Hauptseminar Telematik

SS 2003

Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

Institut für Praktische Informatik und Medieninformatik

Globus Security

Betreuer:
Dipl. Inf. Torsten Strufe

Bearbeiter:
Dirk Harms
dirk.harms@in.stud.tu-ilmenau.de

Inhalt:

1. Einleitung

Das Globus Projekt
Allgemeine Anforderungen an Grid Security

2. Authentifikation

Zertifikate
Lokale Proxys
Remote Proxys
Validierung der Proxyzertifikate
TLS Authentifikationsprozess
Limited Proxys
Restricted Proxy

3. Autorisation

4. Transportsicherung

5. GSI-API

6. MyProxy

7. K5Cert

Online CA
User Longterm-Zertifikat Repository
User Proxy Repository

8. Community Authorisation Service (CAS)

1.Einleitung

Das Globus Projekt

Das Globus Projekt wurde 1996 vom Argonne National Laboratory, der Universität von Südkalifornien und der Universität von Chicago ins Leben gerufen. Es befasst sich mit der Entwicklung von Standards und Software zum Aufbau von Grids. Bei Grids handelt es sich um Infrastrukturen zur integrierten, kollaborativen Nutzung von Ressourcen von unterschiedlichen Organisationen. Das heißt, das es möglich sein soll, Ressourcen, die geographisch weit verteilt und im Besitz unterschiedlichster Organisationen sind, von Nutzern, die verschiedenen Organisationen angehören, auf die gleiche Weise genutzt werden können.

Aus dem Projekt ist das Globus Toolkit hervorgegangen, eine Middleware zum Aufbau von Grids. Es soll beim Aufbau der Grids und bei der Entwicklung von Anwendungen für Grid Systeme helfen. Das Globus Toolkit ist nach einem Schichtenmodell aufgebaut. Dabei gibt es von oben nach unten folgende Schichten:

- Application
- Collectiv
- Ressource
- Connectivity
- Fabric

Applikationen können dabei die Collectiv und die Ressource Schicht umgehen und direkt auf die Connectivity Schicht zugreifen. Die Connectivity Schicht kümmert sich um alle Belange des Nachrichtentransports und um die damit verbunden Sicherheitsmechanismen. Sie stellt Funktionen für die Authentifikation und Autorisation von Nutzern sowie für die Verschlüsselung und den Integritätsschutz der transportierten Daten zur Verfügung.

Allgemeine Anforderungen an Grid Security

Wie bereits erwähnt, sind die Ressourcen im Grid geographisch oft weit verteilt und befinden sich im Besitz verschiedenster Organisationen. Dies stellt eine große Herausforderung für die Grid Security dar, zumal die verschieden Organisationen meist auch unterschiedliche lokale Sicherheitsmechanismen einsetzen, mit denen die Gridsecurity zusammenarbeiten muss. Für den User sollte der Zugriff auf die Ressourcen aber unabhängig von deren Standort möglichst einfach und einheitlich sein.

Die Ressourcenanbieter wiederum dürfen durch das Anbieten von Ressourcen nicht gezwungen sein, ihre bisher verwendeten lokalen Mechanismen zu ersetzten. Deshalb muss die Gridsecurity auf den vorhandenen Systemen aufbauen und nicht diese ersetzen.

Des weiteren wollte man bei der Entwicklung des Globus Toolkit weitesgehend auf bestehenden Standards aufbauen und diese erweitern. Dies wollte man, da die bestehenden Standards meist schon gezeigt hatten, dass sie zuverlässig sind und für sie bereits ausgereifte Implementationen und API's vorhanden sind.

Auch sollten die Sicherheitsmechanismen für den Grid eine einheitliche und standardisierte Schnittstelle bieten, damit sie sich problemlos in Anwendungen integrieren lassen.

2. Authentifikation

Die Authentifikation sollte für den User möglichst einfach zu handhaben und unabhängig von der verwendeten Ressource sein. Sie sollte single sign on unterstützen, so dass der User sich nur ein einziges Mal authentifiziert und dann Zugriff auf alle Gridressourcen hat.

Programmen und Ressourcen sollte es möglich sein, im Namen des Users agieren zu können. Dies ist zum Beispiel dann nötig, wenn für eine Berechnung auf der Ressource A Daten benötigt werden, die auf Ressource B liegen. Die Ressource A sollte sich dann im Namen des Users an der Ressource B anmelden können, um so sie selben Zugriffsrechte auf die Daten zu besitzen, wie sie der User besitzt.

Es sollte eine User basierte Vertauensbeziehung gegeben sein. Das heißt, wenn ein User Zugriff auf Ressourcen von Anbieter A und gleichzeitig Zugriffsrechte auf Ressourcen von Anbieter B hat, auch Ressourcen von A im Namen des Users auf Ressourcen von B zugreifen können, ohne dass A oder B dies explizit erlauben müssen.

Zertifikate

Für die Authentifikation verwendet man bei Globus Public Key Verfahren. Bei Public Key Verfahren hat der User einen Private Key, den nur er wissen darf. Zu diesem Private Key existiert ein zugehöriger Public Key der weiterverbreitet wird. Der Private Key lässt sich nicht aus dem Public Key berechnen. Für eine Authentifikation wird dem User nun eine zufällige Ziffernfolge gegeben, die er mit seinem Private Key verschlüsseln muss. Mit Hilfe des Public Keys des Users kann dann die Folge wieder entschlüsselt werden, um so zu überprüfen, dass mit dem richtigen Private Key signiert wurde.

