目次

概要

オープン ネットワーク コンピューティング環境では、ワークステーションのユーザをネットワーク サービスに対して正確に識別する上で、そのワークステーションを信頼できない場合があります。 Kerberos では、ユーザの身元を確認するために信頼されたサード パーティ認証サービスを

使用するという、代替のアプローチが提供されます。 このドキュメントでは、MIT の Project Athena に実装された Kerberos 認証モデルの概要を説明しています。 クライアント、サーバ、および Kerberos で認証を確立するために使用されるプロトコルを説明しています。 また、必要となるデータベースの管理と複製についても取り上げています。 ユーザ、プログラマ、および管理者の視点での Kerberos について説明しています。 最後に、大規模な Athena 環境における Kerberos の役割と、ユーザ認証用に Kerberos が現在使用されているアプリケーションのリストを紹介しています。 Kerberos の既存のアプリケーションとの統合の事例研究として、Sun のネットワーク ファイル システムへの Kerberos 認証の付加について説明しています。

Kerberos の設計者

- Jennifer G. Steiner (Project Athena, Massachusetts Institute of Technology, Cambridge, MA 02139、steiner@ATHENA.MIT.EDU)
- Clifford Neuman (Department of Computer Science, FR-35, University of Washington, Seattle, WA 98195, bcn@CS.WASHINGTON.EDU) Clifford Neuman は Kerberos の設計および初期実装フェーズで Project Athena スタッフのメンバーでした。
- Jeffrey I. Schiller (Project Athena, Massachusetts Institute of Technology, Cambridge, MA 02139、jis@ATHENA.MIT.EDU)

Kerberos の概要

この記事では、Miller と Neuman がオープン ネットワーク コンピューティングのために設計した認証システムである Kerberos の概要を説明しています。 また、このシステムを使用した MIT の Project Athena について説明します。 「Kerberos が必要とされる背景」セクションでは、オープン ネットワークで新しい認証モデルが必要とされる理由とその要件について説明しています。 「Kerberos について」 セクションでは、Kerberos ソフトウェアのコンポーネントを一覧で掲載し、認証サービスの提供でのこれらの相互対話方式を説明しています。 「Kerberos で使用される名前」セクションでは、Kerberos の名前付けスキームについて説明しています。

「Kerberos の動作の仕組み」では、Kerberos 認証のビルディング ブロックであるチケットとオーセンティケータを取り上げています。 これに続いて、2 つの認証プロトコル、 つまり (ログインに類似した) Kerberos へのユーザの初期認証、およびネットワーク サービスの潜在的コンシューマと潜在的プロデューサの相互認証のプロトコルを説明しています。

Kerberos にはそのクライアントに関する情報のデータベースが必要で、 「「Kerberos データベース」セクションでは、データベース、その管理、およびその修正のためのプロトコルについて説明しています。 「外部の視点からの Kerberos」セクションでは、Kerberos のユーザ、アプリケーション プログラマ、および管理者への Kerberos インターフェイスについて説明しています。 「大規模な Kerberos 環境」セクションでは、Project Athena の Kerberos がその他の Athena 環境にどのように適合するのかを説明しています。 また、異なる Kerberos 認証ドメイン間やレルム間での相互対話 (ここでは Project Athena の Kerberos と MIT の Laboratory for Computer Science で稼働する Kerberos の間の相互関係) についても説明しています。

「Kerberos の未解決の問題」セクションでは、まだ解決していない問題について言及しています。 最後のセクションでは、Project Athena における Kerberos の現状を説明しています。 「付録」では、リモート ファイル システムへのアクセスを希望するユーザを認証するために、ネットワーク ファイル サービスへ Kerberos を適用する方法についての詳細を説明しています。

Kerberos のコンセプト

このドキュメント全体にわたって、不明確な用語、読者にとって新しい用語、または他のドキュメントでは異なる用法で使用される用語などが使用されている可能性があります。 これらの用語については、このドキュメントでの使用方法を次に説明しています。

*ユーザ、クライアント、サーバか。*ユーザによって、プログラムかサービスを利用する人間を意味します。 クライアントも何らかの対象を使用しますが、これは必ずしも人であるとは限らず、プログラムである可能性もあります。 多くの場合、ネットワーク アプリケーションは、 あるマシン上で稼働してリモート サービスを要求するプログラムと、リモート マシン上で稼働してサービスを実行する別のプログラムの 2 つの部分から構成されます。 これらをそれぞれアプリケーションのクライアント側とサーバ側と呼びます。 多くの場合、クライアントがユーザに代わってサーバへの通信を行います。

Kerberos システムが使用される各エンティティは、ユーザであろうとネットワーク サーバであろうと、Kerberos サービスが使用されるため、ある意味ではクライアントです。 したがって、他のサービスのクライアントから Kerberos のクライアントを区別するために、このようなエンティティを示すためにプリンシパルという用語を使用します。 Kerberos のプリンシパルは、ユーザまたはサーバのどちらである場合もあります（後のセクションでは Kerberos のプリンシパルの名前付けについて説明しています）。

*サーバ対保守して下さいか。*サービスをように実行されたいくつかの操作の抽象仕様利用します。 これらの処理が実行されるプロセスは、サーバと呼ばれます。 ある特定の時刻で、特定のサービスを実行している複数のサーバ（通常は異なるマシン上で稼働）が存在することがあります。 たとえば、Athena では、各タイムシェアリング マシン上で稼働する 1 台の BSD UNIX rlogin サーバが存在します。

*キー、プライベートキー、パスワードか。*Kerberos 使用プライベートキー暗号化。 各 Kerberos プリンシパルには、そのプリンシパルと Kerberos にだけ認識される大きい数字、つまりその秘密鍵が割り当てられます。 ユーザの場合、秘密鍵はユーザのパスワードに適用された単方向機能の結果です。 鍵は秘密鍵の省略表記として使用します。

*資格情報か。*残念ながら、このワードに Sunネットワークファイルシステムおよび Kerberosシステム両方のための特別な意味があります。 NFS クレデンシャルまたは Kerberos クレデンシャルのどちらを意味しているのかについては明示的に言及しますが、それ以外の場合、この用語は英語での通常の意味で使用されます。

*マスターおよびスレーブか。*複数のマシンの Kerberos認証 ソフトウェアを実行することは可能性のあるです。 ただし、常に Kerberos データベースの決定的なコピーが 1 つだけ存在します。 このデータベースが収容されるマシンは、マスター マシン（または単にマスター）と呼ばれます。 他のマシンには、Kerberos データベースの読み取り専用コピーが存在することがあり、これらはスレーブと呼ばれます。

Kerberos が必要とされる背景

ネットワーク化されていないパーソナル コンピューティング環境では、パーソナル コンピュータを物理的に安全な状態にすることで、リソースと情報を保護できます。 タイムシェアリング コンピューティング環境では、オペレーティング システムによって互いにユーザが保護され、リソースが管理されます。 各ユーザが読み取りまたは修正できる対象を判断するには、タイムシェアリング システムによる各ユーザの識別が必要です。 これは、ユーザのログイン時に実行されます。

