Attacking Kerberos Deployments

Rachel Engel, Brad Hill and Scott Stender
{rachel, brad, scott}@isecpartners.com

This paper contains supplemental and summary information on the presentation "Attacking Kerberos Deployments" to be delivered at Black Hat USA 2010, July 28, 2010, in Las Vegas, Nevada.

This whitepaper covers the following areas:

- References
- Initial Authentication and Etype Negotiation
- Smart Card Login and PKINIT Kerberos
- Kerberized Application Security
- Related Prior Work

As the full details of the research related to this presentation could fill a small book, we ask the interested reader to refer to the "Independent Research" section of iSEC Partners' website, https://www.isecpartners.com/, to download full whitepapers on each of the major topics discussed here, to be released shortly, including recommendations and step-by-step advice on remediation.

This paper is also not intended to provide a full introduction to or description of the full mechanisms of the Kerberos protocol, as this is well-documented elsewhere. Please see the references on the following page for overview discussion and technical specifications of the various protocols and algorithms employed.

Informative references:

The Kerberos Network Authentication Service
http://www.kerberos.info/

Microsoft Kerberos:
http://msdn.microsoft.com/en-us/library/aa378747(VS.85).aspx

Mark Walla, “Kerberos Explained”, May 2000
http://technet.microsoft.com/en-us/library/bb742516.aspx

Windows 2000 Kerberos Authentication
http://technet.microsoft.com/en-us/library/bb742431.aspx

Garman, Jason. Kerberos: The Definitive Guide. O’Reilly, 2003

Normative references:

Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005

Zhu, L., Jaganathan, K., and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2", RFC 4121, July 2005.

Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, June 2006.

Raeburn, K. and Zhu, L., "Kerberos Principal Name Canonicalization and KDC-Generated Cross-Realm Referrals", draft-ietf-krb-wg-kerberos-referrals-11, July 2008.

Raeburn, K. “Encryption and Checksum Specifications for Kerberos 5.” RFC 3961, February 2005.

Raeburn, K. “Advanced Encryption Standard (AES) Encryption for Kerberos 5.” RFC 3962, February 2005.

Jaganathan, K., Zhu, L., and Brezak, J., “The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows”, RFC 4757, December 2006.

[MS-PKCA]: Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol Specification, [MS-PKCA], June 2010.

[MS-KILE]: Kerberos Protocol Extensions, [MS-KILE], June 2010

[MS-APDS]: Authentication Protocol Domain Support Specification, [MS-APDS], June 2010

[MS-PAC]: Privilege Attribute Data Structure, [MS-PAC], June 2010

Initial Authentication and Etype Negotiation

One major limitation to Kerberos v4 was that it was hard-coded to use only the DES encryption algorithm. DES provided a sufficient level of security when Kerberos v4 was released, but the algorithm is now 35 years old. Unlike Kerberos v4, the Kerberos v5 protocol is not tied to any specific set of cryptographic primitives. By making this a loose binding, Kerberos v5 as a protocol has been able to be resistant to attacks against specific cryptographic primitives. However, with this flexibility comes the need to negotiate primitives that are agreeable the client, KDC, and service. This is accomplished via negotiation of "encryption types."