Dazu ist aber notwendig, dass man sich sicher sein kann, dass der Public Key, den man hat, wirklich zu dem User gehört. Zur Zuordnung der Public Keys verwendet man deshalb Zertifikate nach dem X.509 Standard, welche den Public Key einer global eindeutigen Identität für den User zuweisen. Diese Zertifikate werden dann von einer vertauenswürdigen Stelle, der sogenannten Certificate Authority (CA) signiert, die damit die Richtigkeit der für den User angegeben Identität bestätigt. Diese CA wird dann auch als Aussteller im Zertifikat festgehalten.

Die Identität des Users ist bei der Angabe im Zertifikat hierarchisch aufgebaut und enthält das Land, die Organisation, der der User angehört, und den Namen des Users.

Zertifikate werden immer nur für einen begrenzten Zeitraum ausgestellt, der im Zertifikat festgehalten wird. Ist diese Frist abgelaufen, so ist eine Nutzung des Zertifikates nicht mehr möglich. Ein solches X.509 Zertifikat könnte dann wie folgt aussehen:

```
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 28 (0x1c)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=US, O=Globus, CN=Globus Certification
Authority
Validity
Not Before: Apr 22 19:21:50 1998 GMT
Not After : Apr 22 19:21:50 1999 GMT
Subject: C=US, O=Globus, O=NACI, OU=SDSC, CN=Richard
Frost
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:bf:4c:9b:ae:51:e5:ad:ac:54:4f:12:52:3a:69:
<snip>
b4:e1:54:e7:87:57:b7:d0:61
Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
59:86:6e:df:dd:94:5d:26:f5:23:c1:89:83:8e:3c:97:fc:d8:
<snip>
8d:cd:7c:7e:49:68:15:7e:5f:24:23:54:ca:a2:27:f1:35:17:
```

Für die Signatur nimmt man im allgemeinen einen md5 Hash über das Zertifikat und verschlüsselt diesen mit Hilfe des RSA Algorithmussees mit dem Private Key. Diese Signatur wird dann an das Zertifikat angehängt. Zur Überprüfung eines solchen Zertifikates wird dann wieder ein MD5 Hash gebildet. Die am Zertifikat angehängte Signatur wird mit dem Public Key des Ausstellers entschlüsselt und mit dem gerade gebildeten Hash verglichen. Stimmen beide überein, kann man sich sicher sein, dass das Zertifikat nicht verändert wurde. Jede Veränderung des Zertifikates ist also unmöglich, da dann die Hashes nicht mehr übereinstimmen würden. Auch kann kein neuer Hash angehängt werden, da er mit dem Private Key der CA verschlüsselt werden müsste, damit er mit ihrem Public Key richtig entschlüsselt wird.

Die Server, an denen man sich mit dem Zertifikat anmelden will, müssen natürlich die als Aussteller angegebene CA kennen. Dazu gibt es auch für die CA Zertifikate, die den Public-Key der CA enthalten und auf den Servern vorhanden sein müssen. Außerdem enthält das Zertifikat einen Gültigkeitszeitraum, für den es von der CA ausgestellt wurde.

Bei normalen Public-Key-Infrastrukturen werden die Userzertifikate meist von den CA's aufbewahrt und bei der Authentifikation eines Users an einem Server dort abgerufen. Damit ist es auch möglich, Zertifikate vor deren Ablaufdatum zurückzuziehen. Dies ist bei Globus nicht der Fall, hier reicht der User sein Zertifikat bei der Authentifikation selbst mit an den Server, was ein Zurückziehen der Zertifikate nicht ermöglicht.

Zu Authentifikation sendet der User dann also nur sein Zertifikat mit dem Public Key an den Server und signiert die ihm genannte Zahlenfolge mit dem Private Key. Der Server überprüft dann die Gültigkeit des Zertifikates des Users und die Signatur der Zahlenfolge und stellt so die Identität des Users sicher.

Problematisch bei der Verwendung der Zertifikate ist noch, dass nicht unbedingt alle Ressourcen den selben CA's vertrauen müssen. Dadurch kann es notwendig sein, dass ein User für unterschiedliche Ressourcen verschiedene Zertifikate benötigt. Dies verhindert dann wieder den gewünschten einheitlichen Zugriff. Eine universelle Lösung für dieses Problem existiert noch nicht. Ein Ansatz zur Lösung ist der MyProxy Server, auf den später noch eingegangen wird.

Lokale Proxys

Da sich jeder, der in Besitz des Private Key's gelangt, als der User authentifizieren könnte, sollte der private Key nicht unverschlüsselt auf der Festplatte liegen, sondern mit einem Passwort verschlüsselt werden. Besser noch wird er auf einem austauschbaren Medium wie zum Beispiel einer Smartcard aufbewahrt, das nur während der Authentifizierung in den Rechner gesteckt wird. Dies bedeutet aber, dass der User vor jeder Verwendung und somit bei jedem Anmelden an einer Ressource das Passwort eingeben und eventuell noch die Smartcard einlegen müsste.