多くの異なるコンピュータからのサービスを必要とするユーザ ネットワークでは、コントロールにアクセスできる方法が 3 つあります。 不正アクセスを防止するためには何もせず、ユーザがログインするマシンに依存する方法。 ユーザの身元を証明するホストを必要とするが、ユーザが誰

であるかはホストの判断を信頼する方法。 必要なサービスそれぞれについて、ユーザが自分の身元を証明する必要がある方法。

すべてのマシンが厳密な管理下にあるクローズな環境では、1 番目のアプローチを使用できます。 組織がネットワーク上で通信を行うすべてのホストを管理する場合、これは妥当なアプローチです。

よりオープンな環境では、組織の管理下にあるホストだけを選択して信頼することがあります。 この場合、各ホストはその身元を証明する必要があります。 rlogin と rsh のプログラムでは、このアプローチが使用されます。 これらのプロトコルでは、接続の確立元のインターネット アドレスをチェックすることで認証が実行されます。

Athena の環境では、組織による管理を受けていないホストからの要求を承認できる必要があります。 ユーザはワークステーションを完全に制御でき、 これらをリブートしたり、単独でアップ状態にしたり、独自のテープからブートすることもできます。 したがって、ユーザは対象の各サービスに自身の身元を証明する必要がある、 3 番目のアプローチを適用する必要があります。 サーバも自身の身元を証明する必要があります。 ネットワーク サーバが稼働しているホストを物理的に安全な状態に置くだけでは十分ではなく、 何者かがネットワーク上の他の場所から特定のサーバのように身元を偽る可能性があります。

我々の環境では、識別メカニズムに対していくつかの要件が必要になります。 まず、識別システムは安全である必要があります。 識別メカニズムを回避することが非常に困難である必要があり、これにより潜在的な攻撃者によって認証メカニズムが弱点であると認識されないようにする必要があります。 ネットワークを監視しているユーザが、別のユーザになりすますために必要な情報を取得できないようにする必要があります。 2 番目に、識別システムの信頼性が高いことが必要とされます。 多くのサービスへのアクセスは、認証サービスに依存することになります。 識別システムの信頼性が高くないと、サービスに関するシステムの信頼性は全体として高くはなりません。 3 番目に、識別システムは透過的である必要があります。 理想的には、認証が実行されていることがユーザに認識されるべきではありません。 最後に、識別システムはスケーラブルである必要があります。 多くのシステムでは、Athena のホストとの通信が可能です。 これらすべてで我々のメカニズムがサポートされるわけではありませんが、もしサポートされた場合でも、ソフトウェアの破綻は発生しないはずです。

Kerberos は、上記の要件を満たすための取り組みの結果です。 ユーザはワークステーションにアクセスする際にログインを行います。 ユーザが認識する限りでは、この初期の識別は、ログイン セッションの間に必要なすべてのネットワーク サーバに対する身元証明としては十分です。 Kerberos のセキュリティは、複数の認証サーバのセキュリティに依存するものであり、ユーザのログイン元のシステムや使用されるエンド サーバのセキュリティに依存するものではありません。 適切に認証されたユーザに対して、その身元をネットワーク全体に分散されたサーバに証明するための方法が、認証サーバにより提供されます。

認証は、セキュアなネットワーク環境の基礎となる構成要素です。 たとえば、サーバによってクライアントの身元が確実に認識されている場合、サービスを提供するのかどうか、ユーザに特権が与えられる必要があるのか、サービスの料金をだれに請求する必要があるのか、などをサーバによって判断することができます。 つまり、権限付与と課金のスキームを、Kerberos によって提供される認証に基づいて構築することができ、単独のパーソナル コンピュータやタイムシェアリング システムに同等のセキュリティが提供されることになります。

Kerberos について

Kerberos は、Needham - Schroeder 提唱モデルに基づく信頼されたサードパーティ認証サービスです。 各クライアントが他の各クライアントの身元の真偽についての Kerberos の判断を信用し

ているという点において、Kerberos は信頼されています。 元来のモデルに対して、リプレイの検出を支援するために、タイムスタンプ（現在の日時を示す大きい数字）が付加されています。 リプレイは、メッセージがネットワークから盗まれ、後から再送される場合に発生します。 リプレイの詳細および認証の他の問題については、Voydock と Kent による文献を参照してください。

Kerberos の機能

Kerberos では、そのクライアントのデータベースと秘密鍵が保持されます。 秘密鍵は、Kerberos とそれが属するクライントにだけ認識される大きい数字です。 クライアントがユーザの場合、これは暗号化されたパスワードです。 認証が必要なネットワーク サービスは Kerberos に登録され、これらのサービスを使用する必要があるクライアントも Kerberos に登録されます。 秘密鍵は登録時にネゴシエートされます。

Kerberos ではこれらの秘密鍵が認識されているため、あるクライアントが別のクライアントの主張する身元の信憑性について確証を得るためのメッセージを作成することができます。 Kerberos でもセッション鍵と呼ばれる一時的な秘密鍵が生成され、2 つのクライアントにだけ提供されます。 セッション鍵は、2 つのパーティ間でメッセージを暗号化するために使用できます。

Kerberos では、3 つの異なるレベルの保護が提供されます。 アプリケーション プログラマは、どのレベルが適切なのかをアプリケーションの要件に応じて判断します。 たとえば、一部のアプリケーションではネットワーク接続の初期においてだけ信頼性の確立が必要があり、特定のネットワーク アドレスからのそれ以降のメッセージはその認証されたパーティから発信されていると仮定することができます。 我々の認証されたネットワーク ファイル システムでは、このレベルのセキュリティが使用されます。

他のアプリケーションでは各メッセージの認証が必要ですが、メッセージの内容が開示されるかどうかは問われません。 これらの場合、Kerberos によって安全なメッセージが提供されます。 ただし、非公開のメッセージではより高度なセキュリティが提供され、各メッセージが認証されるだけではなく、暗号化もされます。 たとえば、ネットワーク上でパスワードを送信するために、Kerberos サーバ自体によって非公開のメッセージが使用されます。

Kerberos ソフトウェアのコンポーネント

Athena 実装は、次のような複数のモジュールから構成されています。

- Kerberos アプリケーション ライブラリ
- 暗号化ライブラリ
- データベース ライブラリ
- データベース管理プログラム
- 管理サーバ
- 認証サーバ
- データベース伝搬ソフトウェア
- ユーザ プログラム
- アプリケーション

Kerberos アプリケーション ライブラリでは、アプリケーション クライアントとアプリケーション サーバのインターフェイスが提供されます。 特に、認証要求の作成や読み取りを行うためのルーチンと、安全なメッセージや非公開のメッセージを作成するためのルーチンが含まれています。

