SECURITY AND PRIVACY IN AN ENTERPRISE SEARCH INFRASTRUCTURE FOR MOBILE DEVICES

Christian P. Praher
Institute for Information Processing and Microprocessor Technology (FIM)
Johannes Kepler University Linz

Jakob F. Praher
Mindbreeze Software GmbH

Abstract
With the advent of new powerful smart mobile devices that combine the properties of up to now individual appliances like cell phone, Personal Digital Assistant (PDA), or consumer electronic device, new mobile use cases emerge. Especially in the field of enterprise computing, many new fields of application and possibilities of how to incorporate mobile clients into the corporate environment arise with these powerful mobile devices.
This paper first provides a quick overview about the evolution of mobile computing. The focus of it lies in the issues of security and privacy in the context of an enterprise search environment with regard to mobile computing. Firstly, an overview about a traditional enterprise search security architecture that is only operated within secure corporate walls is given. Then the problems that emerge from integrating mobile clients accessing enterprise information from an insecure network like the Internet are highlighted, and general ways of how to solve these issues are presented. Finally, a web Single Sign On (SSO) enterprise search infrastructure for mobile devices on the basis of common web technologies and the Security Assertion Markup Language (SAML) is proposed.

1. Evolution of Mobile Computing

Mobile computing has become a real catchphrase within information technology over the past few years and the term may denote many different things ranging from portable desktop computers, over mobile phones to embedded, ubiquitous computing devices. Within this paper, mobile computing generally refers to *smartphones*, which represent smart mobile devices that combine the following three characteristics into one single device, that have as yet only been found in individual appliances[12]:

- **Communication**
  As their name implies, smart*phones* are phone centric devices, which means that they comprise all the features usually found in a normal mobile phone. Smartphones also have the property of being heavily interconnected. Besides the traditional cellular networks primarily used for voice communication, smartphones traditionally also possess wireless Local Area Network (LAN) interfaces for high speed data transmission, as well as Body Area Network (BAN) connections like Bluetooth or infrared, utilized for ad hoc networking and as a cable replacement for attaching peripherals to the device.

- **Computing**
  A distinct feature that sets smartphones apart from traditional mobile phones or feature phones[1] , is that they possess a complex operating system that can be extended through third-party applications. Also, smartphones offer Personal Information Manager (PIM) functionalities like calendar, address book, task-lists, etc., which were usually found in Personal Digital Assistants (PDA) and pocket PCs.

- **Consumer Electronics**
  With the ongoing miniaturization of electronic devices, more and more functionalities of typical consumer electronic appliances find their way into mobile phones. Amongst others, typical modern smartphones comprise the features of a digital camera, mp3-player, Global Positioning System (GPS) receiver or accelerometer for measuring acceleration due to gravity.

1.1. Convergence of Technologies

Accompanied by the technical convergence of multiple appliances into one single mobile device, is a convergence of mobile application development with desktop application development.
Three of the most important ways of creating applications for smartphones are *mobile web applications*, *Java Micro Edition (ME) applications* and *native applications*, with each technology having its unique strengths and weaknesses[9].

1.1.1. Native Applications

Generally, native applications as first class citizens of the mobile operating system, offer the best access to information exposed by the operating system, like e.g. the device context, PIM data, or any other accessible data. By being closest to the hardware, they usually also offer the best performance. Drawbacks of native applications are that they are bound to a particular platform and that they typically take longer to implement than their web or Java ME counterparts.
Modern smartphone platforms, like e.g. Apple s iPhone[2] , or Open Handset Alliances (OHA) Android[3] , have very sophisticated native programming libraries that allow for application development similar to that of desktop computers.

1.1.2. Mobile Web Applications

Mobile web applications represent the other side of the mobile application development spectrum. They offer very little to no access to device data, but have the benefit of being operating system agnostic and are traditionally easier to develop than other types of applications.
The techniques for developing mobile web applications have matured a lot over the past few years and have today almost completely converged with desktop web development standards[9]. Only some years ago, back in 1998, the Wireless Application Protocol (WAP) 1.0 was introduced which was built upon a proprietary network protocol stack and offered a very restricted markup language in form of the Wireless Markup Language (WML). WAP 2.x introduced in 2002 showed first tendencies towards an unification of mobile and desktop web development. Finally, todays smartphone devices with their modern mobile browsers like e.g. the WebKit[4] based Apple iPhone

1A feature phone is an average cell phone with additional feature support like amongst others high resolution display, camera, or mp3-player[4]
2http://developer.apple.com/iphone/, last viewed 2008-07-14
3http://code.google.com/android/, last viewed 2008-07-14
4http://webkit.org/, last viewed 2008-07-14

Safari, mark the complete convergence with the traditional World Wide Web (WWW), Internet Protocol (IP) based network protocol stack and the application development standards maintained by the World Wide Web Consortium[5] .

