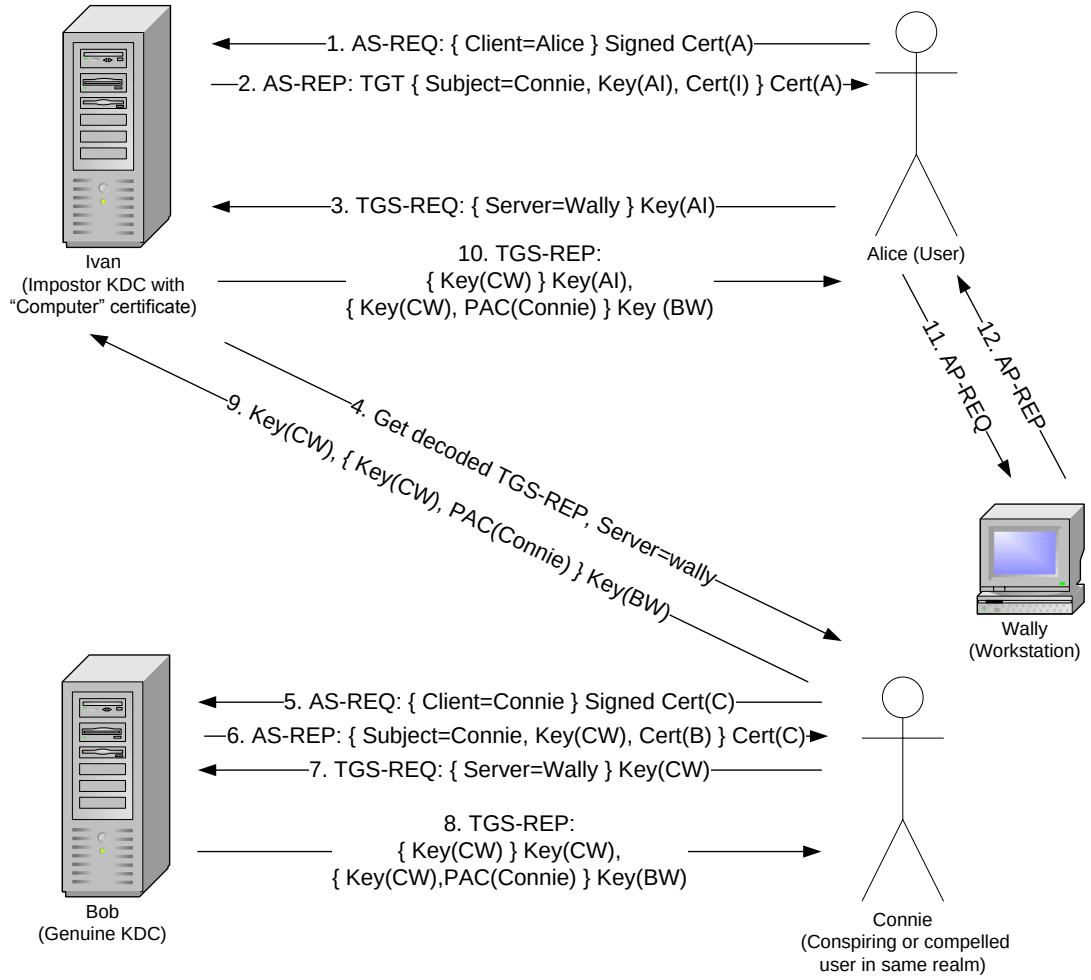
Domain join is an uncommon, one-time event. A typical attacker will find a network of systems already joined to a domain. Is there a way to get around the lack of knowledge of the workstation account password? If the attacker also has control of any ordinary user account in the domain, there is.

Control of the conspiring user account might be obtained in a variety of ways. The attacker may have a willing accomplice in the domain. He may have compromised the password of a non-smart card only account. He may have obtained system-level control of a workstation via some other exploit, and is able to capture the pin and use the inserted smart card of an unwitting user.

To accomplish the exploit, the attacker will force the new victim to unwittingly logon to the workstation as the conspiring user. All users in a domain will typically have rights to logon to all workstations, and this is assumed here.

To describe the attack, we will designate the impostor KDC as Ivan, the conspiring/compromised user as Connie, the victim as Alice, the real KDC as Bob, and the workstation as Wally.