Kerberos での暗号化は、Data Encryption Standard（DES; データ暗号規格）に基づいています。

暗号化ライブラリでは、これらのルーチンが実装されます。 速度とセキュリティのバランスがそれぞれ異なる、暗号化のための複数の方法が提供されます。 DES Cypher Block Chaining (CBC) モードに対する拡張である Propagating CBC モードも提供されます。 CBC では、エラーが伝搬されるのは現在の暗号ブロックでだけですが、PCBC ではエラーがメッセージ全体で伝搬されます。 このため、エラーが発生すると、メッセージの一部だけではなく、全体が役に立たなくなります。 暗号化ライブラリは独立したモジュールであり、他の DES 実装または別の暗号化ライブラリと置換される場合があります。

もう 1 つの置換可能なモジュールは、データベース管理システムです。 データベース ライブラリの現在の Athena 実装では ndbm が使用されていますが、もともとは Ingres が使用されていました。 他のデータベース管理ライブラリも使用可能です。

Kerberos データベースが必要とするものは単純です。 レコードはプリンシパルごとに格納され、プリンシパルの名前、秘密キー、有効期限が若干の管理情報とともに含まれています。 (有効期限は、その翌日からエントリが無効になる日付です。 通常は登録時に 2、3 年後が設定されます。)

実際の名前、電話番号などの他のユーザ情報は、別のサーバ (Hesiod ネームサーバ) によって保持されます。 このように、パスワードなどの機密情報はかなり高いセキュリティ対策を使用して、Kerberos により処理できます。 その一方で、Hesiod が保持する非機密情報は別の取り扱いを受けます。 たとえば、暗号化されずにネットワークを介して送信されることがあります。

データベースの管理をツールが行う場合のように、Kerberos サーバではデータベース ライブラリが使用されます。

管理サーバ (または KDBM サーバ) では、読み取りと書き込みのネットワーク インターフェイスがデータベースに提供されます。 プログラムのクライアント側は、ネットワーク上の任意のマシンで実行される場合があります。 ただし、サーバ側は、データベースへの変更を行うために Kerberos データベースが収容されるマシンで実行される必要があります。

それに対して、認証サーバ (または Kerberos サーバ) では、Kerberos データベース上で読み取り専用の操作 (プリンシパルの認証、セッション鍵の生成など) が実行されます。 このサーバでは Kerberos データベースが修正されないため、マスター Kerberos データベースの読み取り専用コピーが収容されているマシン上で実行される可能性があります。

データベース伝搬ソフトウェアでは、Kerberos データベースの複製が管理されます。 各マシン上で稼働する認証サーバのコピーを使用して、複数の異なるマシン上にデータベースのコピーを配置することが可能です。 これらの各スレーブ マシンでは、特定の間隔でマスター マシンから Kerberos データベースのアップデートが受信されます。

最後に、Kerberos への口グイン、Kerberos パスワードの変更、および Kerberos チケットの表示または破棄 (チケットについては後で説明しています) のために、エンド ユーザ プログラムが存在します。

Kerberos で使用される名前

エンティティの認証には、そのエンティティの名前付けが必要です。 認証のプロセスは、そのクライアントが要求で名前が指定されているクライアントであることの確認です。 名前の構成要素は次のとおりです。 Kerberos では、ユーザとサーバの両方に名前が付けられます。 認証サーバの視点では、これらは同等です。 名前は、プライマリ名、インスタンス、およびレルムで構成され、名前.インスタンス@レルムのように表されます。

プライマリ名は、ユーザまたはサービスの名前です。 インスタンスは、プライマリ名の派生形を区別するために使用されます。 ユーザの場合、インスタンスは「ルート (root) 」インスタンスや「管理 (admin) 」インスタンスなどの特権を伴います。 Athena 環境でのサービスの場合、インスタンスは通常、サーバが稼働するマシンの名前です。 たとえば、rlogin サービスには異なるホスト上に異なるインスタンスがあり、 rlogin.priam は priam と名付けられたホスト上の rlogin サーバです。 Kerberos チケットは、単一の名前が付けられたサーバに対してだけ有効です。 したがって、同じサービスの異なるインスタンスにアクセスするには、個別のチケットが必要になります。 レルムは、認証データが維持される管理エンティティの名前です。 たとえば、異なる機関はそれぞれ独自の Kerberos マシンを所有し、異なるデータベースを収容している場合があります。 これらには、別々の Kerberos レルムがあります。 (レルムについては「他の Kerberos との相互対話」で詳しく説明します。)

Kerberos の動作の仕組み

このセクションでは、Kerberos 認証プロトコルについて説明しています。 上記のように、Kerberos 認証モデルは Needham - Schroeder 鍵配布プロトコルに基づくものです。 ユーザがサービスを要求する場合、その身元が証明される必要があります。 これを実行するために、チケットとともに、そのチケットが盗まれたものではなく、もともとユーザに発行されたチケットであることの証明がサーバに提示されます。 Kerberos による認証には 3 つのフェーズがあります。 1 番目のフェーズでは、ユーザが他のサービスへのアクセスを要求するために使用されるクレデンシャルを取得します。 2 番目のフェーズでは、ユーザは特定のサービスのための認証を要求します。 最後のフェーズでは、ユーザはこれらのクレデンシャルをエンド サーバへ提示します。

Kerberos クレデンシャル

Kerberos 認証モデルで使用されるクレデンシャルには、 チケットとオーセンティケータの 2 種類があります。 どちらも秘密鍵暗号化に基づいていますが、それぞれ異なる鍵を使用して暗号化されます。 チケットは、認証サーバとエンド サーバの間で、チケットが発行された人物の身元を安全に受け渡すために使用されます。 チケットによっても、チケットを使用している人物がチケットの発行対象の人物と同一であることを確認するために使用される情報が受け渡されます。 オーセンティケータには追加情報が含まれ、これをチケット内の情報と比較すると、チケットを提示するクライアントがチケットの発行対象のクライアントと同一であることが証明されます。

チケットは、単一のサーバと単一のクライアントに対してだけ有効です。 チケットには、サーバの名前、クライアントの名前、クライアントのインターネット アドレス、タイムスタンプ、ライフタイム、およびランダム セッション鍵が含まれています。 この情報は、チケットの使用対象のサーバの鍵を使用して暗号化されます。 チケットが発行されると、チケットの有効期限が切れるまで、名前が付けられたサーバにアクセスするために名前が付けられたクライアントによって複数回使用される場合があります。 チケットはサーバの鍵で暗号化されるため、ユーザによるチケットの修正を懸念する必要なく、ユーザによるチケットのサーバへの受け渡しを許可することが安全であることに注意してください。

チケットとは異なり、オーセンティケータは 1 回だけ使用できます。 クライアントがサービスを使用する際は常に、新規のオーセンティケータが生成される必要があります。 クライアントは独自にオーセンティケータを構築できるため、このことは問題になりません。 オーセンティケータには、クライアント名、ワークステーションの IP アドレス、および現在のワークステーションの時刻が含まれます。 オーセンティケータは、チケットの一部であるセッション鍵で暗号化されます。

初期 Kerberos チケットの取得

ユーザがワークステーションにアクセスする際には、ある 1 つの情報、 つまりユーザのパスワードによってだけその身元が証明されます。 認証サーバとの初期の交換は、パスワードが攻撃対象となる可能性を最小限にする設計になっているのと同時に、ユーザはこのパスワードを知らないと自分自身を適切に認証できないようになっています。 ユーザにとっては、ログインのプロセスはタイムシェアリング システムへのログインと同じように見えます。 ところが、表面下ではまったく異なっています。

