Kerberos (protocol)

Kerberos (/ˈkɜːrbərɒs/) is a computer-network authentication protocol that works on the basis of *tickets* to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. Its designers aimed it primarily at a client–server model, and it provides mutual authentication—both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks.

Kerberos	
Developer(s)	Massachusetts Institute of Technology
Stable release	Version 5, Release 1.19.3 / 14 March 2022[1]
Written in	C
Operating system	Cross-platform
Size	8512k (source code)
Type	Authentication protocol
Website	web.mit.edu/kerberos/ (https://web.mit.edu/kerberos/)

Kerberos builds on symmetric-key cryptography and requires a trusted third party, and optionally may use public-key cryptography during certain phases of authentication.[2] Kerberos uses UDP port 88 by default.

The protocol was named after the character *Kerberos* (or *Cerberus*) from Greek mythology, the ferocious three-headed guard dog of Hades.

History and development

Massachusetts Institute of Technology (MIT) developed Kerberos to protect network services provided by Project Athena.[3][4] The protocol is based on the earlier Needham–Schroeder symmetric-key protocol. Several versions of the protocol exist; versions 1–3 occurred only internally at MIT.

Kerberos version 4 was primarily designed by Steve Miller and Clifford Neuman.[5] Published in the late 1980s, version 4 was also targeted at Project Athena.

Neuman and John Kohl published version 5 in 1993 with the intention of overcoming existing limitations and security problems. Version 5 appeared as RFC 1510, which was then made obsolete by RFC 4120 in 2005.

Authorities in the United States classified Kerberos as "Auxiliary Military Equipment" on the US Munitions List and banned its export because it used the Data Encryption Standard (DES) encryption algorithm (with 56-bit keys). A Kerberos 4 implementation developed at the Royal Institute of Technology in Sweden named KTH-KRB (rebranded to Heimdal at version 5) made the system available outside the US before the US changed its cryptography export regulations (around 2000). The Swedish implementation was based on a limited version called eBones. eBones was based on the exported MIT Bones release (stripped of both the encryption functions and the calls to them) based on version Kerberos 4 patch-level 9.

In 2005, the Internet Engineering Task Force (IETF) Kerberos working group updated specifications. Updates included:

- Encryption and Checksum Specifications (RFC 3961).
- Advanced Encryption Standard (AES) Encryption for Kerberos 5 (RFC 3962).
- A new edition of the Kerberos V5 specification "The Kerberos Network Authentication Service (V5)" (RFC 4120). This version obsoletes RFC 1510, clarifies aspects of the protocol and intended use in a more detailed and clearer explanation.
- A new edition of the Generic Security Services Application Program Interface (GSS-API) specification "The Kerberos Version 5 Generic Security Service Application Program Interface

(GSS-API) Mechanism: Version 2" (RFC 4121).

MIT makes an implementation of Kerberos freely available, under copyright permissions similar to those used for BSD. In 2007, MIT formed the Kerberos Consortium to foster continued development. Founding sponsors include vendors such as Oracle, Apple Inc., Google, Microsoft, Centrify Corporation and TeamF1 Inc., and academic institutions such as the Royal Institute of Technology in Sweden, Stanford University, MIT, and vendors such as CyberSafe offering commercially supported versions.

Microsoft Windows

Windows 2000 and later versions use Kerberos as their default authentication method.[6] Some Microsoft additions to the Kerberos suite of protocols are documented in RFC 3244 "Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols". RFC 4757 documents Microsoft's use of the RC4 cipher. While Microsoft uses and extends the Kerberos protocol, it does not use the MIT software.

Kerberos is used as the preferred authentication method: in general, joining a client to a Windows domain means enabling Kerberos as the default protocol for authentications from that client to services in the Windows domain and all domains with trust relationships to that domain.[6]

In contrast, when either client or server or both are not joined to a domain (or not part of the same trusted domain environment), Windows will instead use NTLM for authentication between client and server.[6]

Internet web applications can enforce Kerberos as an authentication method for domain-joined clients by using APIs provided under SSPI.