The attack proceeds as follows:



1. Alice's attempt to logon with a smart card (PKINIT AS-REQ) is intercepted by Ivan.
2. Ivan sends an AS-REP, with a TGT key, back to Alice, encrypted with her public key. The client principal of the AS-REP and TGT is not set to Alice's identity, but to Connie's. (the encryption notation of the AS-REP in the diagram has been simplified for clarity)
3. Alice accepts Ivan's AS-REP, and makes a TGS-REQ for Wally, to complete her logon.
4. Ivan asks Connie for credentials to Wally.
5. Connie makes an AS-REQ to Bob.
6. Bob gives Connie a TGT.
7. Connie makes a TGS-REQ for Wally.
8. Bob gives Connie his TGS-REP for Wally.
9. Connie decrypts the TGS-REP and sends its contents (Connie's copy of the session key, and Connie's ticket to Wally) to Ivan.
10. Ivan forms a TGS-REP for Alice containing the session key received from Connie, and Connie's ticket to Wally (as originally issued by Bob).
11. Alice receives the TGS-REP. She can decode and extract the session key and ticket. She forms an authenticator and makes an AP-REQ to Wally.
12. Wally decrypts the ticket, validates the authenticator (formed with the same session key in the ticket) and PAC checksum (created by Bob), and creates a logon session for Connie.

The careful reader will have some questions about this progression. The first question arises at step 2, when Ivan sends Alice a TGT for the client principal name Connie. Why would Alice's client accept this as a valid AS-REP, when she asked for a ticket for Alice? The answer is user principal name canonicalization.

Not yet a standard, the user principal name canonicalization process is described by the IETF Kerberos Working Group Draft 11 of "Kerberos Principal Name Canonicalization and KDC-Generated Cross-Realm Referrals"[17] The purpose of user principal name canonicalization is to allow a user to logon with a familiar name that is an alias for the user's Kerberos principal name. This allows for administrators to move a user's principal to other realms without the user having to know this has happened, or, in the case of smart cards, to associate a third-party (e.g. government-issued) smart card with a Kerberos principal in the realm. When user principal name canonicalization is used, "The user principal name no longer has a direct relationship with the Kerberos principal or realm," and "If the "canonicalize" KDC option is set, then the KDC MAY change the client and server principal names and types in the AS response and ticket returned from the name type of the client name in the request. In a TGS exchange, the server principal name and type may be changed."[18] The KDC must authenticate the name mapping with a checksum using the AS reply key, but in this attack the client already trusts Ivan as the KDC and is using Ivan's chosen AS reply key.

Windows implements user principal name canonicalization by default, so it will accept the AS-REP for Connie in step 2. (In fact, there is no way to disable user principal name canonicalization.) In a smart card enabled Active Directory Domain, we can observe this in action by starting the "Active Directory Users and Computers" tool, and under the "Account" tab on the user properties page, switching the "User logon name:" of two users. UserA's smart card will now log them on as UserB, and vice-versa. The certificate subject on the smart card is not your account – it is only a pointer to your account, and the client cannot know a-priori what account it points to.

This brings us to the next question. Who has been logged in? Alice thinks she is logged on, but she has logged on as Connie. If we began by assuming that the attacker already has control of Connie, is this really an elevation of privilege?

It isn't yet. But the attacker is in a very good position to finish the task. Alice has logged in, with a smart card and her PIN, at a healthy and trusted workstation. That she would actually be logged in as another user account is totally unexpected behavior, and it is unlikely she would suspect or detect it before the attacker can take action. The attacker has a user sitting at a workstation she trusts, with her smart card inserted, and the attacker is in control of her interactive session. The attacker has control of Connie's credentials, and can be the Domain Controller to Connie. He can push an executable to run immediately at logon via group policy or other mechanisms.

---

[17] http://tools.ietf.org/html/draft-ietf-krb-wg-kerberos-referrals-11
[18] *ibid.*

To complete the elevation of privilege, the attacker might run an executable which simulates installation of updates, and after a period, shows the workstation unlock screen. Alice may insert her smart card and enter her PIN to unlock the workstation. The real smart card login or unlock sequence requires the secure attention sequence (Ctrl-Alt-Delete) but Alice may easily fail to notice this step has been left out. If she does, the trojan program can capture the PIN, unlock the smart card cryptographic service provider, make a user-mode AS-REQ to the real KDC and decrypt the resulting TGT. The attacker now has access to all resources (except interactive logon) as Alice for the renewal lifetime of the ticket. The attacker can also extract Alice's NT OWF from the TGT, which will provide indefinite access to any network resources which accept NTLM authentication. If Alice is not a smart card only user, the OWF can also be used for non-PKINIT Kerberos authentication, and may be subject to a brute force attack to recover the plaintext password. The attacker might also use the short-term access to Alice's credentials to install malware that provides long-term control of Alice's future sessions at that workstation. If Alice is in the local administrators group, the attacker can take full control of the workstation.