1.1.3. Java ME

Java ME applications effectively lie in between native and web applications, by offering more functionalities than mobile web applications and still being easer to develop than native applications. Additionally, Java ME possesses the advantage of being operating system independent. A major drawback of the Java ME platform is that it is very fragmented with varying levels of API support amongst different devices[9].
Like native and web applications, the Java ME platform has evolved with the wireless device market, by offering new libraries specifically targeted at new handset functionalities like Bluetooth, Web Services or location APIs. The Mobile Service Architecture (MSA)[6] currently marks the latest step in the Java ME evolution.

1.2. Smartphone versus Desktop Computing

In case of todays new smartphone devices, the difficulties in application development are not so much rooted in API deficiencies, but rather stem from other mobile device inherent limitations like e.g. limited processing power, storage capacities and battery life, a completely different form factor and totally different usage scenarios compared to desktop computers[9].
This of course also impacts mobile application security and privacy. Whereas it is e.g. no big issue to prompt a user on a desktop computer several times for a password, this can rapidly become very cumbersome on a mobile device, which only features a 12-button cell phone keypad.
With mobile devices it is possible to access company data over insecure networks, which demands for augmented security and new ways of protecting data.

2. Enterprise Search Security

The primary purpose of an enterprise search infrastructure is to provide people with a highly efficient access to the vast information stored within their enterprise data sources. Access is typically performed by the means of a search interface. The connected data sources can be as various as the corporate application environment and could e.g. comprise local and network file systems, Content Management Systems (CMS), Document Management Systems (DMS), web servers, etc. Basically, the enterprise search infrastructure acts as an agent system that analyzes and accesses information on behalf of the acting client user (see figure *1*).

5http://www.w3.org/, last viewed 2008-07-14
6http://java.sun.com/javame/technology/msa/, last viewed 2008-07-14

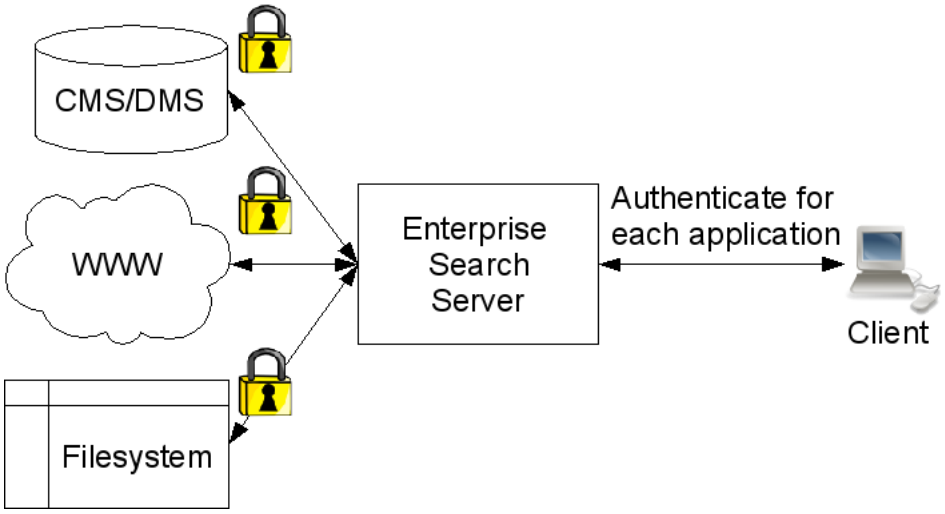

Figure 1: Enterprise search security basic architecture

Naturally, the various different connected data sources may all implement their own security mechanisms with proprietary authentication and authorization schemes. As figure *1* shows, this consequently means that the user has to provide her appropriate credentials for every application which is involved in the search result. In a real world scenario this could mean that the user has to provide different credentials for every application. Often the user may not even have a chance to know which credentials to supply, since the enterprise search application hides away the implementation details of the underlying data source and provides the search interface through a portal like federating interface.