ユーザはユーザ名の入力が求められます。 ユーザ名が入力されると、ユーザ名とチケット認可サービスとして知られる特殊なサービス名が指定された要求が認証サーバへ送信されます。

認証サーバでは、クライアントに関する認識がチェックされます。 クライアントが認識されている場合、後でクライアントとチケット認可サーバの間で使用されるランダム セッション鍵が生成されます。 次に、クライアント名、チケット認可サーバ名、現在の時刻、チケットのライフタイム、クライアントの IP アドレス、および作成されたばかりのランダム セッション鍵が含まれる、チケット認可サーバのチケットが作成されます。 これはすべて、チケット認可サーバと認証サーバにだけ認識されている鍵で暗号化されます。

次に、認証サーバによってチケットとともにランダム セッション鍵と追加情報がクライアントに戻されます。 この応答は、ユーザのパスワードから導出されたクライアントの秘密鍵で暗号化されますが、この鍵が認識されるのは Kerberos とクライアントだけです。

クライアントによって応答が受信されると、ユーザはパスワードの入力を求められます。 このパスワードは DES 鍵に変換され、認証サーバからの応答を復号化するために使用されます。 チケットとセッション鍵は、他の情報とともに将来の使用のために保存され、ユーザのパスワードと DES 鍵はメモリから消去されます。

交換が完了すると、チケット認可チケットのライフタイムの間にユーザの身元の証明用にワークステーションによる使用が可能な情報が取得されます。 ワークステーション上のソフトウェアが改ざんされていない限り、他者がチケットの期限を越えてユーザの身元を偽ることを可能にする情報は存在しません。

Kerberos サービスの要求

便宜上、ここでは、ユーザがすでに対象のサーバのチケットを所有しているものと仮定します。 サーバへアクセスするために、アプリケーションによってクライアント名、IP アドレス、および現在の時刻が含まれるオーセンティケータが構築されます。 次に、オーセンティケータは、サーバのためのチケットで受信されたセッション鍵で暗号化されます。 次に、個々のアプリケーションによって定義された方法で、クライアントによってオーセンティケータがチケットとともにサーバへ送信されます。

オーセンティケータとチケットがサーバで受信されると、サーバではチケットが復号化され、チケットに含まれるセッション鍵を使用してオーセンティケータが復号化された上で、チケット内の情報がオーセンティケータ内の情報、要求の発信元の IP アドレス、および現在の時刻と比較されます。 すべてが一致すると、要求の続行が許可されます。

クロックは数分の範囲内で同期されていると仮定されています。 要求内の時刻が前後に大きくずれている場合、サーバではこの要求が以前の要求のリプレイが試行されているものとして処理されます。 また、サーバには、タイムスタンプがまだ有効である過去のすべての要求の追跡を保持することも許可されています。 さらにリプレイ アタックをかわすために、すでに受信済みのものと同じチケットとタイムスタンプを使用して受信された要求は廃棄できます。

最後に、クライアントによってサーバの身元の証明も必要であると指定される場合、サーバでは

クライアントによってオーセンティケータ内で送信されたタイムスタンプに身元が追加され、セッション鍵で結果が暗号化されて、結果がクライアントに戻されます。

この交換の最後では、Kerberos に従って、サーバによってクライアントの主張する身元が正しいことが確認されます。 相互認証が発生する場合、クライアントによってもサーバの身元が正しいことが確認されます。 さらに、クライアントとサーバによって他者に認識されていない鍵が共有されるため、比較的最近のメッセージは他のパーティから発信されたその鍵で暗号化されたと安全に判断することができます。

Kerberos サーバ チケットの取得

チケットは、単一のサーバに対してだけ有効であることに留意してください。 したがって、クライアントによって使用される各サービスに対しては、個別のチケットを取得することが必要です。 個々のサーバのチケットは、チケット認可サービスから取得できます。 チケット認可サービスそれ自体がサービスであるため、前記のセクションで説明したサービス アクセス プロトコルが利用されます。

まだ要求されていないチケットがプログラムで必要となると、チケット認可サーバに要求が送信されます。 要求には、チケットの要求対象のサーバ名とともに、前記のセクションで説明したようにチケット認可チケットおよび構築されたオーセンティケータが含まれます。

次に、上記のように、チケット認可サーバによってオーセンティケータとチケット認可チケットがチェックされます。 有効な場合、チケット認可サーバによって、クライアントと新規のサーバ間で使用される新規のランダム セッション鍵が生成されます。 次に、クライアント名、サーバ名、現在の時刻、クライアントの IP アドレス、および生成されたばかりの新規のセッション鍵が含まれる、新規のサーバのためのチケットが構築されます。 新規のチケットのライフタイムはチケット認可チケットの残余期限の最小値になっており、これがサービスに対するデフォルトです。

次に、チケット認可サーバによってチケットがセッション鍵や他の情報とともにクライアントへ戻されます。 ただし、この時点で、応答はチケット認可チケットの一部であったセッション鍵で暗号化されます。 これにより、ユーザはパスワードを再入力する必要がなくなります。

Kerberos データベース

ここまでは、Kerberos データベースへの読み取り専用アクセスが必要になる操作について説明しました。 これらの操作は、マスター マシンとスレーブ マシンの両方で稼働可能な認証サービスによって実行されます。

このセクションでは、データベースへの書き込みアクセスが必要な操作について説明しています。 これらの操作は、Kerberos Database Management Service（KDBM）と呼ばれる管理サービスによって実行されます。 現在の実装では、マスターの Kerberos データベースにだけ変更が許可されることが規定されており、 スレーブ コピーは読み取り専用です。 したがって、KDBM サーバが稼働できるのはマスター Kerberos マシンでだけです。

（スレーブでも）認証が発生する可能性はありますが、マスター マシンがダウン状態の場合には管理要求にサービスを提供できないことに注意してください。 経験的には、管理要求の頻度が低いため、このことによって問題が発生することはありませんでした。

KDBM では、パスワードを変更するためのユーザからの要求が処理されます。 ネットワークを介して KDBM へ要求を送信するこのプログラムのクライアント側は、kpasswd プログラムです。 KDBM でも、データベースへのプリンシパルの追加さらには既存のプリンシパルのパスワードの

変更を行う場合がある Kerberos 管理者からの要求が受け付けられます。 管理プログラムのクライアント側でもネットワークを介して KDBM へ要求が送信されますが、これは kadmin プログラムです。

KDBM サーバ

KDBM サーバでは、データベースへのプリンシパルの追加または既存のプリンシパルのパスワードの変更を行うための要求が受け付けられます。 このサービスは、チケット認可サービスによってこのサービス用のチケットが発行されることはないという点で特異です。 その代わりに、認証サービス自体が使用される必要があります (チケット認可チケットを取得するために使用される同じサービス)。 この目的は、ユーザにパスワードの入力を要求することです。 ユーザによるパスワードの入力が必要でなかったとしたら、ユーザがワークステーションから離れた場合に他のユーザがアクセスして元のユーザのパスワードを変更することが可能になってしまいますが、このような事態は回避する必要があります。 同様に、管理者が保護されないままワークステーションから離れてしまった場合、他のユーザがシステム内の任意のパスワードを変更する可能性があります。