Encryption Types

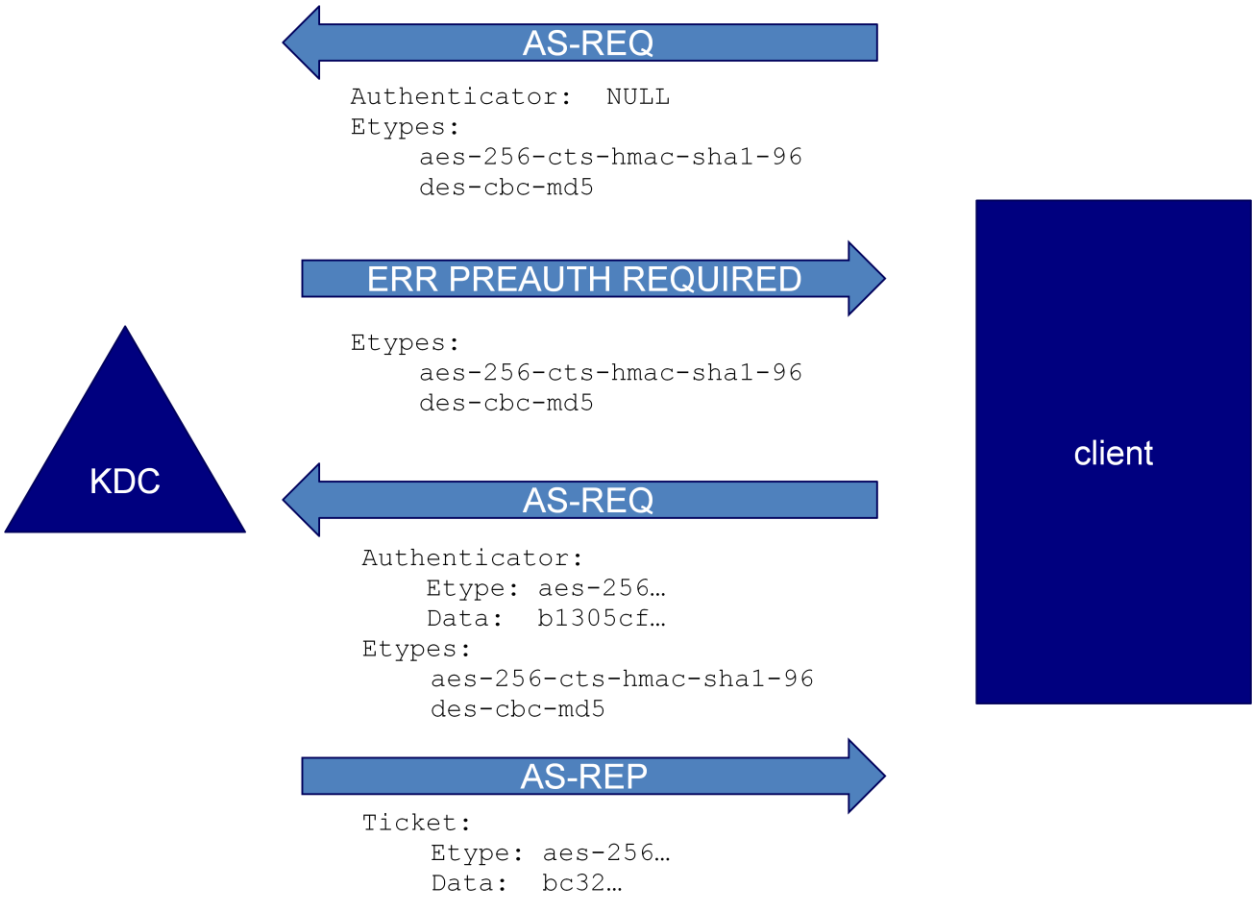
Encryption Types, or "etypes" for short, are a roughly-standard set of cryptographic primitives and standards for their use in Kerberos. In other words, they describe a set of a cipher, integrity function, block cipher mode, and guidance on encrypting and hashing data, much like a TLS cipher suite.

Each etype is referenced in protocol messages using an integer-type identifier. For example, RFC 3961 assigns the value "3" to the des-cbc-md5 etype and provides details for how to manage data protected according to this standard. Similarly, RFC 3962 defines standards for AES-based etypes, and RFC 4757 provides guidance for RC4-based etypes used by Microsoft Windows.

Negotiation of Etypes

The etype used to protect sensitive data in protocol messages must either be negotiated or inferred from context. Examples of protected data include the authenticator in the AS-REQ message and the ticket used in a TGS-REQ. This discussion will focus on the etype selected for the AS-REQ authenticator.

Most implementations of Kerberos now use pre-authentication, meaning the client must include an authenticator in its AS-REQ message in order to receive an AS-REP with a ticket. A summary of a pre-authenticated AS exchange follows:



Authentication Service Exchange

In such a scenario, the list of valid etypes to be used for the authenticator is the union of those proposed by the client in the anonymous AS_REQ and by the server in the ERR PREAUTH REQUIRED message. The client is responsible for choosing the "best" among these, and uses it to protect the authenticator. Almost every default deployment of Kerberos will select a strong etype using this method.