A solution to that problem is to use a Single Sign On (SSO) architecture that authenticates the user once and allows for accessing various data sources with one login. Today most enterprise SSO environments are typically created by the use of the Kerberos[7] protocol, initially created by the Massachusetts Institute of Technology (MIT) in the 1980s. Figure *2* shows a high level view of a SSO based enterprise search architecture.

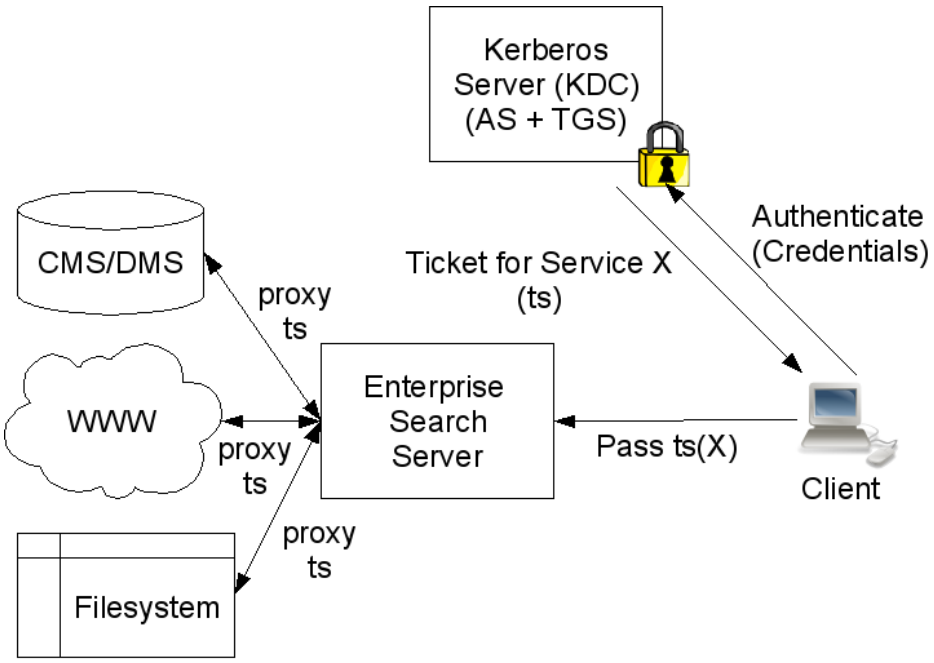

Figure 2: Enterprise search security Single Sign On (SSO) architecture

7http://web.mit.edu/Kerberos/, last viewed 2008-07-14

The basic Kerberos authentication process is as follows[11, 3]:
The user first types her credentials as user name and password into her client login window. Only the user name is transferred to the Kerberos Key Distribution Center (KDC). The private password is a shared secret known only to the KDC and the user and is never transferred over the network, but rather used as a private key for encrypting message digests . The KDC consists of Authentication Service (AS) and Ticket Granting Service (TGS). The AS authenticates the user and gives her a Ticket Granting Ticket (TGT), which gets stored in the cache of the client machine and usually is valid for some hours. The TGT serves as an authentication token for getting access tickets from the TGS to the secure applications within the enterprise search environment. The TGS returns a new service ticket (ts) for every secure application the client wishes to access. With the given ticket from the TGS, the client can finally authenticate itself at the secured application and can establish a connection
Within the enterprise search infrastructure, the search query service acts as an agent satisfying the requesting user s information need. This so called user impersonation is performed by delegating (proxying) the received tickets. It does not itself perform any authorization and is also not aware of the contents of the received tickets

While Kerberos has many advantages like SSO, strong security, wide adoption, etc., it also suffers from some drawbacks that make it particularly hard to implement in a distributed mobile scenario.
Kerberos is usually operated within secure corporate walls behind a firewall. To enable access to a Kerberos KDC from the Internet, the corporate firewall has to be configured to open ports for Kerberos that otherwise do not need to be accessible.
Furthermore under some configurations, Kerberos may not work when the clients use Network Address Translation (NAT) or dynamic IP addresses via Dynamic Host Configuration Protocol (DHCP).
The long validity of the Ticket Granting Ticket (TGT) that spares the user from repeatedly retyping its credentials could also impose a security threat to mobile devices. If the mobile device gets lost or stolen shortly after the TGT is issued, the user stays authenticated for several hours until the ticket expires, which leaves great potential for damage to the enterprise data.
Another serious drawback for mobile devices is that only very few platforms support Kerberos out of the box.