Microsoft Windows and Windows Server include `setspn`, a command-line utility that can be used to read, modify, or delete the Service Principal Names (SPN) for an Active Directory service account.[7][8]

Unix and other operating systems

Many Unix-like operating systems, including FreeBSD, OpenBSD, Apple's macOS, Red Hat Enterprise Linux, Oracle's Solaris, IBM's AIX, HP-UX and others, include software for Kerberos authentication of users or services. A variety of non-Unix like operating systems such as z/OS,

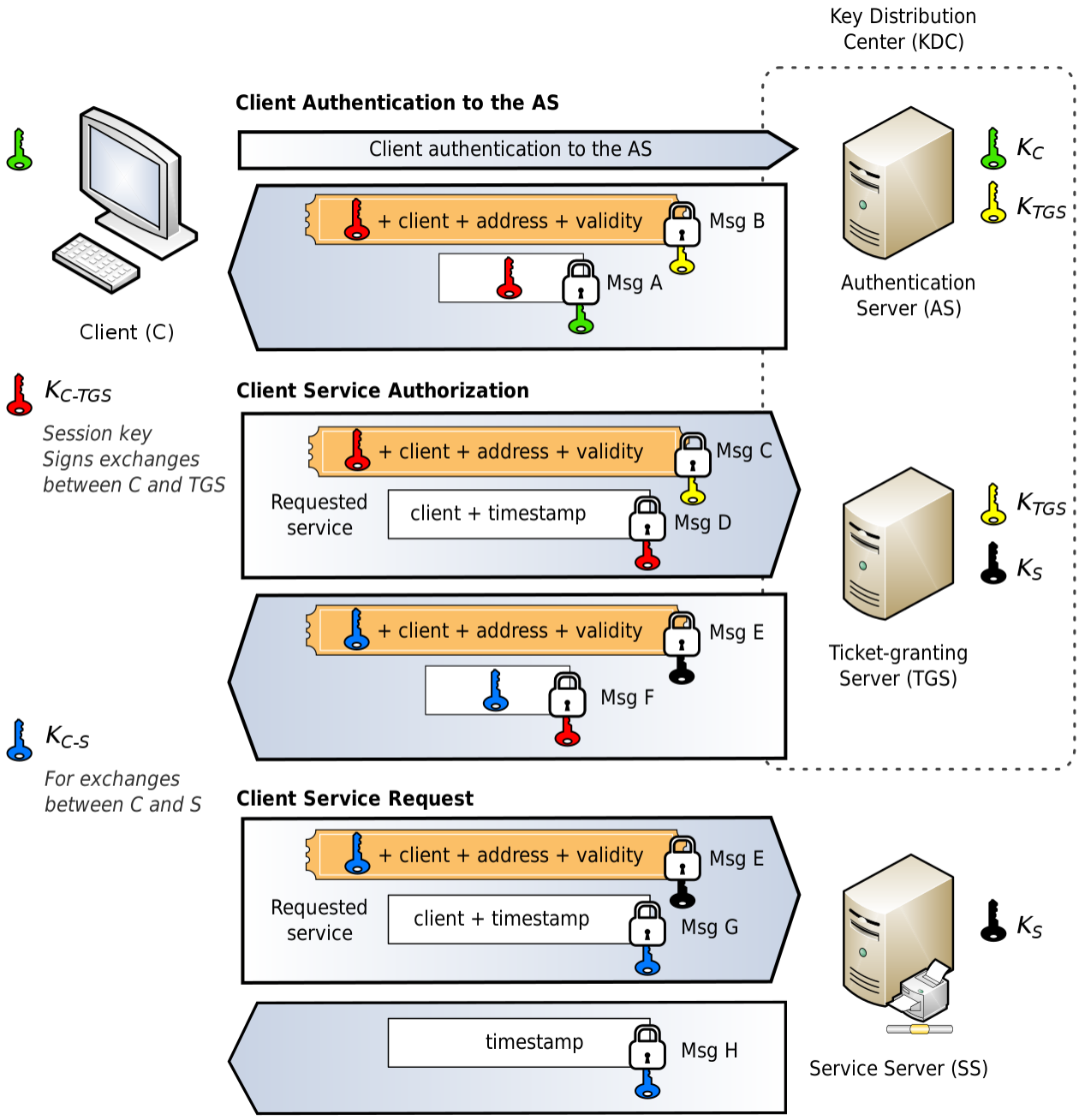
IBM i and OpenVMS also feature Kerberos support. Embedded implementation of the Kerberos V authentication protocol for client agents and network services running on embedded platforms is also available from companies.