Um dies zu vermeiden, verwendet man lokale Proxys. Für einen lokalen Proxy wird ein neuer Private Key und der dazugehörige Public Key generiert. Dann wird ein neues Zertifikat angelegt, welches den Public Key enthält. Die Identität im neuen Zertifikat entspricht dann der Identität des Nutzers, der den Proxy anlegt, erweitert um einen Hinweis, dass es sich um einen Proxy handelt. Für das obige Beispiel wäre die Identität des Proxys also:

C=US, O=Globus, O=NACI, OU=SDSC, CN=Richard Frost, CN=Proxy

Das neue Zertifikat wird danach vom User mit seinem private Key signiert. Somit ist auch für dieses Zertifikat sichergestellt, dass es nicht unbemerkt verändert wird. Für alle Authentifikationsvorgänge wird danach nur noch der lokale Proxy, also der neu erstellte Private Key und das neue Zertifikat verwendet. Das Zertifikat für den lokalen Proxy hat immer nur eine kurze Gültigkeitsdauer, standardmäßig sind dies zwölf Stunden.

Der Private Key des Proxys wird unverschlüsselt aufbewahrt und ist nur durch den Zugriffschutz des Dateisystems geschützt. Da er aber nur kurzfristig gültig ist, ist die Gefahr hierbei deutlich geringer als beim unverschlüsselten Aufbewahren des Private Keys des Users. Der User kann den erstellten Proxy auch jederzeit wieder zerstören, allerdings zerstört dies nur den lokalen Proxy und keine weiteren Proxys, die eventuell von ihm delegiert wurden.

Remote Proxys

Nun sollen sich aber auch Ressourcen im Namen des Users authentifizieren können. Dazu verwendet man Remote Proxys. Remote Proxys funktionieren vom Prinzip her genauso wie lokale Proxys. Sie werden allerdings auf anderen Rechnern angelegt und nicht vom Nutzer selbst, sondern von einem bereits bestehenden Lokalen oder Remote Proxy.

Die Erstellung eines Remote Proxys läuft wie folgt ab:

1. Die Ressource, welche einen Proxy delegiert bekommen möchte, sendet eine „Delegetion Init“ Nachricht an den Rechner, von dem aus der Proxy delegiert werden soll.
2. Ist dieser bereit, der Ressource einen Proxy zu delegieren, antwortet sie mit einem „Delegetion Init“.
3. Die Ressource, welche den Proxy möchte, generiert ein neues Schlüsselpaar.
4. Der Public-Key des neuen Schlüsselpaares wird nun in einem „Credential Request“ an den Rechner gesendet, der bereits einen Proxy hat.
5. Der Empfänger generiert ein neues Zertifikat mit dem empfangenen Public-Key, signiert dieses mit dem Private Key seines Proxys und sendet es in einer „Delegtion Complete“ Nachricht zurück.

Die Ressource hat somit jetzt einen Proxy mit Private Key und einem signierten Zertifikat, mit dem es sich im Namen des Users, der den ersten Proxy angelegt hatte, authentifizieren kann. Für die Identität des neuen Zertifikates wird hierbei die Identität des alten Proxys genommen und um eine Komponente erweitert. Diese Komponente sollte für diesen Rechner und den Proxy eindeutig sein, so dass keine zwei Zertifikate mit dem gleichen Namen ausgestellt werden.

Als Aussteller wird in dem neuen Proxyzertifikat die Identität des delegierenden Proxys eingetragen. Auch darf die Gültigkeit des neuen Zertifikates die Gültigkeit des ursprünglichen Proxys nicht überschreiten. Außerdem wird in jedem Proxyzertifikat vermerkt, die wievielte Delegation dieser Proxy bereits darstellt.

Bei der Delegation des Proxys sollten allerdings eine Verbindung zwischen den beiden Rechnern bestehen, bei der die Integrität der Daten geschützt wird. Dies ist vor allem beim Senden des Public Key vom Anfordernden an den Delegierenden wichtig, da ein Angreifer hier sonst den neu erstellten Public Key durch einen eigenen ersetzten könnte und so ein gültiges Zertifikat für sein eigenes Schlüsselpaar hätte. Eine Verschlüsselung der Verbindung ist nicht notwendig, da ja nur der Public Key über das Netz gesendet wird, den ja jeder wissen darf.

Validierung der Proxyzertifikate

Nun ist es allerdings notwendig, dass bei der Anmeldung an einer Ressource mit Hilfe eines Proxy Zertifikates überprüft wird, dass dieses auch gültig ist. Im Gegensatz zu den Userzertifikaten ist das Proxyzertifikat nicht von einer CA unterzeichnet, deren Public Key der Ressource bekannt ist, sondern nur von einem anderen Proxy oder dem User.

Die Clients schicken bei der Authentifizierung immer die komplette Zertifikatkette, also das Userzertifikat und alle erstellten Proxyzertifikate bis hin zu dem des momentanen Proxy an den Server.