This exchange is interesting for two reasons: First, when using password-based authentication, the AS-REQ authenticator is encrypted with a long-lived secret key that is derived from the user's password. Successful compromise of this key provides a password-equivalent credential for Kerberos purposes. Second, the majority of this exchange is performed without authentication, meaning there is no key to protect protocol exchanges. The active attacker is free to manipulate most of this sensitive exchange.

Downgrading Etypes

The obvious target for attacks against this message must be the negotiated etypes. In fact, an active attacker can compel the exchange to use any supported etype for the authenticator by inter-

cepting in one of two places: the unauthenticated AS-REQ or the PREAUTH REQUIRED error message.

Modifying the unauthenticated AS-REQ allows the attacker to "lie" to the KDC about the capabilities of the client. The KDC will then select etypes it supports from the attacker-supplied list of presumably weaker etypes, and reply to the client with a PREAUTH REQUIRED error. This error includes a list of agreed-upon etypes and includes etype-specific parameters required by the client - salts, optional parameters, etc. If the client supports anything from the attacker-influenced list the server provides, then the client will emit a poorly-protected authenticator using the etype of the attacker's choosing. If no supported client-supported etype is in this list, the protocol exchange will either retry or stop.

Modifying the PREAUTH REQUIRED error message provides similar capabilities for the attacker. In effect, the attacker "lies" to the client about the server's capabilities, but with two additional benefits: First, the attacker is able to compel the client to emit protected data using etypes that may not be supported by the server. Second, the attacker is able to control the optional data passed along with the etype, such as salt data. As with modifying the AS-REQ, both client and server must support the selected etype for an end-to-end authentication with a weak etype to be successful. If the etype the attacker chooses is not supported by the KDC, it will return an ETYPE NOT SUPPORTED error, and optionally can provide a list of supported etypes for the client to try once more.

BENEFITS OF AN ETYPE DOWNGRADE

An attacker that downgrades the etype of an AS exchange can accomplish one of two major attacks based on the authenticator emitted by the client:

First, the newest etypes, those that are based on the AES encryption algorithm, have been designed to resist efficient password grinding attacks. The computational cost of deriving a key from a user's password has been increased using PBKDF2, which repeatedly hashes the user's password material according to a configurable iteration count. This method, combined with a trustworthy salt, can be combined to make recovering sufficiently complex passwords computationally infeasible.

Downgrading to an older encryption type allows for established methods for password grinding to be successful once more. Frank O' Dwyer demonstrated the feasibility of such attacks even against an etype considered to be strong – RC4-HMAC. Furthermore, an attacker can set favorable values for those etypes that accept an optional salt and iteration count to aid in active or perhaps even precomputed password grinding attacks. The details of accomplishing such attacks will be heavily influenced by the Kerberos client implementation and configuration, and would make an interesting future research project.

Second, the attacker can compel the client and server to downgrade to an authenticator using an encryption algorithm whose key can be broken via brute force. The various DES-based etypes and the Windows-defined RC4 export-grade etypes are candidates for such attacks. Recovering the key used to encrypt the authenticator provides a long-lived password-equivalent credential for Kerberos purposes.

This attack could be achieved via an active downgrade of the AS-REQ exchange, capture of a weakly-protected authenticator, and an offline crack of the password. One company demonstrated a full exploration of the DES key space in 3 days using specialized hardware at Black Hat DC 2010, making such attacks quite possible and timely.

PROTECTING AGAINST ETYPE DOWNGRADE

Several implementation details of common Kerberos clients and servers make the feasibility of such attacks less general and more specific. However, the first and best way to prevent such attacks is to only support "strong" etypes.

Thankfully, recent versions of Kerberos implementations have taken steps towards eliminating the use of the weakest etypes. Documentation for MIT, Windows, and Solaris implementations of Kerberos indicate that DES etypes are either disabled in the latest versions or on their way out.