Protocol

Description

The client authenticates itself to the **Authentication Server (AS)** which forwards the username to a **key distribution center (KDC)**. The KDC issues a **ticket-granting ticket (TGT)**, which is time stamped and encrypts it using the **ticket-granting service's (TGS)** secret key and returns the encrypted result to the user's workstation. This is done infrequently, typically at user logon; the TGT expires at some point although it may be transparently renewed by the user's session manager while they are logged in.

When the client needs to communicate with a service on another node (a "principal", in Kerberos parlance), the client sends the TGT to the TGS, which usually shares the same host as the KDC. The service must have already been registered with the TGS with a **Service Principal Name (SPN)**. The client uses the SPN to request access to this service. After verifying that the TGT is valid and that the user is permitted to access the requested service, the TGS issues ticket and session keys to the client. The client then sends the ticket to the **service server (SS)** along with its service request.



Kerberos negotiations

The protocol is described in detail below.

User Client-based Login without Kerberos

1. A user enters a username and password on the client machine(s). Other credential mechanisms like pkinit (RFC 4556) allow for the use of public keys in place of a password.

The client transforms the password into the key of a symmetric cipher. This either uses the built-in key scheduling, or a one-way hash, depending on the cipher-suite used.

2. The server receives the username and symmetric cipher and compares it with the data from database. Login was a success if the cipher matches the cipher that is stored for the user.

Client Authentication

1. The client sends a cleartext message of the user ID to the AS (Authentication Server) requesting services on behalf of the user. (Note: Neither the secret key nor the password is sent to the AS.)
2. The AS checks to see whether the client is in its database. If it is, the AS generates the secret key by hashing the password of the user found at the database (e.g., Active Directory in Windows Server) and sends back the following two messages to the client:
   - Message A: *Client/TGS Session Key* encrypted using the secret key of the client/user.
   - Message B: *Ticket-Granting-Ticket* (TGT, which includes the client ID, client network address, ticket validity period, and the *Client/TGS Session Key*) encrypted using the secret key of the TGS.
3. Once the client receives messages A and B, it attempts to decrypt message A with the secret key generated from the password entered by the user. If the user entered password does not match the password in the AS database, the client's secret key will be different and thus unable to decrypt message A. With a valid password and secret key the client decrypts message A to obtain the *Client/TGS Session Key*. This session key is used for further communications with the TGS. (Note: The client cannot decrypt Message B, as it is encrypted using TGS's secret key.) At this point, the client has enough information to authenticate itself to the TGS.

Client Service Authorization

1. When requesting services, the client sends the following messages to the TGS:
   - Message C: Composed of the message B (the encrypted TGT using the TGS secret key) and the ID of the requested service.
   - Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the *Client/TGS Session Key*.
2. Upon receiving messages C and D, the TGS retrieves message B out of message C. It decrypts message B using the TGS secret key. This gives it the *Client/TGS Session Key* and the client ID (both are in the TGT). Using this *Client/TGS Session Key*, the TGS decrypts

message D (Authenticator) and compares the client IDs from messages B and D; if they match, the server sends the following two messages to the client:

- Message E: *Client-to-server ticket* (which includes the client ID, client network address, validity period, and *Client/Server Session Key*) encrypted using the service's secret key.
- Message F: *Client/Server Session Key* encrypted with the *Client/TGS Session Key*.