3. Mobile Web Single Sign On (SSO) Infrastructure

As figure *3* illustrates, besides the limited capabilities of mobile devices, the most significant difference regarding security compared to desktop computers operated within the premises of the company, is that they are typically operated from outside and are not physically connected to the corporate networks. Direct access to the company s network is provided only by the means of Virtual Private Networks (VPNs).

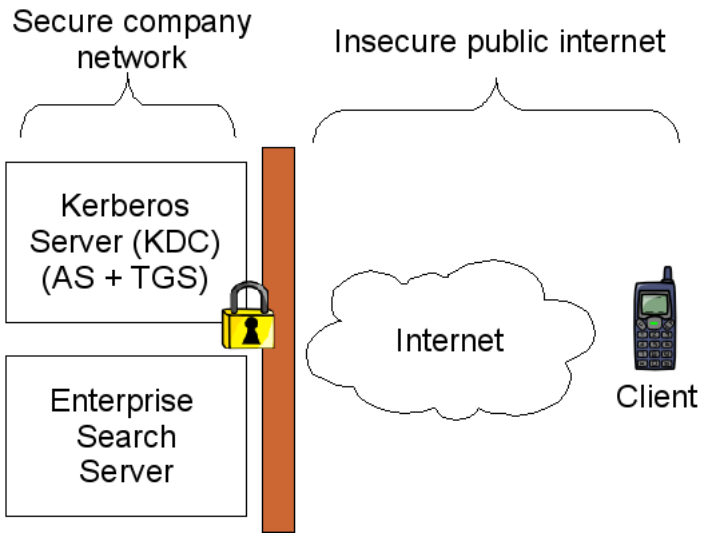

Figure 3: Access to enterprise search services from outside the secure corporate network

An interesting alternative to accessing the whole corporate network infrastructure through a VPN is to create an architecture that allows a secure access to the enterprise search infrastructure over the insecure Internet. In this sense making the application s native network stack security aware, enables possibilities for the security infrastructure to provide client specific forms of user authentication and authorization.

The following claims should by all means be supported by such an architecture:

- The support of mobile wireless devices must not compromise the existing security architecture.
- The infrastructure must answer to the possible technical limitations of mobile devices.
- The authentication mechanism must correspond to the handling of mobile devices, which is very different to that of desktop computers.
- As little as possible sensitive data should be stored on the mobile device to avoid data corruption in case of theft and loss. If nonetheless the device gets stolen/lost, their should still be a last security barrier that keeps an unauthorized user from instantly accessing the enterprise data.

3.1. General Possibilities of Securely Authenticating a Mobile Device over the Internet

The basic technologies driving the WWW like IP, Transmission Control Protocol (TCP), Hyper Text Transfer Protocol (HTTP), eXtensible Markup Language (XML), Transport Layer Security (TLS), etc. can almost be considered *ubiquitous* and are hence a good starting point for a wireless client implementation. Many platforms, like native applications, Java ME and even modern mobile AJAX based web applications have support for these technologies[9].

3.1.1. TLS server side certificates and HTTP Basic Authentication

The simplest approach of creating a secure mobile client would be to use basic authentication over encrypted secure HTTP (HTTPS). It has the advantage of being available on a great number of devices and platforms. Even simple WAP 1.x/2.x mobile browsers are capable of supporting this technology, and by designing the client search interface as simple WML or XHTML (MP) website, one would reach a great number of devices.
However, this approach has some serious drawbacks that outweigh the benefit of a large client base. The most significant disadvantage is that the user would have to provide separate credentials for every data source connected to the enterprise search server (see figure *1*), which renders this approach inappropriate in practice.

3.1.2. TLS client side certificates

An extended version of the TLS approach is to use client side certificates in lieu of HTTP basic authentication. This bears the advantage of strong two factor authentication. Another benefit is that the user would never have to type in any passwords, since the client certificate would unambiguously identify her.
Still, two disadvantages strongly militate against this architecture. Firstly, the installation of the client certificate together with the private key in form of e.g. a PKCS#12 file, would mean a high security risk if the device was lost or stolen. An additional safeguarding mechanism like e.g. remote wipe would have to be employed, to prevent unhindered access to the enterprise data in case of device corruption. The second, in terms of finding a workaround implementation even more severe disadvantage is, that client certificates can not be securely delegated[10]. Since in enterprise search queries are performed on behalf of the user, she has to be impersonated for gathering information about access rights to respective data sources. Impersonation based on client certificates would mean that the private key together with the public client certificate would have to be handed over to the server. This is practically unacceptable as it would open a dangerous loophole for malicious servers, which would then have the entire client identity.