KDBM サーバによって要求が受信されると、変更の要求者の認証されたプリンシパル名を、要求の対象のプリンシパル名と照合することで、承認が行われます。 これらが同じ場合は、要求が許可されます。 これらが同じではない場合、KDBM サーバによって (マスターの Kerberos システム上のファイルに格納された) アクセス コントロール リスト (ACL) が参照されます。 要求者のプリンシパル名がこのファイル内で見つかった場合は要求が許可されますが、見つからない場合は拒否されます。

慣例により、NULL インスタンス (デフォルト インタンス) が含まれる名前は、アクセス コントロール リスト ファイルには表示されず、 代わりに管理インスタンスが使用されます。 したがって、ユーザが Kerberos の管理者になるためには、そのユーザ名の管理インスタンスが作成され、アクセス コントロール リストに追加される必要があります この慣例を使用すると、管理者は通常のログインに使用するパスワードとは異なるパスワードを Kerberos 管理のために使用できます。

KDBM プログラムへのすべての要求は、許可されるか拒否されるかにかかわらず、ログに記録されます。

kadmin プログラムと kpasswd プログラム

Kerberos の管理者は kadmin プログラムを使用して、データベースへのプリンシパルの追加や既存のプリンシパルのパスワードの変更を行います。 管理者は kadmin プログラムを呼び出すと、管理インスタンス名のパスワードの入力を求められます。 このパスワードは、KDBM サーバのチケットを取得するために使用されます。

ユーザは、kpasswd プログラムを使用して、Kerberos パスワードを変更する場合があります。 ユーザはこのプログラムを呼び出すと、以前のパスワードの入力を求められます。 このパスワードは、KDBM サーバのチケットを取得するために使用されます。

Kerberos データベースの複製

各 Kerberos レルムには、認証データベースのマスター コピーが収容される、マスターの Kerberos マシンが存在します。 (必ずしも必要というわけではありませんが) システムの他の部分のスレーブ マシン上には、データベースの追加の読み取り専用コピーを配置することができます。 データベースの複数コピーの存在が有利な点は、複製についてよく言われるように、 高度

な可用性と優れたパフォーマンスです。 マスター マシンがダウン状態の場合でも、いずれかのスレーブ マシンでの認証が可能です。 複数のマシンのいずれかで認証を実行する機能により、マスター マシンでのボトルネック発生の可能性が低くなります。

ところが、データベースの複数のコピーを保持することにより、データの整合性の問題が発生します。 不整合の問題に十分対処できる非常に簡単な方法が見つかっています。 マスター データベースは、1 時間ごとにダンプされます。 データベースはスレーブ マシンへそのまま送信され、スレーブ マシンによって各自のデータベースがアップデートされます。 kprop と呼ばれるマスター ホスト上のプログラムによって、各スレーブ マシンで稼働する kpropd と呼ばれるピア プログラムへアップデートが送信されます。 まず、kprop によって、送信対象の新しいデータベースのチェックサムが送信されます。 チェックサムは、マスターとスレーブの Kerberos マシンの両方で保持される、Kerberos マスター データベースの鍵で暗号化されます。 次に、データはネットワークを介してスレーブ マシン上の kpropd へ転送されます。 スレーブ伝搬サーバでは受信されたデータのチェックサムが算出され、これがマスターによって送信されたチェックサムに一致する場合、この新しい情報を使用してスレーブのデータベースがアップデートされます。

Kerberos データベース内のすべてのパスワードは、マスター データベース鍵で暗号化されるため、傍受者はネットワークを介してマスターからスレーブへ受け渡される情報を悪用できません。 ただし、スレーブでマスター ホストからの情報だけが受け取られるようにして、さらにデータの改ざんが検出されるようにすることは必須であるため、チェックサムが使用されます。

外部の視点からの Kerberos

このセクションでは実用的な観点から Kerberos について説明していますが、まず最初にユーザの視点から、次にアプリケーション プログラマの視点から、そして最後に Kerberos 管理者のタスクの視点から説明しています。

Kerberos ユーザの視点

すべてが正常な場合、ユーザが Kerberos の存在を認識することはほとんどありません。 UNIX 実装では、ログイン プロセスの一環としてチケット認可チケットが Kerberos から取得されます。 ユーザの Kerberos パスワードの変更は、kpasswd プログラムの機能です。 また、Kerberos チケットはユーザのログアウト時に自動的に破棄されます。

チケット認可チケットのライフタイム (8 時間) より長くユーザのログイン セッションが継続すると、Kerberos 認証済みのアプリケーションが次回の実行で失敗するため、Kerberos の存在がユーザに認識されます。 この Kerberos チケットは有効期限切れとなります。 この時点で、チケット認可サーバの新規チケットを取得するために、ユーザは kinit プログラムを実行できます。 ログイン時には、これを取得するためにパスワードが入力される必要があります。 ユーザが好奇心から klist コマンドを実行すると、Kerberos 認証を必要とするサービスのためにユーザに代わり通知なしで取得されたすべてのチケットに驚かされることがあります。

プログラマの視点からの Kerberos

Kerberos アプリケーションを書いているプログラマは、クライアント側とサーバ側から構成される既存のネットワーク アプリケーションに対して、よく認証を付け加えることになります。 このプロセスのことを、プログラムの「Kerberos 化処理」と呼びます。 Kerberos 化処理には通常、サービスの初期要求で認証を実行するための Kerberos ライブラリに対するコールが含まれています。 また、アプリケーション クライアントとアプリケーション サーバの間で引き続き送信されるメッセージとデータを暗号化するための DES ライブラリに対するコールも含まれています。

最も一般的に使用されるライブラリ関数は、クライアント側の krb_mk_req、サーバ側の krb_rd_req です。 krb_mk_req ルーチンでは、要求されるものとして、ターゲット サーバの名前、インスタンス、およびレルムなどのパラメータが使用されますが、送信されるデータのチェックサムである場合もあります。 次に、krb_mk_req のコールで戻されたメッセージが、クライアントにより、アプリケーションのサーバ側へネットワークを介して送信されます。 サーバでは、このメッセージが受信されると、ライブラリ ルーチン krb_rd_req へのコールが実行されます。 このルーチンでは、送信者の主張する身元の真偽についての判断が返されます。

アプリケーションによって、クライアントとサーバ間で送信されるメッセージを非公開にすることが求められた場合、ライブラリ コールが krb_mk_priv (krb_rd_priv) に対して行われ、両者によって共有されるようになったセッション鍵でメッセージが暗号化 (復号化) されます。