Client Service Request

1. Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the Service Server (SS). The client connects to the SS and sends the following two messages:
   - Message E: From the previous step (the *Client-to-server ticket*, encrypted using service's secret key).
   - Message G: A new Authenticator, which includes the client ID, timestamp and is encrypted using *Client/Server Session Key*.
2. The SS decrypts the ticket (message E) using its own secret key to retrieve the *Client/Server Session Key*. Using the sessions key, SS decrypts the Authenticator and compares client ID from messages E and G, if they match server sends the following message to the client to confirm its true identity and willingness to serve the client:
   - Message H: The timestamp found in client's Authenticator (plus 1 in version 4, but not necessary in version 5[9][10]), encrypted using the *Client/Server Session Key*.
3. The client decrypts the confirmation (message H) using the *Client/Server Session Key* and checks whether the timestamp is correct. If so, then the client can trust the server and can start issuing service requests to the server.
4. The server provides the requested services to the client.

Drawbacks and limitations

- Kerberos has strict time requirements, which means that the clocks of the involved hosts must be synchronized within configured limits. The tickets have a time availability period, and if the host clock is not synchronized with the Kerberos server clock, the authentication will fail. The default configuration per MIT (http://web.mit.edu/Kerberos/krb5-1.5/krb5-1.5.4/doc/krb5-admin/Clock-Skew.html) requires that clock times be no more than five minutes apart. In practice, Network Time Protocol daemons are usually used to keep the host clocks synchronized. Note that some servers (Microsoft's implementation being one of them) may return a KRB_AP_ERR_SKEW result containing the encrypted server time if both clocks have

an offset greater than the configured maximum value. In that case, the client could retry by calculating the time using the provided server time to find the offset. This behavior is documented in RFC 4430 (https://datatracker.ietf.org/doc/html/rfc4430) .

- The administration protocol is not standardized and differs between server implementations. Password changes are described in RFC 3244.
- In case of symmetric cryptography adoption (Kerberos can work using symmetric or asymmetric (public-key) cryptography), since all authentications are controlled by a centralized key distribution center (KDC), compromise of this authentication infrastructure will allow an attacker to impersonate any user.
- Each network service that requires a different host name will need its own set of Kerberos keys. This complicates virtual hosting and clusters.
- Kerberos requires user accounts and services to have a trusted relationship to the Kerberos token server.
- The required client trust makes creating staged environments (e.g., separate domains for test environment, pre-production environment and production environment) difficult: Either domain trust relationships need to be created that prevent a strict separation of environment domains, or additional user clients need to be provided for each environment.

Vulnerabilities

The Data Encryption Standard (DES) cipher can be used in combination with Kerberos, but is no longer an Internet standard because it is weak.[11] Security vulnerabilities exist in many legacy products that implement Kerberos because they have not been updated to use newer ciphers like AES instead of DES.

In November 2014, Microsoft released a patch (MS14-068) to rectify an exploitable vulnerability in Windows implementation of the Kerberos Key Distribution Center (KDC).[12] The vulnerability purportedly allows users to "elevate" (and abuse) their privileges, up to Domain level.

See also

- Single sign-on
- Identity management
- SPNEGO

- S/Key
- Secure remote password protocol (SRP)
- Generic Security Services Application Program Interface (GSS-API)
- Host Identity Protocol (HIP)
- List of single sign-on implementations