3.1.3. Application level authentication

In the recent years, many standards and specifications have evolved in the field of Web Services that allow for securing HTTP based applications at the *application level*. Application level means that the security information is not a separate layer inaccessible from the layer above, but is directly encoded into the transferred meta data. Web Services-Security (WS-Security)[8] , Security Assertion Markup Language (SAML)[9] , XML Signature[10] and XML Encryption[11] are amongst the most

8http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss, last viewed 2008-07-18
9http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security, last viewed 2008-07-18
10http://www.w3.org/TR/xmldsig-core/, last viewed 2008-07-18
11http://www.w3.org/TR/xmlenc-core/, last viewed 2008-07-18

important specifications for secure Web Services.

WS-Security markup can be included into the SOAP-header and defines three general-purpose mechanisms for associating security tokens with message content, which are[8]:

- User name Token Profile
- X.509 Certificate Token Profile
- SAML (Security Assertion Markup Language) Token Profile

This means that WS-Security is the messaging language, whereas SAML is the security language[1]. WS-Security is very closely related to SOAP based Web Services as it is directly defined in the SOAP header. This is opposed to SAML, which provides many different bindings like e.g. a SOAP binding, HTTP POST binding, HTTP-Redirect binding, URI binding, etc. [6] and is hence independent of the underlying carrier technology. This allows it to be used in non SOAP based distributed scenarios, which are especially attractive for mobile devices, since they are technologically less demanding. E.g. the Internet search company Google proposes a web-based Single Sign-On (SSO) service for their Google Apps[2] on basis of SAML and its HTTP-Redirect binding.
The following shows an overview of a proposed architecture for securely authenticating mobile clients over the Internet by using SAML, which conforms to the claims stated in the previous section.

3.2. Proposed Web Single Sign On (SSO) Enterprise Search Infrastructure for Mobile Devices

Figure *4* shows a high level view of a web based SSO enterprise search infrastructure for mobile devices on the basis of SAML, proposed by the authors of this paper.