Die Überprüfung der Gültigkeit der Zertifikatkette und damit der Gültigkeit des letzten Proxys läuft dann wie folgt ab:

1. Überprüfen, ob das ursprüngliche Zertifikat des Nutzers von einer vertrauenswürdigen CA ausgestellt wurde, und mit dem bekannten Public Key dieser überprüfen, dass das Zertifikat gültig ist, also die Signatur der CA stimmt.
2. Überprüfen, dass der Aussteller des ersten Proxyzertifikates der User ist und die Identität eine Erweiterung der Identität des Users ist. Mit dem Public Key aus dem Zertifikat des Users überprüfen, ob dieses erste Proxyzertifikat mit dem Private Key des Users signiert und nicht verändert wurde.
3. Für jedes Proxyzertifikat wird überprüft, dass der Aussteller der letzte Proxy ist, die Identität eine Erweiterung der Identität des letzten Proxys ist, die Signatur mit dem Private Key des letzten Proxys ausgestellt wurde und das Zertifikat nicht nachträglich verändert wurde.

Durch die schrittweise Überprüfung kann nach und nach jedes Zertifikat als gültig angesehen werden, bis man bei dem letzten Zertifikat angelangt ist, welches dem Proxy gehört ,der sich authentifizieren will. Damit kann man dann davon ausgehen, dass der Proxy das Recht hat, im Namen des Users zu agieren und die Authentifikation vornehmen.

TLS Authentifikationsprozess

Die eigentliche Authentifikation geschieht mit Hilfe des Transport Layer Security (TLS) Authentifikationsprotokolls. Ziel dabei ist es, dass sich Client und Server gegenseitig authentifizieren, das heißt es, wird nicht nur überprüft, dass der Client der ist für den er sich ausgibt, sondern auch, dass es der richtige Server ist, an dem man sich anmeldet, und nicht nur jemand, der vorgibt, dieser Server zu sein.

Dabei findet folgender Kommunikationsablauf statt:

1. „Client Hello“ vom Client zum Server

   Mit der Client Hello Nachricht gibt der Client dem Server bekannt, dass er mit der Authentifikation beginnen möchte. Dabei sendet der Client mit, welche Methoden er für die Authentifikation wünscht und unterstützt, und welche Parameter, also Verschlüsselungstyp, Schlüssellänge und Integritätsschutz, für die später aufgebaute Verbindung gewünscht werden. Außerdem sendet er eine Zufallszahlenfolge mit, die später mit für die Generierung des Schlüssels für die aufzubauende Verbindung genutzt wird.

2. „Server Hello“ vom Server zum Client

   In der Server Hello Nachricht schickt der Server zurück, welche der vom Client vorgeschlagenen Authentifizierungsmethoden er bevorzugt, und welche Verschlüsselungsalgorithmen für die spätere Verbindung möglich sind. Auch er sendet eine Zufallszahlenfolge für die Schlüsselgenerierung mit.

3. „Certifikate“ vom Server zum Client

   In der Certifikate Nachricht sendet der Server sein eigenes Zertifikat an den Client, damit dieser später den darin enthalten Public Key verwenden kann, um die Identität des Servers sicherzustellen.

4. „Certifikate Request“ vom Server zum Client

   Mit der Certifikate Request Nachricht teilt der Server dem Client mit, dass dieser sich authentifizieren muss. Dabei sendet er auch die Identitäten der CA’s mit, deren .Zertifikate er akzeptiert.

5. „Server Hello Done“ vom Server zum Client

   Mit dem Server Hello Done gibt der Server dem Client bekannt, dass er erst einmal mit der Authentifikation fertig ist und nun die Authentifikation des Clients erwartet.

6. „Certificate“ vom Client zum Server

   Mit der Certifikate Message schickt der Client nun sein Zertifikat bzw. die Zertifikatkette, falls es sich um einen Proxy handelt, an den Server.

7. „ClientKeyExchange“ vom Client zum Server

   Für die ClientKeyExchange generiert der Client wiederum eine Zufallszahlenfolge, die später mit für den Verbindungsschlüssel verwendet werden soll. Diese wird aber nicht wie die ersten beiden Zufallszahlen unverschlüsselt übertragen, sondern wird mit dem Public Key aus dem Zertifikat des Servers verschlüsselt. Damit ist sichergestellt, dass nur der Server, der im Besitz des Private-Key zu dem gesendeten Zertifikat ist, auch die Möglichkeit hat, die Nachricht zu entschlüsseln und an den Verbindungsschlüssel für die folgende Datenübertragung kommt.

8. „CertifikateVerify“ vom Client zum Server

   Für die CertifikateVerify Nachricht bildet der Client einen Hash über alle vorher mit dem Server ausgetauschten Nachrichten und signiert diese mit seinem Private Key. Der Server kann dann mit dem Public Key aus dem Zertifikat des Clients überprüfen, dass die Signatur wirklich mit dem dazugehörigen Private Key generiert wurde. Dadurch, dass der Hash über alle vorangegangen Nachrichten gebildet wurde und diese auch eine vom Server generierte Zufallszahl enthalten, ist sichergestellt, dass sich ein Angreifer nicht durch das erneute Senden früher mitgehörter Authentifikationen anmelden kann.

9. „ChangeChipperSpec“ vom Client zu Server

   Mit ChangeChipperSpec gibt der Client bekannt, dass er nun zur Verschlüsselung mit dem ausgetauschten Schlüssel gemäss den vereinbarten Verbindungsparametern übergehen wird.

10. „Finished“ vom Client zum Server

    Mit der Finished-Nachricht sendet der Client dem Server noch einmal einen Hash über alle ausgetauschten Nachrichten.