However, several plausible deployment patterns remain that are susceptible to downgrade:

- Windows 2008 / Windows Vista and previous all will emit and accept DES-based etypes
- Windows 7 clients will emit export-grade RC4, though authenticators of this type will not be accepted by any recent Windows DC in its default configuration.
- MIT Kerberos 1.7 and below all accept DES-based etypes
- Even when not enabled by default, DES is often enabled for interoperability purposes

The patch levels and configuration of the examples above can influence the supported etypes; it is always best to test a specific configuration in order to determine the feasibility of attacks.

Because of the expense and difficulty in staging such tests, the authors highly recommend that users take a conservative route to ensure protection. All Kerberos instances, clients and servers alike, should be configured to support only the AES-based etypes. MIT's implementation and those that are substantially similar have a straightforward mechanism for configuring etypes. Furthermore, Windows 2008 R2 / Windows 7 recently introduced a group policy setting to accomplish the same. The authors recommend consulting with the documentation for the specific implementation used to ensure proper configuration.

Smart Card Login and PKINIT Kerberos

Smart Card logon is accomplished through extensions to Kerberos that allow Public Key credentials (usually in an X.509 based PKI) to replace user passwords for the initial authentication between the user and the Key Distribution Center (KDC). The security of the overall logon process with a smart card depends on the verification policies for these certificates. iSEC Partners' research has identified a flaw in the default verification policies for clients connecting to Windows Server KDCs that renders these clients vulnerable to compromise.

Subject CNAME to DNS name matching (as is done for TLS) is the most familiar certificate verification policy. Kerberos does not use this algorithm because a client may not know what server to contact as its KDC, instead discovering it dynamically through NetBIOS broadcasts, DNS SRV record lookups and/or unauthenticated CLDAP searches. Since the name is not retrieved from a secure source, no subject name check is performed on the certificate presented by the KDC. If name verification is not generally used, how do clients verify the KDC certificate?

For all the common PKINIT implementations, all certificates involved must be issued by a specially designated trust root, specified either by a configuration file or stored in the Active Directory. After chaining to the specified trust root, MIT and Heimdal Kerberos follow the recommended algorithm of RFC 4556 by default, and check for the id-pkinit-PKPKdc EKU in the KDC's certificate. Windows KDCs do not include this EKU in their default configuration. Without it, how do clients identify a KDC? Windows' behavior is not fully documented, but a new group policy option in Windows Vista SP1 and Windows Server 2008 gives an important clue. In the text for the policy "Require Strict KDC Validation,"[1] 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[2], 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.

---

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

[2] 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

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, are there other certificates that have this EKU? 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. 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.

iSEC Partners' testing showed that this was actually possible. 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 of the "Web Server" and "Computer" templates issued by the enterprise authority, as if the AP-REP came from a genuine KDC.

PRACTICAL EXPLOITATION AND ELEVATION OF PRIVILEGE

For the MIT and Heimdal clients, the 'kinit' command with a smart card from the command line allows an easy elevation of privilege path using this flaw. An active network attacker can insert himself into the traffic flow, spoof the KDC and then 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. 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.