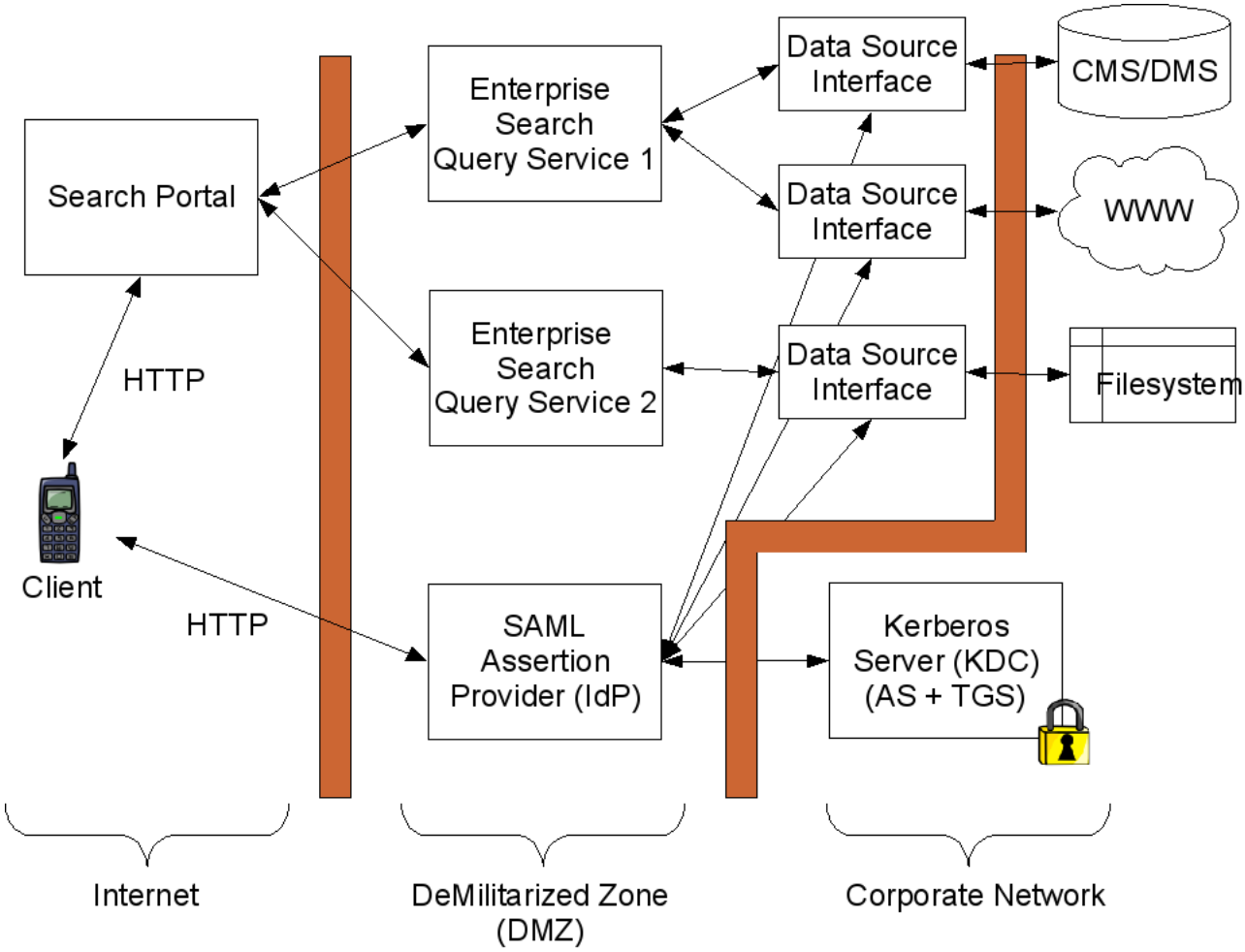

Figure 4: Proposed mobile web Single Sign On (SSO) enterprise search infrastructure

The following is a brief overview of the proposed architecture:

- There exists one single web *Search Portal* that federates *n* different enterprise search Query Services. Figure *4* shows two, *Query Service 1* and *Query Service 2*.
- The mobile user directly accesses this single portal over HTTP. The actual mobile client could be a native, Java ME or even a mobile AJAX web application.
- The next key entity is the *SAML Assertion Provider* or *Identity Provider (IdP)*. It acts as a meta authentication framework and knows, depending on the actual enterprise application infrastructure, how a SAML *assertion* has to look like. A SAML assertion is a package of information that supplies statements made by a SAML authority like e.g. an Identity Provider[7]. Most importantly the SAML assertion holds the *authentication* statement, which specifies that a subject was authenticated by *particular means* at a particular time. The means of authentication describe *how* the subject was authenticated and is described in the authentication context. SAML 2 supports many authentication context classes, like e.g. Kerberos, Password, Public Key X.509, Smartcard, etc.[5]. Besides the authentication statement, the SAML 2 specification defines the *attribute* and *authorization decision* statements.
- Every Query Service is connected to *n* different data sources, like e.g. a file system or a

CMS/DMS, via *Data Source Interfaces*. Such a Data Source Interface has access to the SAML Assertion Provider and can hence check the assertions forwarded by the Query Services on behalf of the accessing subject for validity. In order for a new Data Source to be integrated into the enterprise search environment, a new SAML aware Data Source Interface has to be provided.

- Finally, Kerberos acts as usual by maintaining the user access base and issuing the tickets necessary for authentication and authorization.

A typical use case within the above described infrastructure could be as follows:
The Search Portal completely delegates authentication to the SAML Assertion Provider. This can be realized by redirecting the HTTP user agent s location to the URL of the SAML Assertion Provider. This ensures that the Search Portal obtains user credentials in the form of a SAML delegable assertion. The credentials together with the search request is passed on to selected Query Services. Like the Portal the individual Query Service does not directly perform data source specific authorization of the credentials and simply forwards it to all its connected Data Source Interfaces. Since the Data Source Interfaces are connected to the SAML Assertion Provider, they can authorize the given assertion with the included user authentication information against the Assertion Provider and take specific authorization actions depending on their Data Source s access policies.