11. „ChangeChipperSpec“ vom Server zum Client

    Auch der Server signalisiert dem Client mit ChangeChipperSpec den Wechsel auf die verschlüsselte Verbindung.

12. „Finished“ vom Server zum Client

    Genauso wie der Client sendet der Server beim Finished noch einmal einen Hash über alle zuvor ausgetauschten Nachrichten.

Wenn dieser Ablauf komplett ist, ist sichergestellt, dass sowohl Client als auch Server die sind, für die sie sich ausgeben, und es besteht einen verschlüsselte Verbindung mit einem gemeinsamen symmetrischen Schlüssel. Dieser Schlüssel wurde aus den Zufallszahlenfolgen in den beiden Hello Nachrichten und der aus der ClientKeyExchange Nachricht erstellt. Da die ClientKeyExchange Nachricht mit dem Public Key des Servers verschlüsselt war, kennt auch jemand, der die ganze Kommunikation mitgehört hat, den gemeinsamen Schlüssel nicht. Nur der Server mit dem passende Private-Key kann die Nachricht entschlüsseln und so den selben symmetrischen Schlüssel berechnen wie der Client.

Limited Proxys

Nachteil der Remote Proxys ist, dass ein Angreifer, der Zugriff auf eine Ressource erlangt hat, mit Hilfe der auf der Ressource vorhandenen Remote Proxys auf alle andere Ressourcen zugreifen kann, auf die auch die ursprünglichen Ersteller der ersten Proxys Zugriff haben.

Um dies einzuschränken, verwendet man Limited Proxys. So kann beim Delegieren entschieden werden, dass eine Ressource für ihre Aufgabe keinen Full sondern nur einen Limited Proxy benötigt und nur ein solcher delegiert wird. Bei einer Authentifikation an einer anderen Ressource kann diese dann entscheiden, ob sie einen Limited Proxy akzeptiert.

Im allgemeinen akzeptieren GridFtp Server auch Limited Proxys, während ein Gatekeeper keine Limited Proxys akzeptiert. Somit kann an einen Rechner, der nur ein paar Daten holen, mit diesen einen Berechnung durchführen und die Ergebnisse wieder auf einem GridFtp Server speichern soll, auch ein Limited Proxy delegiert werden. Würde dieser dann kompromittiert werden, wäre zwar ein Zugriff auf den GridFtp möglich, aber es könnten nicht über den Gatekeeper neue Aufgaben in den Grid gegeben werden. Damit könnte dann zum Beispiel verhindert werden, dass der Angreifer einen Auftrag in den Grid gibt, bei dem er möglichst viele Rechner anfordert und auf diesen Programme starten lässt, die ihm Zugang zu diesen Rechnern verschaffen. Mit dem Limited Proxy kann also schon einmal erreicht werden, dass durch einen kompromittierte Ressource weitere Ressourcen bedroht sind. Allerdings hat der Angreifer immer noch Zugriff auf alle Daten auf den GridFtp Servern.

Restricted Proxy

Die Limited Proxys erlauben nur eine viel zu grobe Einschränkung der Zugriffsrechte, welche nicht wirklich ausreichend ist, um sich vor kompromittierten Ressourcen zu schützen. Deshalb führt man Restricted Proxys ein. Bei Restricted Proxys ist im Zertifikat eine Policy enthalten, wofür dieser Proxy verwendet werden darf.

So könnte für eine Rechenaufgabe im Zertifikat genau festgelegt werden, welche Daten mit dem Proxy vom GridFtp geholt werden dürfen, welches Programm ausgeführt werden darf und wohin die Ergebnisse gespeichert werden sollen. Ein Angreifer hätte damit nur noch Zugriff auf die für die eigentliche Aufgabe benötigten Daten und nicht auf sämtliche Daten auf allen GridFtp Servern, auf die der User Zugriff hat. Es kann auch festgelegt sein, dass auf die zu holenden Daten nur lesend zugegriffen werden darf. Damit kann der Angreifer also maximal die Ergebnisse dieser einen Berechnung zerstören und an die Ausgangsdaten dieser Berechnung gelangen, aber sonst keinen weiteren Schaden im Grid anrichten.

Die Policy wird im Zertifikat in einen Container eingebettet. Erst in diesem Container wird dann spezifiziert, in welcher Sprache die Policy abgefasst ist. Damit ist die Policy Sprache also nicht schon von vorn herein durch das Zertifikatformat festgelegt, sondern kann auch später noch erweitert oder ausgetauscht werden.

Wichtig ist dabei, dass die Ressourcen auf dehnen Aktionen ausgeführt werden sollen, die Policy Sprache verstehen und die im Zertifikat enthaltene Policy durchsetzen.

3. Autorisation

Nach der erfolgreichen Authentifikation an einer Ressource ist es notwendig, dem Nutzer anhand seiner Globalen ID Nutzerrechte zuzuweisen. Dies geschieht im Normalfall mit Hilfe einer Gridmap Datei. In dieser werden die Globalen ID's aller Nutzer, die Zugriff auf die Ressource haben sollen, eingetragen und ihnen ein Nutzer aus der lokalen Domäne zugeordnet.