References

1. *"Kerberos 5 Release 1.19.3" (https://web.mit.edu/kerberos/krb5-1.19/)* .
2. *RFC 4556, abstract.*
3. *Steiner, Jennifer G.; Geer, Daniel E. (21 July 1988). Network Services in the Athena Environment. Proceedings of the Winter 1988 Usenix Conference. CiteSeerX 10.1.1.31.8727 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.8727)* .
4. *Elizabeth D. Zwicky; Simon Cooper; D. Brent (26 Jun 2000). Building Internet Firewalls: Internet and Web Security (https://archive.org/details/buildinginternet00zwic)* . *O'Reilly. ISBN 9781565928718.*
5. *Steiner, Jennifer G.; Neuman, Clifford; Schiller, Jeffrey I. (February 1988). Kerberos: An authentication service for open network systems. Proceedings of the Winter 1988 USENIX Conference. CiteSeerX 10.1.1.112.9002 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.9002)* . *S2CID 222257682 (https://api.semanticscholar.org/CorpusID:222257682)* .
6. *"What Is Kerberos Authentication?" (https://technet.microsoft.com/pt-br/library/cc780469(v=ws.10).aspx)* . *Microsoft TechNet. Archived (https://web.archive.org/web/20161220084434/https://technet.microsoft.com/pt-br/library/cc780469(v=ws.10).aspx)* *from the original on 2016-12-20.*
7. *Setspn - Windows CMD - SS64.com (https://ss64.com/nt/setspn.html)*
8. *Setspn | Microsoft Docs (https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/cc731241(v=ws.11))*
9. *C., Neuman; J., Kohl. "The Kerberos Network Authentication Service (V5)" (https://tools.ietf.org/html/rfc1510#section-3.2.4)* . *Archived (https://web.archive.org/web/20160821221402/https://tools.ietf.org/html/rfc1510#section-3.2.4)* *from the original on 2016-08-21.*
10. *Clifford, Neuman; Sam, Hartman; Tom, Yu; Kenneth, Raeburn. "The Kerberos Network Authentication Service (V5)" (https://tools.ietf.org/html/rfc4120#section-3.2.4)* . *Archived (https://web.archive.org/web/20160821232023/https://tools.ietf.org/html/rfc4120#section-3.2.4)* *from the original on 2016-08-21.*

11. *Tom, Yu; Love, Astrand.* *"Deprecate DES, RC4-HMAC-EXP, and Other Weak Cryptographic Algorithms in Kerberos" (https://tools.ietf.org/html/rfc6649)* *. Archived (https://web.archive.org/web/20151027034313/http://tools.ietf.org/html/rfc6649)* *from the original on 2015-10-27.*

12. *Seltzer, Larry.* *"Details emerge on Windows Kerberos vulnerability - ZDNet" (https://www.zdnet.com/details-emerge-on-windows-kerberos-vulnerability-7000035976/)* *. ZDNet. Archived (https://web.archive.org/web/20141121173014/http://www.zdnet.com/details-emerge-on-windows-kerberos-vulnerability-7000035976/)* *from the original on 2014-11-21.*

General

- Lynn Root (May 30, 2013) (2 April 2013). "Explain like I'm 5: Kerberos" (http://www.roguelynn.com/words/explain-like-im-5-kerberos) . *Blog of Lynn Root*.
- Microsoft TechNet 2017. "Basic Concepts for the Kerberos Protocol" (https://technet.microsoft.com/en-us/library/cc961976.aspx) . *MSDN Library*.
- Resource Kit Team. "Microsoft Kerberos (Windows)" (http://msdn.microsoft.com/en-us/library/aa378747(VS.85).aspx) . *MSDN Library*.
- B. Clifford Neuman; Theodore Ts'o (September 1994). "Kerberos: An Authentication Service for Computer Networks" (http://gost.isi.edu/publications/kerberos-neuman-tso.html) . *IEEE Communications*. **32** (9): 33–8. doi:10.1109/35.312841 (https://doi.org/10.1109%2F35.312841) . S2CID 45031265 (https://api.semanticscholar.org/CorpusID:45031265) .
- Kohl, John T.; Neuman, B. Clifford; Ts'o, Theodore Y. (1994). "The Evolution of the Kerberos Authentication System". In Brazier, F. M. T.; Johansen, D (eds.). *Distributed open systems*. IEEE Computer Society Press. pp. 78–94. CiteSeerX 10.1.1.120.944 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.120.944) . ISBN 978-0-8186-4292-0. OCLC 1191406172 (https://www.worldcat.org/oclc/1191406172) .
- "Kerberos Overview: An Authentication Service for Open Network Systems" (http://www.cisco.com/en/US/tech/tk59/technologies_white_paper09186a00800941b2.shtml) . Cisco Systems. 19 January 2006. Retrieved 15 August 2012.
- "How Kerberos Authentication Works" (http://learn-networking.com/network-security/how-kerberos-authentication-works) . learn-networking.com. 28 January 2008. Retrieved 15 August 2012.
- "What is Kerberos Authentication?: Logon and Authentication" (https://technet.microsoft.com/pt-br/library/cc780469(v=ws.10).aspx) . Microsoft TechNet. Retrieved 7 December 2016.