Kerberos 管理者の業務

Kerberos 管理者の業務は、データベースを初期化するためのプログラムを実行することから開始されます。 管理インスタンスを伴う Kerberos 管理者の名前など、データベース内に必須のプリンシパルを登録するために別のプログラムを実行する必要もあります。 Kerberos 認証サーバと管理サーバを起動する必要があります。 スレーブ データベースが存在する場合、管理者はデータベースのアップデートをマスターからスレーブに伝搬するためのプログラムが定期的に起動されるようにプログラムを設定する必要があります。

これらの初期手順の実行後、管理者は kadmin プログラムを使用することにより、ネットワークを介してデータベースを操作します。 このプログラムを使用して、新しいプリンシパルを追加したり、パスワードを変更したりできます。

特に、新しい Kerberos アプリケーションがシステムに追加されると、Kerberos 管理者はそれを使用するためにいくつかの手順を実行する必要があります。 サーバはデータベース内に登録される必要があり、秘密鍵が割り当てられる必要があります (通常、これは自動生成されたランダムな鍵です) 。 次に、一部のデータ (サーバの鍵など) がデータベースから抽出され、サーバのマシン上のファイル内にインストールされる必要があります。 デフォルト ファイルは /etc/srvtab です。 サーバからコールされた krb_rd_req ライブラリ ルーチン (上記のセクションを参照) では、そのファイル内の情報が使用され、サーバの秘密鍵で暗号化されて送信されたメッセージが復号化されます。 /etc/srvtab ファイルでは、端末で入力されたパスワードによってユーザが認証される際にサーバが認証されます。

Kerberos 管理者は Kerberos マシンが物理的にセキュアであることを確認する必要もあり、マスター データベースのバックアップを保持することも推奨されます。

大規模な Kerberos 環境

このセクションでは、他のネットワーク サービスとアプリケーションによる使用方法やリモートの Kerberos レルムとの相互対話の方法など、Athena 環境に Kerberos がどのように適合するのかについて説明しています。 Athena 環境についての詳細は、G.W. Treese による文献を参照してください。

他のネットワーク サービスによる Kerberos の使用

複数のネットワーク アプリケーションが Kerberos を使用するために修正されています。 rlogin コマンドと rsh コマンドでは、まず Kerberos を使用した認証が試行されます。 有効な Kerberos チケットを持つユーザは、.rhosts ファイルを設定しないでも別の Athena マシンに rlogin を実行できます。 Kerberos 認証に失敗すると、プログラムによって通常の承認方式 (この場合は

.rhosts ファイル) にフォールバックが行われます。

「post office」から電子メールを取得する必要があるユーザの認証に Kerberos を使用するために、Post Office Protocol (POP) が修正されています。 最近、Athena では Zephyr と呼ばれるメッセージ配信プログラムが開発されましたが、このプログラムでも認証に Kerberos が使用されます。

register と呼ばれる新規ユーザのサインアップのためのプログラムでは、Service Management System (SMS) と Kerberos の両方が使用されます。 SMS からは、新たな Athena ユーザとなる可能性を持つユーザによって入力された情報 (名前や MIT ID 番号など) が有効であるかどうかが判断されます。 次に、要求されたユーザ名が一意であるかどうかが Kerberos を使用して確認されます。 すべてが有効であることが確認されると、ユーザ名とパスワードを含む新しいエントリが Kerberos データベースに作成されます。

Sun のネットワーク ファイル システムを保護するための Kerberos の使用についての詳細は、「付録」を参照してください。

他の Kerberos との相互対話

さまざまな管理組織がユーザ認証のために Kerberos を使用することが予想されます。 また、多くの場合、ある組織のユーザが別の組織のサービスを使用することも予想されます。 Kerberos では、複数の管理ドメインがサポートされています。 Kerberos 内の名前の仕様には、レルムと呼ばれるフィールドがあります。 このフィールドには、その内部でユーザが認証される管理ドメインの名前が含まれています。

通常、サービスは単一のレルム内に登録され、そのレルム用の認証サーバによって発行されたクレデンシャルだけが受け付けられます。 ユーザは、多くの場合、単一のレルム (ローカル レルム) 内に登録されますが、このユーザがローカル レルムによって提供される認証に基づいて、別のレルム (リモート レルム) によって発行されたクレデンシャルを取得することは可能です。 リモート レルム内で有効なクレデンシャルには、ユーザがもともと認証されているレルムが示されます。 リモート レルム内のサービスでは、要求されるセキュリティのレベルやユーザが当初認証されたレルムの信頼のレベルに従って、これらのクレデンシャルを認めるかどうかを選択できます。

レルム間認証を実行するには、各ペアのレルムの管理者がこれらのレルム間で共有される鍵を選択する必要があります。 次に、ローカル レルム内のユーザは、リモート レルム内のチケット認可サーバのチケット認可チケットをローカル認証サーバから要求できます。 このチケットが使用されると、リモート チケット認可サーバによって要求が自分のレルムからではないことが認識され、チケット認可チケットを復号化するために以前に交換された鍵が使用されます。 次に、クライアントのレルム フィールドにクライアントがもともと認証されたレルムの名前が含まれる点を除き、通常と同様にチケットが発行されます。

このアプローチを拡張すると、対象のサービスに到達するまで一連のレルムにわたってユーザが自分自身を認証することができます。 ただし、これを実行するためには、ユーザが認証された当初のレルムの名前だけではなく、使用されたパス全体を記録する必要があります。 このような状況では、サーバによって認識されるユーザの身元は間接的に確認されるものに過ぎません。 この内容を信頼できるのは、パス全体に関係するすべてのユーザが信頼される場合だけです。

Kerberos 問題および未解決の問題

Kerberos 認証メカニズムに関連する数多くの未解決の問題があります。 これらの問題には、チ

ケットのライフタイムを正確に判断する方法、プロキシを許可する方法、ワークステーションの整合性を保証する方法などがあります。

チケットのライフタイムの問題は、セキュリティと利便性の間で適切なバランスを取ることに関連する問題です。 チケットの有効期限が長い場合は、チケットとその関連付けられたセッション鍵が盗まれたり置き違えられたりすると、それらが長期に渡り使用される可能性があります。 ユーザが公開されたワークステーションからログアウトし忘れる場合に、このような情報が盗まれる可能性があります。 また、複数のユーザが許可されるシステム上でユーザが認証されている場合、盗まれたチケットを使用するために必要な情報を、ルート (root) へアクセスできる別のユーザが検索できる可能性があります。 ただし、短いライフタイムをチケットに設定する場合の問題は、有効期限が切れる際に、パスワードの再入力が必要になる新規のチケットをユーザが取得する必要があることです。

プロキシの問題は未解決の問題です。 認証されたユーザは、自分自身の代わりとしてどのようにサーバに他のネットワーク サービスの取得を許可するのでしょうか。 たとえば、ファイルサーバから保護されたファイルへ直接的にアクセスするサービスを使用する場合に、このことが重要になります。 この問題の別の例としては、いわゆる認証転送が挙げられます。 ユーザがワークステーションにログインしていて、リモート ホストにログインする場合には、リモート ホスト上でプログラムを実行する一方で、ユーザがローカルで利用可能な同じサービスにアクセスできれば好都合です。 ユーザがリモート ホストを信頼していないことがあるため、認証転送は常に理想的であるというわけではなく、この状況は困難になります。 現在のところ、この問題に対するソリューションはありません。