After capturing Alice's PIN and leveraging it, the malicious policy executable could then pretend to finish installing updates, remove itself and force a reboot. Alice is very unlikely to be aware of the attack that has just taken place against her.

To complete his ultimate intentions, the attacker repeats this process until a user or workstation with access to the information or privilege level he wants is eventually compromised.

Q: Will a smart card reader with a hardware-integrated PIN pad prevent this attack?

A: No. It will make it harder to execute, because a malicious program cannot capture and store the PIN to make silent use of the smart card whenever it is inserted, but fundamentally the operations of the smart card cryptographic service provider are available from user mode applications at ordinary privilege through various APIs. If the attacker can run code in the user's context, it is very likely he will be able to trick the user into entering her PIN code for what she believes is an ordinary, but is in fact a malicious, signing or encrypting operation. PKINIT Kerberos is inherently a mechanism designed to leverage one-time usage of the smart card into a long term credential: the TGT (and with it, the NT OWF).

Recommendations:

Beginning with Vista SP1 and Windows Server 2008, the following policy option is available under: **Computer Configuration\Administrative Templates\System\Kerberos**

Policy: Require strict KDC validation

iSEC recommends this be set to **Enabled**.

For MIT and Heimdal clients, use the default settings which require the PKINIT EKU.

To prepare for enabling this setting, all Domain Controllers must first be issued an appropriate certificate. The "Domain Controller" and "Domain Controller Authentication" templates available in Active Directory Certificate Services for Windows 2000 and Windows Server 2003 do not contain the necessary EKUs to pass strict validation. Use the new certificate template, "Kerberos Authentication", available in Windows Server 2008, to issue certificates to all Domain Controllers with the proper KDC key purpose EKU OID.

For Windows Server 2008 and later Domain Controllers, if a certificate with this template is present the Domain Controller will automatically prefer it and there is no need to revoke existing certificates issued against the older templates.

Note: Although the previously referenced Microsoft guide to *Certificate Services Enhancements in Longhorn Server* indicates that the Kerberos Authentication template will be used by default for Windows Server 2003 and Server 2008 Domain Controllers when the CA is running on Windows Server 2008, iSEC's limited testing in a small network with default configurations of AD Certificate Services, using Server 2008R2 for both the CA and DCs, showed that the old, default "Domain Controller" template was still used to issue DC certificates.

Note: In iSEC's limited testing, enabling the Kerberos Authentication template and setting the autoenroll permission for Domain Controllers also did not reliably cause all servers to re-enroll. Use "certutil.exe –DCInfo" to audit the certificates in use by DCs and manually enroll any that do not receive new certificates issued with the updated template.

Windows XP clients cannot be hardened against this attack. iSEC recommends upgrading workstations to be used with smart card logon to Windows Vista or Windows 7. If workstations cannot be upgraded, extreme care must be used in the issuance and control of certificates with the Web Server and Computer templates by the Enterprise Certification Authority. Disable enroll and autoenroll permissions for these templates and require that a domain administrator manually approve all CSRs.

Discussion

This KDC impostor attack is identical in its implications to the Scedrov attack of 2005, except that it requires control of both a workstation and a user account to complete the elevation, and exploits improper validation by the client rather than an inherent protocol flaw. The implications of user principal canonicalization and the full elevation of privilege scenario existed in Scedrov's original attack, but were either not fully understood or not documented by his team. It is not clear if a working exploit scenario was proven at the time, or if it was merely a theoretical break.

This attack can be mitigated by some simple configuration steps, but nearly all clients configured to use a Windows-based KDC today will be vulnerable: Windows, MIT and Heimdal alike.