The result of the proposed infrastructure is a delegable and well defined security architecture that solely builds on open Web standards and allows for connecting mobile clients from within insecure networks, especially the Internet. A key aspect of this architecture is that it separates the security- from the search-infrastructure.

3.3. SSO Requirements with Respect to a Mobile Environment

Basically, the proposed infrastructure should be realizable on a large number of smartphone platforms, including native, Java ME or even (mobile AJAX) web applications.
As an additional security enforcement, some kind of *session token* could be sent to the user on authentication. This token could e.g. be a 3 or 4 digit number that the user can easily remember. For every search request, the user has to supply this code for being able to issue the request. In case of theft or loss of the device, this prevents a third party from instant access to sensitive enterprise data.
Another consideration specific to mobile devices could be to reduce the validity of the SAML assertion to a rather short time. Instead of being valid for several hours like e.g. a Kerberos ticket, only several minutes up to an hour could be a reasonable time range for the SAML assertion of a mobile device. This would increase the likelihood of the device being locked in case of theft or loss. As with the proposed session token, this can only be understood as a preliminary precaution until further security enforcements take place.

4. Conclusion

Mobile device technologies evolve at a fast pace, opening new fields of application to distributed mobile infrastructures that have not been able to be realized still some years ago. While hardware features rapidly mature and mobile software application paradigms seamlessly converge with those from desktop computing, mobile devices still possess their unique characteristics that influence application development. Their handling is very different from that of desktop computers and they are operated in quite different scenarios compared to stationary devices.

This paper focuses on the adoption of mobile devices within an enterprise search infrastructure. While new compelling smartphone devices like e.g. Apple s iPhone will most likely be key enablers of a shift in enterprise computing towards the usage of mobile clients, this paper outlined that their incorporation into existing desktop centric architectures is challenging and can not be easily done due to their distinct properties. Mobile devices are usually operated outside the secure corporate walls. Employees working with mobile clients connected to the corporate infrastructure through untrusted networks requires the enterprise infrastructure to adapt.

An approach for leveraging an existing enterprise search infrastructure based on Kerberos authentication is proposed within this paper. Security is provided on the application level by means of the Security Assertion Markup Language (SAML) over HTTP. This combination allows it to be deployed on a large number of mobile client platforms. The proposed infrastructure represents a secure, federated architecture which offers the great benefit of Single Sign On and a clear separation of the security- from the application-infrastructure.

Further refinements of the proposed architecture should involve measures of pro actively preventing third parties from unwarranted access of enterprise data in case of loss and theft of the mobile device.

References

[1] Bromberg, Peter A.: *WS-Security, SAML, Standards and the Future of Webservices*. 2002. URL, http://www.eggheadcafe.com/articles/20020706.asp, last viewed 2008-07-18.

[2] Google, Inc.: *SAML Single Sign-On (SSO) Service for Google Apps*. URL, http://code.google.com/apis/apps/sso/saml_reference_implementation.html, last viewed 2008-07-18.

[3] Hewlett-Packard Development Company, L.P.: *Kerberos White Paper*, MAR 2005. URL, http://docs.hp.com/en/6332/Kerberoswhitepaper.pdf, last viewed 2008-07-18.

[4] Jakl, Andreas: *Symbian OS Overview*, 2007.

[5] OASIS Open Organization: *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*, MAR 2005. URL, http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf, last viewed 2008-07-18.

[6] OASIS Open Organization: *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, MAR 2005. URL, http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf, last viewed 2008-07-18.

[7] OASIS Open Organization: *SAML V2.0 Executive Overview*, APR 2005. URL, http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf, last viewed 2008-07-18.

[8] Oracle Corporation: *WS-Security Authentication*. URL, http://www.oracle.com/technology/tech/java/newsletter/articles/wsaudit/ws_audit.html, last viewed 2008-07-18.

[9] Praher, Christian: *Mobile Service Oriented Architecture in the Context of Information Retrieval*, JUN 2008.

[10] Santos, Nicholas Sean W. Smith: *Limited Delegation for Client-Side SSL*, 2007.

[11] Walla, Mark: *Kerberos Explained*. MAY 2000. URL, http://technet.microsoft.com/en-us/library/bb742516.aspx, last viewed 2008-07-18.

[12] Zheng, Pei Lionel M. Ni: *Smart Phone & next Generation Mobile Computing*. Morgan Kaufmann, 2006.