RFCs

- RFC 1510 (https://datatracker.ietf.org/doc/html/rfc1510) The Kerberos Network Authentication Service (V5) [Obsolete]
- RFC 1964 (https://datatracker.ietf.org/doc/html/rfc1964) The Kerberos Version 5 GSS-API Mechanism

- RFC 3961 (https://datatracker.ietf.org/doc/html/rfc3961) Encryption and Checksum Specifications for Kerberos 5
- RFC 3962 (https://datatracker.ietf.org/doc/html/rfc3962) Advanced Encryption Standard (AES) Encryption for Kerberos 5
- RFC 4120 (https://datatracker.ietf.org/doc/html/rfc4120) The Kerberos Network Authentication Service (V5) [Current]
- RFC 4121 (https://datatracker.ietf.org/doc/html/rfc4121) The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2
- RFC 4537 (https://datatracker.ietf.org/doc/html/rfc4537) Kerberos Cryptosystem Negotiation Extension
- RFC 4556 (https://datatracker.ietf.org/doc/html/rfc4556) Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)
- RFC 4557 (https://datatracker.ietf.org/doc/html/rfc4557) Online Certificate Status Protocol (OCSP) Support for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)
- RFC 4757 (https://datatracker.ietf.org/doc/html/rfc4757) The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows [Obsolete]
- RFC 5021 (https://datatracker.ietf.org/doc/html/rfc5021) Extended Kerberos Version 5 Key Distribution Center (KDC) Exchanges over TCP
- RFC 5349 (https://datatracker.ietf.org/doc/html/rfc5349) Elliptic Curve Cryptography (ECC) Support for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)
- RFC 5868 (https://datatracker.ietf.org/doc/html/rfc5868) Problem Statement on the Cross-Realm Operation of Kerberos
- RFC 5896 (https://datatracker.ietf.org/doc/html/rfc5896) Generic Security Service Application Program Interface (GSS-API): Delegate if Approved by Policy
- RFC 6111 (https://datatracker.ietf.org/doc/html/rfc6111) Additional Kerberos Naming Constraints
- RFC 6112 (https://datatracker.ietf.org/doc/html/rfc6112) Anonymity Support for Kerberos
- RFC 6113 (https://datatracker.ietf.org/doc/html/rfc6113) A Generalized Framework for Kerberos Pre-Authentication

- RFC 6251 (https://datatracker.ietf.org/doc/html/rfc6251) Using Kerberos Version 5 over the Transport Layer Security (TLS) Protocol
- RFC 6448 (https://datatracker.ietf.org/doc/html/rfc6448) The Unencrypted Form of Kerberos 5 KRB-CRED Message
- RFC 6542 (https://datatracker.ietf.org/doc/html/rfc6542) Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Channel Binding Hash Agility
- RFC 6560 (https://datatracker.ietf.org/doc/html/rfc6560) One-Time Password (OTP) Pre-Authentication
- RFC 6649 (https://datatracker.ietf.org/doc/html/rfc6649) Deprecate DES, RC4-HMAC-EXP, and Other Weak Cryptographic Algorithms in Kerberos
- RFC 6784 (https://datatracker.ietf.org/doc/html/rfc6784) Kerberos Options for DHCPv6
- RFC 6803 (https://datatracker.ietf.org/doc/html/rfc6803) Camellia Encryption for Kerberos 5
- RFC 6806 (https://datatracker.ietf.org/doc/html/rfc6806) Kerberos Principal Name Canonicalization and Cross-Realm Referrals
- RFC 6880 (https://datatracker.ietf.org/doc/html/rfc6880) An Information Model for Kerberos Version 5