Most of this activity will appear to be perfectly normal to most network traffic analysis. The basic patterns and protocols are unchanged, the services must be allowed through firewalls, and the key portions of the exploit are hidden inside encrypted authentication protocols. DNSSEC cannot be used to prevent exploitation by an attacker able to ARP spoof or control a gateway, because no certificate DNS name validation is performed as part of PKINIT. IPSec is also unlikely to help, as Kerberos traffic is typically exempt from IPSec when configured via group policy, since it is used, with IKE, to bootstrap IPSec.

Are smart cards better than passwords for network authentication and Windows logon? Absolutely, but they are not a silver bullet. Administrators defending their resources against what are commonly known as "Advanced Persistent Threats (APTs)" need to be aware of the underlying mechanisms and limitations of smart card technology. Installations should take advantage of the latest configuration and hardening options available, administrators should continue to audit and work to eliminate outdated protocols like NTLM from their networks, and privileged users should always refrain from authenticating to low-integrity workstations, even with a smart card.

Tools

As part of this research, iSEC Partners has released an updated version of our CybervillainsCA tool, which now provides command-line options to create .cer, .pem and .p12 files suitable for use as third-party Domain Controller and Smart Card Authentication certifcates. This tool will be available for download at https://www.isecpartners.com/tools.

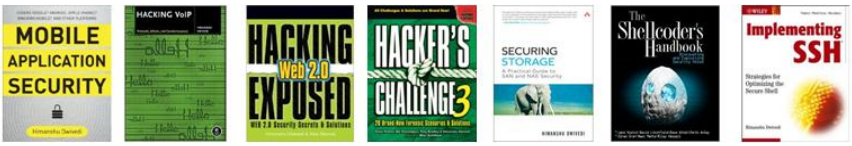
The software developed for the full exploit scenario was a proof-of-concept only and will not be released as it has no legitimate use outside a research context.

Appendix A: About iSEC Partners, Inc.

iSEC Partners is a proven, full-service security firm, dedicated to making Software Secure. Our focus areas include:

- Mobile Application Security
- Web Application Security
- Client/Server Security
- Software Development Lifecycle Security (Microsoft SDL)
- OnDemand Web Application Scanning (Automated/Manual)

Published Books



Notable Presentations

Whitepaper, Tools, Advisories, & SDL Products

- 18 Published Whitepapers
  - Including the first whitepaper on CSRF
- Over 40 Free Security Tools
  - Application, Infrastructure, Mobile, VoIP, & Storage
- Advisories on products from vendors inlcuding Google, Apple, and Adobe
- Free SDL Products
  - SecurityQA Toolbar (Automated Web Application Testing)
  - Code Coach (Secure Code Enforcement and Scanning)
  - Computer Based Training (Java & WebApp Security)