Dies bedeutet aber, dass jeder User, der Zugriff auf eine Ressource haben soll, auf dieser explizit eingetragen werden muss. Da oft viele verschiedene Forschungsgruppen aus verschiedensten Organisationen auf die Ressourcen zugreifen, bedeutet dies einen sehr hohen Aufwand. Es wäre günstiger wenn der Ressourcenanbieter nur noch für ganze Gruppen Rechte definieren müsste und diese Gruppen sich dann selbst verwalten. Dies wird mit CAS versucht, worauf später noch eingegangen wird.

Für Kerberos Domänen steht SSLK5 und PKINIT zur Verfügung. Bei diesen Systemen handelt es sich um Server, die es dem Client ermöglichen, mit seinem Proxy Zertifikat ein Kerberos Ticket zu bekommen. Dieses Ticket dient dann der weiteren Authentifikation und Autorisation innerhalb der lokalen Kerberos Domäne. Dabei ist SSLK5 die derzeit genutzte simplere Variante, welche mit derzeitigen Domänekontrollern verwendbar ist. Bei PKINIT handelt es sich um einen IETF Draft. Es soll SSLK5 ablösen, sobald es in Domänekontrollern verfügbar ist.

4.Transportsicherung

Da es sich bei den im Grid ausgetauschten Daten auch um vertrauliche Informationen handeln kann, ist es natürlich auch notwendig, die Kommunikation zu schützen. Dabei ist es wichtig, dass Anwendungen selbst entscheiden können, ob sie eine Transportsicherung wünschen und in welcher Form. Da in Grids oft sehr große Datenmengen ausgetauscht werden, wäre es unsinnig, alle Daten von vorneherein zu verschlüsseln, egal ob es notwendig ist oder nicht, da dies viel Rechenleistung benötigen kann und unter Umständen Performanceeinbußen bei der Übertragung mit sich bringt.

Des weiteren sollte die Transportsicherung verschiedenste Kommunikationsprotokolle unterstützen und nicht nur IP Netzwerke bzw. TCP/IP. Dies ist notwendig, da in Grids oft sehr hohe Datenmengen transportiert werden müssen und gewisse Echtzeitanforderungen für den Datentransport existieren können, könnte dies andere Netzwerkstrukturen wie zum Beispiel ATM erfordern.

Zur Transportsicherung wird bei Globus TLS verwendet. Da TLS schon zur Authentifikation verwendet wird, kann so die einmal aufgebaute Verbindung gleich weiter genutzt werden, und es ist kein weiterer Schlüsselaustausch nötig. Des weiteren unterstützt TLS auch verschiedenen Verschlüsselungsstärken und den Integritätsschutz der Nachrichten, so dass die verwendete Transportsicherung an die Anwendung anpassbar ist.

Die Verschlüsselung geschieht bei TLS mittels symmetrischer Verfahren. Dabei verwenden beide Kommunikationspartner den selben Schlüssel. Dieser Schlüssel wird während des Authentifikationsprozesses ausgetauscht und dabei mittels unsymmetrischer Verschlüsselung geschützt. Unterstützt werden bei TLS die Verschlüsselungsverfahren DES, Triple DES, IDEA, RC4, RC2. Für die Sicherung der Datenintegrität stehen md5 Checksummen und SHA zur Verfügung. Der Vorteil der symetrischen Verschlüsselung gegenüber den unsymmetrischen Public-Key Verfahren ist, dass die Verschlüsselung und Entschlüsselung der Daten deutlich schneller geht.

5.GSI-API

Die GSI-API soll Entwicklern von Gridanwendungen einen einfachen Weg bieten, die Sicherheitsmechanismen von Globus in ihre Anwendung zu integrieren. Sie basiert auf der Generic Security Services API (GSS-API). Diese ist ein IETF Standard, um Authentifikation, Nachrichtenintegrität und Vertraulichkeit für die Kommunikation zu Anwendungen hinzuzufügen. Diese wurde um Funktionen für Single Sign On und Proxy Delegation erweitert.

Da die GSS und damit auch die GSI-API für Entwickler recht kompliziert zu nutzen ist, bietet man die globus_gss_assist API an, welche die Entwicklung vereinfachen soll. Dabei sind die Sicherheitsmechanismen weiterhin von der Kommunikationsmethode getrennt, und die API kann somit unabhängig vom verwendeten Kommunikationsprotokoll verwendet werden.

Für noch einfachere Entwicklung wird die globus_io API angeboten, hier fällt aber die Trennung zur Kommunikationsmethode weg. Dies rührt daher, dass globus_io als API zur Entwicklung der Kommunikation zwischen Grid Anwendungen gedacht ist, und als solche die entsprechenden Sicherheitsmechanismen enthält und nicht als reine Verschlüsselungs-API. Die globus_io API bietet Sockets an, die den BSD Sockets sehr ähnlich sind und sowohl synchrone als auch asynchrone Kommunikation ermöglichen. Über Parameter können die gewünschten Verschlüsselungs- und Quality of Service Merkmale eingestellt werden. Damit können Programmierer auch Grid Anwenndungen mit den gewohnten Socket Primitiven programmieren und müssen sich nicht erst in neue API’s einarbeiten.

6.MyProxy

Man möchte mit dem Globus Toolkit auch Web-based-computing ermöglichen und so die Nutzung des Grids noch einfacher gestalten. Dabei existiert allerdings das Problem, dass gängige Browser zwar TLS zur Authentifizierung unterstützen, aber keine Proxy Delegation.