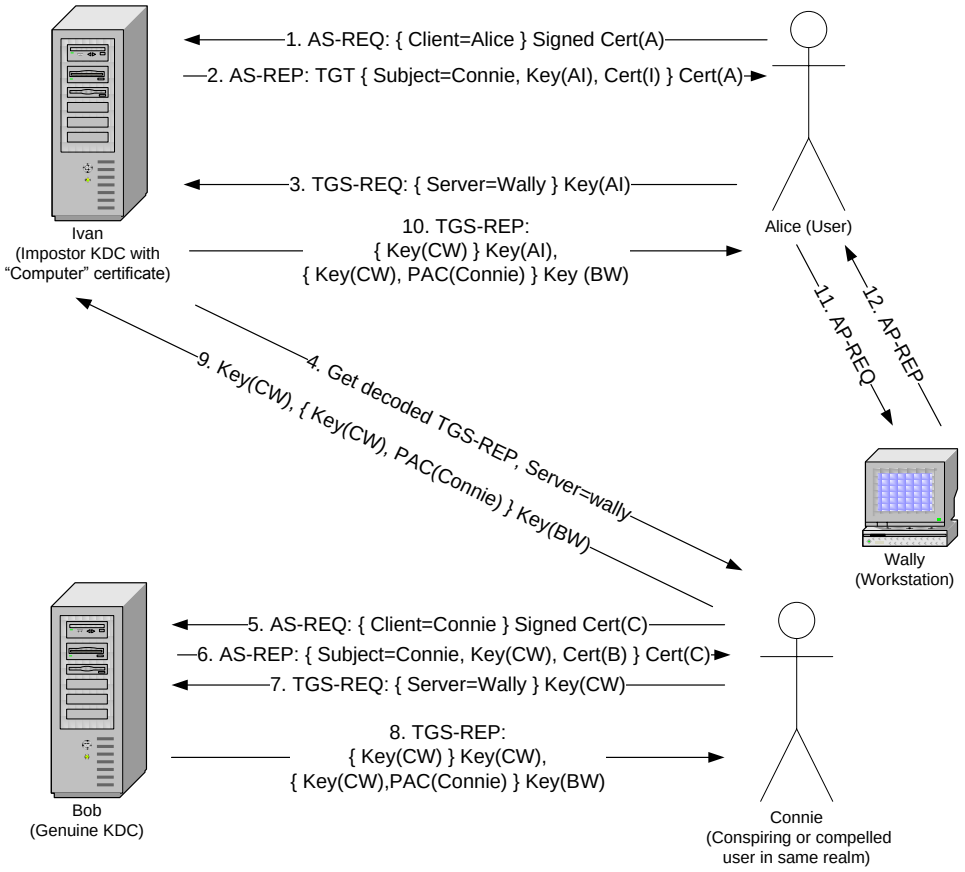
At this point the naïve attack fails. The impostor KDC can convince the user it is genuine with only a captured certificate, but to formulate a valid service ticket for the workstation computer account, it must know the secret shared between the computer and the genuine KDC. Public Key Kerberos provides 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.

Domain join is one scenario in which no workstation 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.

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. 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"[3] 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."[4] 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.

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

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.

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, (provided in the supplemental credential buffer of the PAC for smart card logins to support NTLM) 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.

DISCUSSION

This KDC impostor attack is identical in its implications to an attack formulated by Scedrov[5], et. al. in 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, and the attacks are hidden in the encrypted portions of the protocol. 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.

This attack can be mitigated by first deploying the new "Kerberos Authentication" template, available in Active Directory Certificate Services on Windows Server 2008 and later, to all Domain Controllers, then enabling the "Require Strict KDC Validation" policy on Vista SP1 and greater clients, and removing pkinit_require_eku=false from the configuration options of MIT and Heimdal clients connecting to Windows KDCs. Windows XP systems cannot be hardened against this attack at the time of writing. If upgrading from Windows XP is not an option, the only protection available is to disable the Computer template on the enterprise certification authority and carefully control the issuance of Web Server or other certificate templates with the Server Auth EKU to only highly-trusted servers.

---

[5] I. Cervesato, A. Jaggard, A. Scedrov, J. Tsay, C. Walstad, "Breaking and Fixing Public-Key Kerberos", May, 2007, http://www.qatar.cmu.edu/iliano/abstracts/papers/ic07.pdf

After initial authentication to the KDC, the ultimate goal of Kerberos is the ability to authenticate and communicate securely to a service such as LDAP, CIFS or RPC. The act of authenticating to a particular service with Kerberos is accomplished by the AP-REQ message. Although Kerberos can be used to provide very strong cryptographic protections for application protocols, many Kerberized services cut corners in their implementation, with the result that the AP-REQ message can be used in replay attacks against such poorly written services.

There are several types of replay attacks. Kerberos specifies that services should maintain a replay cache to detect messages that have been previously seen, and recommends that the cache be shared among all services operating as the same Kerberos Principal. Rather than a simple replay against the intended target service, a more interesting use of a replay is to capture and re-purpose an AP-REQ to authenticate to a different service, one which may provide or require different security guarantees. For example, is it possible to take the authenticator sent for a low-value cleartext protocol, and use it to authenticate to a high-value service?