Comments and questions to: Brad Hill, brad@isecpartners.com


```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP Desktop 9.5.1 (Build 1557)

mQENBENGrWoBCADYj3A87oEsj15nMGUkerMDBYJNfY561CuQuBh+MbQdoKwn8l8t
qaMGNQlQ00CCYn0qF3EN2FJGKer+10D1GL+BUUENnv4SeNMIWor6SAEAVlyvZE/j
3nj888qNTaXcdKgplTvSRnXYMQZoSj4rQSeCyEy0OrGVsp++GcryNoUMBay0hIPw
MP/a2Z5fde6jOygDwoERPHr3eip50ISs7Xu60PqSMjSnLKoEFDow5SdEl4D4l/Fz
H7UGTDXb8fFYjSuDAONMR1hZJwzUpC0iKbJq6ZLt2gypyLMo7vgD2BXYY8dlTca5
eXhYF7+jcyzEai5matks4pOC4tsocPoeIvobABEBAAG0IUJyYWQgSGlsbCA8YnJh
ZEBpc2VjcGFydG5lcnMuY29tPokBbQQQAQIAVwUCQ0atajAUgAAAAAAgAAdwcmVm
ZXJyZWQtZW1haWwtZW5jb2RpbmdAcGdwLmNvbXBncG1pbWUHCwkIBwMCCgIZAQUb
AwAAAAMWAgEFHgEAAAAEFQgJCgAKCRASYD2BCTAzDoxTCADRXQfeB6yNE7i7KqEf
HD0LwCuO++7Xl106jc4jX1FjGmWUNpIhBQmKogh0ov+KFNPspl88Z/qwLVpGIi4U
Vrys0bVYVAX/no5q00PBSKQN3kgFGX6ttVo9rAbfHeHDXqFqPPQ9L0YoLiTtl3DD
LWmNlcwvPbuTkCnC62nAOI9VzRStRrbZOutGaTIlRV0vIg4baHADRayn30qLmys7
ZcwE5G5si30PO4IrKz7OY58vDA4b6EQIgrWrtoQnZ/CHVjncBQAdPPvlmUia2yjA
ZyOWhfUT57ioO4iH0avjvWHU0KFaWkvREnYRYjnFnh6TRJ1eoVxE3csIekxOdl3C
AJ0liQBGBBARAgAGBQJDUB1ZAAoJEAc00PTkWgzwUZEAnifkf2lF3wmwl9j2gkno
Qc1U5CVTAKDPWfk2oIdMTIEDkF1pbvjb5cjBH7kBDQRDRq1qAQgAylkYfKfqlErQ
6VLkDgbtSGdp5z9XwbejhFUmsPway+94IgQ0aeyKChFq15gkF69ojVeAIH5Vq8Gj
FWRZFgxL1tEnqPh2YyWaS5czQCMOc9vx23m2PZuWpYR6batqPm5KqrwUlSCfuDAH
f8NBnaJdx8CeCUIhxv+nr8+cK5C1UIczbaVA1Af2tQM/YqHU8KOQk0rYGMSt7pBK
3stWYv9KwKXZ/5JHJ1oX2KSfeVRXLL9MUW4jFl6CFX6pjVUy+V2ZlFFWs333KYwW
SPgL9RAf8/DyTYKC6mtp5n6DvYZEfuvu7cWB3NmtDFwHn3hG3DH6OsW+bG276spX
+b7EYX1U8wARAQABiQEiBBgBAgAMBQJDRq1qBRsMAAAAAAoJEBJgPYEJMDMOEOMI
AK6KrIzSvGecxDP0zi1gEf7FxREVW12RpdrqU+xa7471Bw1gDMMQr4+NKYtzudXp
lzn/BPAufzjqQyPUx3oXfCs15hd5DyQ18d1nl7ddyj6U9YyJCl+rRroYIykgnUCz
oiEFd4iL/ZSTx5KEWWa/xokTezflwQ2CMIosfxzDj4I0I38yHw00Z8+tZdBdr5Ku
PT6civy1GOaFziVUUVYubBpvHpj2VfPJA743KgB3xToo670HgLiTWAu/fXVaMHsN
b3/oNLhZfSDKkZe/v2yrxW3Jv5K81iB7zgaSi7JK9UFEXMh+8CE9sAGAV6ePRYif
IjXflxgeKp9H0gKPN0wA8bw=
=xgGH
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN CERTIFICATE-----
MIIHHDCCBgSgAwIBAgIKIoQQ/gAAAAABwDANBgkqhkiG9w0BAQUFADBfMRMwEQYK
CZImiZPyLGQBGRYDY29tMRwwGgYKCZImiZPyLGQBGRYMaXNlY3BhcnRuZXJzMSow
KAYDVQQDEyFJbmZvcm1hdGlvbiBTZWN1cml0eSBQYXJ0bmVycyBMTEMwHhcNMDkx
MTI0MjIxODAyWhcNMTAxMTI0MjIxODAyWjCBpTETMBEGCgmSJomT8ixkARkWA2Nv