Um dies zu umgehen, hat man MyProxy Server eingeführt. Dabei delegiert der Nutzer einen temporären Proxy an den MyProxy Server. Dies geschieht zusammen mit einem Tag und einem Passwort. Wird nun per Web ein Auftrag in den Grid gestellt, geschieht dies zusammen mit dem Tag und dem Passwort. Damit kann sich dann die entsprechende Ressource im Grid vom MyProxy Server einen Proxy delegieren lassen.

Es erfordert allerdings einen extra Client, um den Proxy vorher zu delegieren. Um dies zu vermeiden, soll es in Zukunft ermöglicht werden, dass User ihren Private Key auf dem MyProxy Server ablegen. Dieser kann dann mit Hilfe des Private Keys selber temporäre Proxys anlegen und delegieren, wenn Anforderungen gestellt werden. Dabei soll es auch möglich sein, dass User mehrere Keys und Zertifikate ablegen, die von unterschiedlichen CA's signiert wurden und so für unterschiedliche Ressourcen nutzbar sind. Bei einer Anfrage durch einen Ressource soll dann immer automatisch das für diese Ressource passenden Zertifikat ausgesucht werden. Damit ist es kein Problem mehr, wenn unterschiedliche Ressourcen nicht der selben CA vertrauen.

7.K5Cert

Mit Hilfe von K5Cert soll es Usern, die sich bereits an ihrer Kerberos Domäne angemeldet haben, ermöglicht werden, ohne erneute Anmeldung auf Grid Ressourcen zuzugreifen. Dabei gibt es drei verschiedene Varianten.

Online CA

Bei der Online CA generiert der K5Cert bei jeder Anforderung selbst ein Schlüsselpaar, stellt ein Zertifikate für den neuen Public Key aus, und signiert dieses selbst. Damit fungiert der K5Cert Server auch als CA. Dies bedeutet natürlich, dass alle Ressourcen, auf die zugegriffen werden soll, den Server auch als CA anerkennen müssen.

Des weiteren besteht natürlich die Gefahr, dass ein Angreifer in den Besitz des privaten Schlüssels der CA kommt, da dieser ja ständig verfügbar sein muss, und damit sich selbst Zertifikate ausstellt. Die CA muss auch ständig am Netz hängen da sie ja on demand Zertifikate ausstellen muss, was bei herkömmlichen CA's nicht der Fall ist und das Risiko deutlich erhöht.

User Longterm-Zertifikat Repository

Beim User Longterm-Zertifikat Repository legen die User ihre privaten Keys und die dazugehörigen Zertifikate auf dem Server ab. Diese werden dann bei Anforderungen vom K5Cert Server verwendet, um Proxys zu delegieren. Das hat im Gegensatz zur Online CA den Vorteil, dass die Zertifikate von irgendeiner anderen CA ausgestellt sein können, die von allen Ressourcen akzeptiert wird.

Ein erfolgreicher Angreifer würde allerdings dabei auch in den Besitz aller privaten Keys gelangen und hätte damit wieder unbeschränkten Zugriff auf alle Ressourcen. Allerdings könnten die Keys auf dem zentralen Rechner auch besser geschützt sein als bei den einzelnen Usern, da diese auch sorglos mit dem Key umgehen könnten, was durch die zentrale Lagerung ausgeschlossen ist.

User Proxy Repository

Beim User Proxy Repository wird vom User mit seinem privaten Key ein temporär begrenzter Proxy generiert, der auf dem K5Cert abgelegt wird. Von diesem Proxy aus delegiert der K5Cert bei Anforderungen dann weitere Proxys. Dies hat den Vorteil, dass ein Angreifer, der Zugriff auf den K5Cert Server erlangt, nur Zugriff auf die temporären Proxys hat. Damit ist sein Zugriff auf die Gridressourcen auf die Rechte der gerade aktiven Nutzer und die Lebensdauer der temporären Proxys beschränkt.

Später könnten auch Restricted Proxys verwendet werden, die nur die Rechte beinhalten die aktuell benötigt werden, was die Möglichkeiten eines erfolgreichen Angreifers weiter einschränken würde. Nachteil dieser Methode ist, dass der User immer wieder selbst Proxys delegieren und sich selbst um die Sicherheit seines Private Keys kümmern muss.

8.Community Authorisation Service (CAS)

CAS will sich dem Problem der Autorisation der Nutzer annehmen. So ist die Autorisation durch Gridmap Files sehr unzureichend. Dabei lassen sich die Globalen User nur auf lokale abbilden. Die Einschränkung der Nutzerrechte ist auf die Möglichkeiten des darunter liegenden lokalen Systems begrenzt und muss auch für jedes System neu und unter Umständen auf andere Weise definiert werden.

Auch sind keine Hierarchien möglich, um die Struktur der Institutionen oder Forschungsgruppen, welche die Ressourcen verwenden, abzubilden und so ein gewisses Selbstmanagement zu ermöglichen. So muss jeder hinzukommende Nutzer extra von jedem Ressourcenanbieter eingetragen werden, was den Verwaltungsaufwand für die Ressourcen enorm erhöht.