This may be possible when multiple services operate with the same Kerberos account principal. Prior work on Windows by Tikkanen and Kasslin has shown that this is possible on Windows services. As most of the services on a given host all run under the same identity, it is possible to capture and re-purpose AP-REQs across protocols. How do such attacks work, generally, do they apply to Kerberized services on non-Windows platforms, and can they be prevented?

AP-REQ Construction and Protections

When a user authenticates to a particular service, one of the services provided by Kerberos is creation of a temporary session key known to both parties. The client uses this session key to construct the authenticator in its AP-REQ, including in it a number of pieces of information that can be used by the server to detect replay attacks:

> cksum: This field is intended to be a keyed checksum that can be used to tie the generic AP-REQ authenticator to application protocol data in the same packet. This might seem like what one wants in order to detect replays, but there are a few problems with how the field functions in practice. The field is occasionally left blank, leaving no proof that the program that sent the authenticator has any connection whatsoever to the data that is sent subsequently. Other services set the cksum to a magic number, which creates a service binding, serving only to prevent AP_REQs from other services from being replayed against services with no or a different magic number. Finally, even if the cksum is filled in, the AP-REQ is only sent during the initial authentication, so any protocol data after the first packet of a bidirectional protocol would be left unprotected.

ctime,cusec: These are timestamp fields. They give the receiving service an idea of the freshness of the AP-REQ. If an attacker makes a copy of the AP-REQ and sends it later, the time will be out of synch. This gives the attacker a limited window within which replay attacks are possible.

cname (encrypted portion of ticket): Includes the network address of the client, but this can be spoofed.

An additional protection against replay is provided by server side caching of a portion of the AP-REQ. If an AP-REQ is received twice, the second instance is assumed to be a replay attack.

The parameters of a successful replay attack are that the attacker must be actively intercepting and modifying traffic on the network. An attacker with ettercap or similar tools can copy the AP-REQ destined for a vulnerable service and use it to establish an authenticated session with the vulnerable service. The AP-REQ will appear to be fresh, will not appear in the replay cache, and if the integrity protection of the service is poor, no manner of replay will be detected, providing the attacker authenticated access.

PROTECTING AGAINST HIJACKING

In all cases, the first and best means to protect against hijacking a Kerberos authentication is to use the session key to protect the application data that follows. This cryptographic binding ensures that, even if an attacker is able to authenticate using a stolen AP-REQ, they are unable to formulate authenticated packets for the service.

For UNIX programmers, the use of this session key is often built into the client and protocol itself – Kerberized services often call libkerb directly to use the session key. Developers using PAM or GSSAPI are often purposefully abstracted from the use of Kerberos; like other abstracted authentication methods, it is best to ensure that the services assumed by your application are being fulfilled by the underlying API.

For Windows programmers, the use of the Kerberos session key is almost certainly abstracted from the programmer. Thankfully, every major protocol that uses Kerberos "transparently" provides some mechanism for providing cryptographic binding. Unfortunately, sometimes this requires actions by the developer, system administrator, or both. The following table summarizes the binding facilities available for each major protocol:

Protocol Name	Dev Binding	Admin Binding
LDAP	Use LDAP API to require Signing/Sealing . ADSI also supports the use of these attributes, but does not support forcing the use of Kerberos.	Specify "Require Signing" security policy for client and server.
RPC	Set binding on client to require Packet Integrity or better. Check the same in the security callback on the server.	N/A
DCOM	Set proxy blanket on client to require Packet Integrity or better. Configure service to require the same authorization level.	Set machine-wide default and per-App DCOM authorization levels using the Component Services MMC plug-in.
SMB / Named Pipe	N/A	Specify "Digitally sign communications always" for network clients and servers
HTTPS	Transparent in most applications	Enable Extended Protection registry keys and on web applications

MUTUAL AUTHENTICATION AND SPNS

Mutual authentication is one of the cornerstone services provided by Kerberos (in contrast to NTLM or the most typical uses of TLS). In the most general terms, mutual authentication means that the client and server authenticate each other at the completion of an exchange. As the discussion of replay attacks may have intimated, the technical details are a bit more complex. At best, authentication of the service by the client in Kerberos means, "the identity mapped to this SPN."