bTEcMBoGCgmSJomT8ixkARkWDGlzZWNwYXJ0bmVyczETMBEGA1UECxMKTXlCdXNp
bmVzczEOMAwGA1UECxMFVXNlcnMxETAPBgNVBAsTCFNCU1VzZXJzMRIwEAYDVQQD
EwlCcmFkIEhpbGwxJDAiBgkqhkiG9w0BCQEWFWJyYWRAaXNlY3BhcnRuZXJzLmNv
bTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAwRjyUnZyukk5Npr3+ZddPEB9
b+tedqGvjYZDfb6MkLHXdC9YuIEFSsYbgBaFYOhB+E5jBMlox1xC7KY9InL7v1kQ
UgguKlyL6zcpJja/K5N3XB2hdwCp5IFfcDCA1+LSrsLXSLsbtdRMBREQoIwaNUMk
6/O74fktl846lS4xUTsCAwEAAaOCBBUwggQRMBcGCSsGAQQBgjcUAgQKHggAVQBz
AGUAcjAdBgNVHQ4EFgQUHiwcUIOdOuFv+z1gKnbZ7Fx0D0MwCwYDVR0PBAQDAgWg
MB8GA1UdIwQYMBaAFOnGwjQgtj+hyhenBHtE7MjDYjiwMIIBkgYDVR0fBIIBiTCC
AYUwggGBoIIBfaCCAXmGgdBsZGFwOi8vL0NOPUluZm9ybWF0aW9uJTIwU2VjdXJp
dHklMjBQYXJ0bmVycyUyMExMQyxDTj1kYzAxLENOPUNEUCxDTj1QdWJsaWMlMjBL
ZXklMjBTZXJ2aWNlcyxDTj1TZXJ2aWNlcyxDTj1Db25maWd1cmF0aW9uLERDPWlz
ZWNwYXJ0bmVycyxEQz1jb20/Y2VydGlmaWNhdGVSZXZvY2F0aW9uTGlzdD9iYXNl
P29iamVjdENsYXNzPWNSTERpc3RyaWJ1dGlvblBvaW50hlNodHRwOi8vZGMwMS5p
c2VjcGFydG5lcnMuY29tL0NlcnRFbnJvbGwvSW5mb3JtYXRpb241MjBTZWN1cml0
eSUyMFBhcnRuZXJzJTIwTExDLmNybIZPZmlsZTovL1xcZGMwMS5pc2VjcGFydG5l
cnMuY29tXENlcnRFbnJvbGxcSW5mb3JtYXRpb24gU2VjdXJpdHkgUGFydG5lcnMg
TExDLmNybDCCAVcGCCsGAQUFBwEBBIIBSTCCAUUwgcsGCCsGAQUFBzAChoG+bGRh
cDovLy9DTj1JbmZvcm1hdGlvbiUyMFNlY3VyaXR5JTIwUGFydG5lcnMlMjBMTEMs
Q049QUlBLENOPVB1YmxpYyUyMEtleSUyMFNlcnZpY2VzLENOPVNlcnZpY2VzLENO
PUNvbmZpZ3VyYXRpb24sREM9aXNlY3BhcnRuZXJzLERDPWNvbT9jQUNlcnRpZm1j
YXRlP2Jhc2U/b2JqZWN0Q2xhc3M9Y2VydGlmaWNhdGlvbkF1dGhvcml0eTB1Bggr
BgEFBQcwAoZpaHR0cDovL2RjMDEuaXNlY3BhcnRuZXJzLmNvbS9DZXJ0RW5yb2xs
L2RjMDEuaXNlY3BhcnRuZXJzLmNvbV9JbmZvcm1hdGlvbiUyMFNlY3VyaXR5JTIw
UGFydG5lcnMlMjBMTEMuY3J0MCkGA1UdJQQiMCAGCisGAQQBgjcKAwQGCCsGAQUF
BwMEBggrBgEFBQcDAjBHBgNVHREEQDA+oCUGCisGAQQBgjcUAgOgFwwVYnJhZEBp
c2VjcGFydG5lcnMuY29tgRVicmFkQGlzZWNwYXJ0bmVycy5jb20wRAYJKoZIhvcN
AQkPBDcwNTAOBggqhkiG9w0DAgICAIAwDgYIKoZIhvcNAwQCAgCAMAcGBSsOAwIH
MAoGCCqGSIb3DQMHMA0GCSqGSIb3DQEBBQUAA4IBAQAAuF3TL6GD+MaTBy3Uprpt
kUNGEpj2dOHY10G5YBVNBooUfYMz/MjjIG8FCTSew/zCjb34bQCYv8V5iLzRuYg5
Vl0SQuqBxHzQUPK7XZWZKTTqQk3rEIBBv9+3bLxstWwSPTzxOP41NwIsfEUlIYNy
Q8fbpWiyV6Lw9xN/eQaHXZnFKmCgglBBeNGpp5D2YVluKbXYlhunFm5IhpUn1ixS
sgGepjWemFYwoslFAWudSFkCDxZBeXSeLMN6t7zFYZy3cuq+p5DdkD2cB0KfOuTM
zDoKfusLdc1Z1YGPINnInNhmFXZbNwN0Frrmdqv U8+W2nqqIJs+HlQD0yJlJ83GZ
-----END CERTIFICATE-----
```