Further reading

- "Novell Inc's Comment to the Proposed Settlement between Microsoft and the Department of Justice, pursuant to the Tunney Act" (https://www.usdoj.gov/atr/cases/ms_tuncom/major/mtc-00029523.htm) . *Civil Action No. 98-1232 (CKK): United States of America v. Microsoft Corporation*. Department of Justice. 29 January 2002. Retrieved 15 August 2012.
- Bryant, Bill (February 1988). "Designing an Authentication System: A Dialogue in Four Scenes" (http://web.mit.edu/kerberos/www/dialogue.html) . *Humorous play concerning how the design of Kerberos evolved*. MIT.
- Hornstein, Ken (18 August 2000). "Kerberos FAQ, v2.0" (https://web.archive.org/web/20021203013358/http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html) . Secretary of Navy. Archived from the original (http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html) on 3 December 2002. Retrieved 15 August 2012.
- Bellovin, S. M.; Merritt, M. (1 October 1990). "Limitations of the Kerberos authentication system". *ACM SIGCOMM Computer Communication Review*. **20** (5): 119–132. doi:10.1145/381906.381946 (https://doi.o

rg/10.1145%2F381906.381946) . S2CID 8014806 (https://api.semanticscholar.org/CorpusID:8014806) .

- Neuman, B.C.; Ts'o, T. (September 1994). "Kerberos: an authentication service for computer networks". *IEEE Communications Magazine*. **32** (9): 33–38. doi:10.1109/35.312841 (https://doi.org/10.1109%2F35.312841) . S2CID 45031265 (https://api.semanticscholar.org/CorpusID:45031265) .
- Bella, Giampaolo; Paulson, Lawrence C. (1998). "Kerberos Version IV: Inductive analysis of the secrecy goals". *Computer Security — ESORICS 98*. Lecture Notes in Computer Science. Vol. 1485. pp. 361–375. doi:10.1007/BFb0055875 (https://doi.org/10.1007%2FBFb0055875) . ISBN 978-3-540-65004-1.
- Abdelmajid, N.T.; Hossain, M.A.; Shepherd, S.; Mahmoud, K. (2010). "Improved Kerberos Security Protocol Evaluation using Modified BAN Logic". *2010 10th IEEE International Conference on Computer and Information Technology*. pp. 1610–1615. doi:10.1109/CIT.2010.285 (https://doi.org/10.1109%2FCIT.2010.285) . ISBN 978-1-4244-7547-6. S2CID 6246388 (https://api.semanticscholar.org/CorpusID:6246388) .

External links

Wikimedia Commons has media related to ***Kerberos***.

- Kerberos Consortium (http://www.kerberos.org/)
- Kerberos page (http://web.mit.edu/kerberos/) at MIT website
- Kerberos Working Group (https://web.archive.org/web/20040707075602/http://www.ietf.org/html.charters/krb-wg-charter.html) at IETF website
- Kerberos Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/kerberos/kerberos-sequence-diagram.pdf) Archived (https://web.archive.org/web/20150326142031/http://eventhelix.com/RealtimeMantra/Networking/kerberos/kerberos-sequence-diagram.pdf) 2015-03-26 at the Wayback Machine
- Heimdal/Kerberos implementation (https://github.com/heimdal/heimdal/wiki)

title=Kerberos_(protocol)&oldid=1093303076"

Last edited 28 days ago **by** David Eppstein