もう 1 つの問題 (Athena 環境において重要な問題) は、ワークステーションで稼働するソフトウェアの整合性を保証する方法です。 非公開のワークステーションでは、それを使用するユーザがそのワークステーションを管理できるため、このことは大きな問題になりません。 ところが、公開されたワークステーションでは、ユーザのパスワードを保存するログイン プログラムが他者のアクセスにより修正されている可能性があります。 我々の環境で現在利用可能な唯一のソリューションは、公開されたワークステーションで稼働するソフトウェアの他者による修正を困難にすることです。 優れたソリューションとしては、信頼することができるとユーザが認識しているシステムからユーザの鍵が移動しないようにすることが挙げられます。 これを実行する方法の 1 つとしては、暗号化を実行する機能を備えたスマート カードをユーザが保有していた場合に、認証プロトコルで要求することが挙げられます。

Kerberos のステータス

Kerberos のプロトタイプ バージョンは、1986 年 9 月に製品化されました。 1987 年 1 月以降、Kerberos は Project Athena の 5,000 人のユーザ、650 台のワークステーション、65 台のサーバの唯一の認証手段でした。 さらに、現在、Kerberos は Athena の複数のタイムシェアリング システムへのアクセスを制御するために、.rhosts ファイルの代わりに使用されています。

Kerberos 確認応答

Kerberos は当初、Jeff Schiller 氏と Jerry Saltzer 氏の提言を取り入れ、Steve Miller 氏と Clifford Neuman 氏によって設計されました。 以後、多くの人たちがプロジェクトに関与してきました。これら協力者の一部としては、Jim Aspnes 氏、Bob Baldwin 氏、John Barba 氏、Richard Basch 氏、Jim Bloom 氏、Bill Bryant 氏、Mark Colan 氏、Rob French 氏、Dan Geer 氏、John Kohl 氏、John Kubiatowicz 氏、Bob Mckie 氏、Brian Murphy 氏、John Ostlund 氏、Ken Raeburn 氏、Chris Reed 氏、Jon Rochlis 氏、Mike Shanzer 氏、Bill Sommerfeld 氏、Ted T'so 氏、Win Treese 氏、Stan Zanarotti 氏が挙げられます。

また、Dan Geer 氏、Kathy Lieben 氏、Josh Lubarr 氏、Ken Raeburn 氏、Jerry Saltzer 氏、Ed Steiner 氏、Robbert van Renesse 氏、Win Treese 氏の提言によって、このドキュメントの初期草稿が大きく改善されました。

Jedlinsky、J.T. Kohl、および W.E. Sommerfeld 著、『The Zephyr Notification System』、Usenix Conference Proceedings（1988 年冬）

M.A. Rosenstein、D.E. Geer、および P.J. Levine 著、Usenix Conference Proceedings（1988 年冬）

R. R. Sandberg、D. Goldberg、S. Kleiman、D. Walsh、および B. Lyon 著、『Design and Implementation of the Sun Network Filesystem』、Usenix Conference Proceedings（1985 年夏）

付録： SUN の Network File System（NFS; ネットワーク ファイル システム）への Kerberos の適用

Project Athena ワークステーション システムの主要なコンポーネントとして、ユーザのワークステーションとその非公開のファイル ストレージ（ホーム ディレクトリ）の間でのネットワークの介入があります。 すべての非公開ストレージは、この目的専用の一連のコンピュータ（現在は VAX 11/750）に常駐しています。 これにより、一般的に利用可能な UNIX ワークステーションでサービスを提供できます。 ユーザがこれら一般的に利用可能なワークステーションのいずれかにログインする際には、ユーザの名前とパスワードをローカルに常駐するパスワード ファイルと照合するのではなく、Kerberos を使用してユーザの信憑性を判断します。 ログイン プログラムによってユーザ名の入力が求められます（どの UNIX システムでも同じ）。 このユーザ名は、Kerberos チケット認可チケットを取得するために使用されます。 ログイン プログラムでは、チケットの復号化のための DES 鍵を生成するために、パスワードが使用されます。 復号化に成功すると、ユーザのホーム ディレクトリが Hesiod ネーミング サービスを参照して検索され、NFS を介してマウントされます。 次に、ログイン プログラムによってユーザのシェルに制御が受け渡され、ワークステーションにホーム ディレクトリが「接続」されたため、従来型のユーザごとのカスタマイゼーション ファイルの実行が可能になります。 Hesiod サービスもローカル パスワード ファイル内でエントリを構築するために使用されます（これは /etc/passwd 内で情報を検索するプログラムのためのものです）。

ここでは、リモート ファイル サービス配布のいくつかのオプションから、Sun のネットワーク ファイル システムを選択しています。 ただし、このシステムは根本的に我々のニーズに合致するものではありません。 NFS では、（ファイル サーバの視点では）すべてのワークステーションが 2 つのカテゴリ（信頼されるか、または信頼されないか）に分類されることを前提としています。 信頼されないシステムはどのファイルにもアクセスできませんが、信頼されるシステムはどのファイルにもアクセスできます。 信頼されるシステムは、全面的に信頼されます。 信頼されるシステムは友好的管理で管理されるということが前提となっています。 具体的に言えば、信頼されるワークステーションからファイル サービス システムの任意の有効なユーザになりすますことで、システム上のほぼすべてのファイルにアクセスできるようになります（「ルート（root）」に保持されているファイルだけは除外）。

我々の環境では、（従来的な意味での UNIX システム管理において）ワークステーションの管理は現在それを使用しているユーザの手中にあります。 マシンと同じ物理ロケーションにきわめて非友好的なユーザが居るとすべてのコンソール機能にアクセスできるという事実を考慮すると、その非友好的なユーザによる侵入が可能であると判断されるため、ワークステーションでのルート（root）パスワードの秘密鍵は設定されていません。 したがって、NFS による信頼性の解釈では、ワークステーションを完全には信頼できません。 我々の環境で適切なアクセス コントロールを可能にするために、基盤の NFS ソフトウェアに修正を行い、スキームに Kerberos を統合する

必要がありました。

Kerberos の非修正 NFS

（University of Wisconsin 提供の）初期の NFS の実装では、（NFS の用語では「クレデンシャル」と呼ばれる）各 NFS 要求に含まれた一片のデータの形式で認証が提供されていました。 このクレデンシャルには、要求者の一意のユーザ ID（UID）と要求者のメンバシップのグループ ID（GID）のリストに関する情報が含まれています。 これにより、この情報はアクセスのチェック用に NFS サーバによって使用されます。 信頼されているワークステーションと信頼されていないワークステーションの間での差異は、そのクレデンシャルが NFS サーバで受け付けられるかどうかという点です。

Kerberos によって変更される NFS

我々の環境では、クレデンシャルにワークステーションのユーザの UID だけが示されている場合にのみ、NFS サーバではワークステーションからクレデンシャルが受け付けられる必要があります。