How do self-organizing services get their SPN in a secure manner? This is a difficult question to answer. It is much easier to list the ways in which this is commonly done insecurely.

- Ask the attacker what their SPN is
- Call an API that asks the attacker what their SPN is (e.g. rpc_mgmt_inq_server_princ_name)
- Ask DNS what the attacker's SPN is (SRV record lookup)
- Pull the attacker's SPN out of a service definition served by…the attacker (e.g. WSDL <spn> or <upn> elements)
- Fail to set a proper, fully-qualified, SPN
- Fail to set any SPN whatsoever

A well-organized system may provide for secure methods of service discovery, but requirements are often application and environment-specific. The simple summary is that the SPN must be determined in a secure manner, and that protocols must use the negotiated session key to actually assure that they are talking to the intended server.

Some developers rely on the "Mutual Authentication" semantics in the Kerberos protocol itself, but this feature is surprisingly useless. Either a session key is used to provide cryptographic protections for every packet sent in each direction – in which case mutual authentication is implicit – or it is not, in which case the mutual authentication portion of the Kerberos protocol cannot protect against an active attacker unless the entire protocol payload plus the AP-REQ and AP-REP can be guaranteed to fit in single packets..

Conclusion

Although Kerberos is one of the oldest cryptographic protocols still in wide use for mainstream, modern, distributed systems, it has still proven to be a fruitful area for security research, and the possibilities for further work are yet to be exhausted, especially considering the wide variety of clients, versions and configurations commonly deployed today.

Secure deployment of Kerberos requires careful attention to detail on the part of developers and administrators. Look for elaborated whitepapers containing specific, audience-appropriate guidance on the topics discussed here on the "Independent Research" section of iSEC Partners' website, https://www.isecpartners.com/.

Related Prior Work

Kasslin, K. and Tikkanen, A, “Attacks on Kerberos V”, http://users.tkk.fi/autikkan/kerberos/, 2000-2004

O’Dwyer, Frank, “Feasibility of attacking Windows 2000 Kerberos Passwords.” http://www.frankodwyer.com/blog/?page_id=183, 2002

S. M. Bellovin, M. Merritt., “Limitations of the Kerberos Authentication System.”, 1990, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.6087&rep=rep1&type=pdf

I. Cervesato, A. Jaggard, A. Scedrov, J. Tsay, C. Walstad, “Breaking and Fixing Public-Key Kerberos”, May, 2007, http://www.qatar.cmu.edu/iliano/abstracts/papers/ic07.pdf

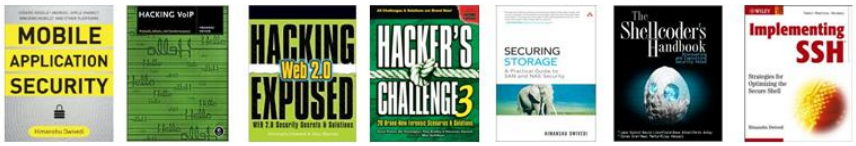
Hur, Mark. “FPGA Cluster Demonstrates Massively Parallel, Hardware-Accelerated DES Cracking.” http://www.picocomputing.com/pdf/PR_Pico_DES_BH_Jan_29_2010.pdf, January 2010

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
- OnDemand Web Application Scanning (Automated/Manual)

Published Books



Notable Presentations

RSA CONFERENCE
Black Hat
web2.0 EXPO
InfoWorld
COMPUTERWORLD STORAGE NETWORKING WORLD SNIA

Whitepaper, Tools, Advisories, & SDL Products

- 12 Published Whitepapers
  - Including the first whitepaper on CSRF
- 37 Free Security Tools
  - Application, Infrastructure, Mobile, VoIP, & Storage
- 9 Advisories
  - Including products from 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)