Bei CAS soll die Verwaltung der Zugriffsrechte zentralisiert werden. Die Gruppen sollen in der Lage sein, die Rechte für die einzelnen Gruppenmitglieder selbst zuzuweisen. Die Ressourcenanbieter sollen nur noch eine lokale Policy definieren müssen, die dann für ganze Gruppen gilt.

Dafür werden auf dem CAS Server alle Nutzer eingetragen und in einzelne Gruppen eingeteilt, welche auch wieder in Untergruppen besitzen können. Dabei braucht der Administrator nur die Rechte für die Gruppenleiter zuweisen, diese können dann selbst Nutzer und Untergruppen anlegen und diesen Rechte übertragen. Außerdem werden alle verfügbaren Ressourcen auf dem Server eingetragen und können auch wieder in Gruppen zusammengefasst werden.

Die einzelnen Policys, die gespeichert werden, drücken dann aus, welcher User auf welcher Ressource welche Aktionen ausführen darf. Dies kann beinhalten, welche Programme gestartet werden dürfen, auf welche Daten zugegriffen werden darf, und ob eigenen Programme auf der Ressource ausgeführt werden dürfen.

Will ein Nutzer nun eine Grid Ressource nutzen, findet folgender Ablauf statt:

1. „CAS Request“ vom User zum CAS Server

   Der CAS Request enthält eine Auflistung der Rechte, die der Nutzer für seine Aufgabe benötigt und vom CAS Server gewährt bekommen möchte.

2. „CAS Reply“ vom CAS Server zum User

   Entsprechen die vom Nutzer angeforderten Rechte der auf dem CAS Server eingetragenen Policy für den Nutzer schickt dieser ihm den CAS Reply mit einer Capability. Bei der Capability handelt es sich um ein vom CAS Server signiertes Zertifikat, in dem eine Policy eingebettet ist, die den angeforderten Rechten entspricht.

3. „Ressource Request“ vom User an Ressource

   Im Ressource Request stellt der User nun seine Anforderung an eine Ressource und sendet dabei die vom CAS Server ausgestellte Capability mit. Die Ressource überprüft nun, ob das Capability Zertifikat gültig ist, und ob die Anforderung des Users sowohl der im Zertifikat eingebetteten als auch der lokalen Policy entspricht.

4. „Ressource Reply“ von Ressource an User

   Hat die Anforderung den Policys entsprochen und der User kann seine Aktion ausführen, wird dies von der Ressource mit einem Ressource Reply bestätigt.

Außer einer besseren Verwaltung der Nutzerrechte bietet CAS mit den eingebetteten Policys auch den schon mit den Resricted Proxys angestrebten Schutz vor kompromittierten Ressourcen.

Um die Policy in das Zertifikat einbetten zu können, benötigt man aber eine Policy Sprache, die dann auch von den Ressourcen interpretiert werden muss. Die Policy Sprache ist durch das Zertifikatformat nicht festgelegt und jederzeit austauschbar. Man hat sich auch nicht auf eine spezielle Sprache festgelegt, die man unterstützen will. Im Moment sind als Sprachen Controlled English, Ponder und Authorisation Spezification Language (ASL) vorgesehen.

Die derzeitigen Implementationen beschränken sich allerdings auf eine Policy Sprache, die aus einer simplen Auflistung von Rechten besteht. Jedes dieser Rechte enthält eine Liste von Objekten, und die darauf erlaubten Aktionen.

Um beim Erstellen eigener CAS Anwendungen das Auswerten der Policys auf den Ressourcen zu vereinfachen, wird die Policy Evaluation API angeboten. Diese ist eine Erweiterung der Generic Authorisation and Access Controll API.

Derzeit sind als CAS fähige Anwendungen nur der CAS Server selbst, Anwendungen zur Verwaltung der Policys auf dem CAS Server, CAS Clients und ein CAS fähiger FTP Server verfügbar.

Quellen:

Ian Foster, Carl Kesselman, Steven Tuecke
„The Anatomy of the Grid"
http://www.globus.org/research/papers/anatomy.pdf

Randy Butler Von Welch, Douglas Engert, Ian Foster, Steven Tuecke, John Volmer, Carl Kesselman
„A National-Scale Authentication Infrastructure"
http://www.globus.org/documentation/incoming/butler.pdf

Laura Pearlman, Von Welch, Ian Foster, Carl Kesselman, Steven Tuecke
„A Community Authorization Service for Group Collaboration"
http://www.globus.org/security/CAS/CAS_2002_Revised.pdf

„Globus Toolkit Developer Tutorial - Grid Security Infrastructure"
http://www.globus.org/about/events/US_tutorial/slides/Dev-04-Security1.pdf

S. Tuecke
„Grid Security Infrastructure (GSI) Roadmap"
http://www.gridforum.org/security/ggf1_2001-03/drafts/draft-ggf-gsi-roadmap-02.doc

S. Tuecke, D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman, C. Kesselman
„Internet X.509 Public Key Infrastructure Proxy Certificate Profile"
http://www.globus.org/security/standards/draft-ietf-pkix-proxy-06.pdf

Adam Hess, Jared Jacobson, Hyrum Mills, Ryan Wamsley, Kent E. Seamons, Bryan Smith
„Advanced Client/Server Authentication in TLS"
http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/hess.pdf

„Introduction to The Globus Toolkit"
http://www.globus.org/about/events/US_tutorial/slides/Intro-02-Globus2.pdf