明確なソリューションの 1 つは、UID と GID の単なるインジケータというクレデンシャルの性質を、完全装備の Kerberos 認証のデータへ変更することです。 ただし、このソリューションが適用されると、著しいパフォーマンス上の不利益をこうむることになります。 すべてのディスク読み取りと書き込みの動作を含め、クレデンシャルは NFS 操作のたびに交換されます。 各ディスク トランザクションに Kerberos 認証を取り込むことにより、トランザクションごとに（ソフトウェア内で実行される）完全装備の暗号化が数多く追加されるため、エンベロープ計算によると、許容できないパフォーマンスが発生します（カーネル アドレス空間への Kerberos ライブラリ ルーチンの配置も必要になります）。

次に述べるように、混成のアプローチが必要でした。 基本的な考え方は、NFS サーバによって、クライアント ワークステーションから受け取られたクレデンシャルを、サーバ システム上の有効な（おそらく別の）クレデンシャルへマッピングするようにすることです。 このマッピングは、NFS トランザクションごとにサーバのカーネル内で実行され、有効なカーネル クレデンシャル マッピングの確立に先立ち、Kerberos の中程度の認証で実行されるユーザレベルのプロセスによって、「マウント」時に設定されます。

この実装のために、（クライアント システムではなくサーバ システムだけで必要な）新しいシステム コールをカーネルに追加しましたが、このコールにより、クライアント ワークステーションから着信するクレデンシャルがサーバ（存在する場合）での使用に有効なクレデンシャルへマッピングされるマッピング機能の制御が提供されます。 基本的なマッピング機能によって次のタプルがマッピングされます。

このマッピング先はサーバ システム上の有効な NFS クレデンシャルです。 CLIENT-IP-ADDRESS は、クライアント システムによって提供される NFS 要求パケットから抽出されます。 注： UID-ON-CLIENT を除く、クライアント生成のクレデンシャル内のすべての情報が廃棄されます。

マッピングが存在しない場合は、設定に応じて、2 種類のどちらかの方法でサーバによる応答が行われます。 我々の友好的な設定では、アクセス権限を持たず、一意の UID を持つユーザ「nobody」のクレデンシャルに対するマッピング不可な要求がデフォルトとして設定されています。 非友好的なサーバでは、着信する NFS クレデンシャルの有効なマッピングが見つからない場合、NFS アクセス エラーが返されます。

我々の新しいシステム コールは、カーネル常駐マップでのエントリの追加と削除に使用されます。 サーバ システム上で特定の UID へマッピングするすべてのエントリをフラッシュする機能や、特定の CLIENT-IP-ADDRESS からすべてのエントリをフラッシュする機能も提供されています。

新しいトランザクション タイプである Kerberos 認証マッピング要求を受け入れるために、（サーバ システム上で NFS マウント要求が処理される）マウント デーモンを修正しました。 基本的には、マウント プロセスの一環で、ワークステーション上での UID-ON-CLIENT の表示（Kerberos オーセンティケータで暗号化）とともに、クライアント システムによって Kerberos オーセンティケータが提供されます。 サーバのマウント デーモンによって、Kerberos プリンシパル名がローカルのユーザ名に変換されます。 次に、このユーザ名は特別なファイル内で検索され、ユーザの UID と GID のリストが取得されます。 効率性のため、このファイルは鍵としてのユーザ名が含まれた ndbm データベース ファイルになっています。 この情報からは、NFS クレデンシャルが構築され、この要求の <CLIENT-IP-ADDRESS, CLIENT-UID> タプルの有効なマッピングとしてカーネルへ受け渡されます。

アンマウント（マウント解除）時には、要求がマウント デーモンへ送信され、以前に追加されたマッピングがカーネルから削除されます。 ログアウト時に要求を送信して、対象サーバ上の現在のユーザ用のマッピングを無効にすることも可能であり、これでワークステーションが次のユーザで利用可能になる前に、（必要なく）存在している残りのマッピングが削除されます。

修正 NFS での Kerberos セキュリティの意義

この実装は完全にセキュアなわけではありません。 まず第一に、ユーザ データは依然としてネットワーク上を暗号化されない形式、つまり傍受可能な形式で送信されます。 低レベルかつトランザクションごとの認証は、要求パケット内で暗号化されずに提供される <CLIENT-IP-ADDRESS, CLIENT-UID> ペアに基づいています。 この情報は偽造される可能性があるため、セキュリティが危険にさらされます。 ただし、有効なマッピングが配置されているのは、ユーザが積極的にファイルを使用している間（つまり、ログイン中）だけなので、この形式の攻撃は対象のユーザがログイン中の間だけに制限されることに注意する必要があります。 ユーザがログインしていない場合、IP アドレスの偽造によるユーザのファイルへの未承認のアクセスは許可されません。

Kerberos の参考資料

1. S.P. Miller、B.C. Neuman、J.I. Schiller、および J.H. Saltzer 著、Section E.2.1： Kerberos Authentication and Authorization System、M.I.T. Project Athena、Cambridge, Massachusetts（1987 年 12 月 21 日）
2. E. Balkovich、S.R. Lerman、および R.P. Parmelee 著、『Computing in Higher Education: The Athena Experience』、Communications of the ACM、Vol. 28(11)、pp. 1214-1224、ACM（1985 年 11 月）
3. R.M. Needham および M.D. Schroeder 著、『Using Encryption for Authentication in Large Networks of Computers』、Communications of the ACM、Vol. 21(12)、pp. 993-999（1978 年 12 月）
4. V.L. Voydock および S.T. Kent 著、『Security Mechanisms in High-Level Network Protocols』、Computing Surveys、Vol. 15(2)、ACM（1983 年 6 月）
5. National Bureau of Standards、『Data Encryption Standard』、Federal Information Processing Standards Publication 46、Government Printing Office、Washington、DC（1977 年）
6. SP Dyer 著、『Hesiod』、Usenix Conference Proceedings（1988 年冬）

7. W.J. Bryant 著、『Kerberos Programmer's Tutorial』、MIT Project Athena（執筆中）
8. W.J. Bryant 著、『Kerberos Administrator's Manual』、MIT Project Athena（執筆中）
9. G.W. Treese 著、『Berkeley Unix on 1000 Workstations： Athena Changes to 4.3BSD』Usenix Conference Proceedings（1988 年冬）
10. C.A. DellaFera、M.W. Eichin、R.S. French、D.C. Jedlinsky、J.T. Kohl、および W.E. Sommerfeld 著、『The Zephyr Notification System』、Usenix Conference Proceedings（1988 年冬）
11. M.A. Rosenstein、D.E. Geer、および P.J. Levine 著、Usenix Conference Proceedings（1988 年冬）
12. R. R. Sandberg、D. Goldberg、S. Kleiman、D. Walsh、および B. Lyon 著、『Design and Implementation of the Sun Network Filesystem』、Usenix Conference Proceedings（1985 年夏）

関連情報

- Kerberos サポート ページ
- テクニカルサポートとドキュメント - Cisco Systems