Secure Communication Framework for Mobile Devices

Byung S. Yang, Soren Dreijer, Sugih Jamin, Sarit Mukherjee, and Limin Wang

byungsuk@umich.edu, halloko@hotmail.com, jamin@eecs.umich.edu,
sarit@alcatel-lucent.com, liminwang@research.bell-labs.com
Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2122, USA

Abstract. The WebBee system is a complete software framework that supports security-sensitive applications for mobile, handheld devices. Though the WebBee system is designed to support security for users, there is a critical window of time between when a device is compromised, and when security is once again established. Security is reestablished by one of two mechanisms. The first is enacted when the user is able to notify the system of the compromise. Secondly, security will also be reestablished when the security keys are renewed. In the paper, we introduce a challenge-response system which is tailored for the WebBee environment to support its unique characteristics. The challenge system has been designed and implemented on top of the WebBee system so that the system as a whole can maximize security even in an emergency. The design scheme has been carefully chosen to minimize the impact on the existing WebBee infrastructure and to integrate with other detection mechanisms easily. The overall performance of the WebBee system is not affected by the challenge system significantly because challenges are only assigned in limited numbers.

1 Introduction

As we continue to move toward a world that is becoming increasingly more mobile, it is more crucial than ever to evolve established security mechanisms in order to guarantee high security for the future. Simple password-based authentication can be easily compromised once a user password is stolen or correctly guessed, resulting in the need for more secure authentication schemes. One such proposed initiative is the challenge-response system. Internet banking services are examples of organizations that have adapted challenge-response mechanisms to resist confidential theft attacks. Hiltgen et. al. proposed a challenge/response-based, short-time password authentication method, as well as a two-stage, PKI-based authentication method for Internet banking [1]. We believe this kind of challenge-response system also has great potential in mobile environments - where phones are lost or stolen - and confidential information, such as user names and passwords, could be leaked and exploited by malicious third-parties. Unlike "Secure Internet Banking Authentication" [1], which is focused on session-based authentication, mobile environments have the added variable that frequency of authentication must be taken into account, since mobile devices are more susceptible to theft due to their mobility. Therefore, we propose a challenge-response system that challenges a user based on dynamic properties such as his or her current geographical location, the types of applications the user has instantiated, and database records of the user's previous actions. The system adapts to the behavioral patterns of the user, and access to the server's applications is adjusted according to the failure rate of solving the challenges.

The system is integrated into our current project - WebBee [4] [10], a secure coordination and communication infrastructure suitable for a team of first responders equipped with commercial off-the-shelf handheld devices. WebBee is designed to function through a mesh network even in circumstances where disasters have compromised public network service. WebBee clients can choose to communicate over the mesh network infrastructure even if the public network service is present, so as to deliver better latency and security. Through this network, clients are provided various services, from general Web scraping to mission-critical reports.

2 Background

To protect the WebBee system, two complementary methods have been employed: prevention and detection [6] [7]. Prevention techniques are used to establish secure channels between the server and clients through

authentication and encryption so that only legitimate users can have access to the system. WebBee has implemented both authentication and encryption: forward secure signatures and server-assisted signatures are integrated for upload security, and Quorum-based download security is provided with reduced rekeying overhead for scalability [5] [10]. These security protocols are developed based on an SSL scheme which is identified as one of the methods to prevent online-channel breaking attack [1]. The challenge-response system also helps thwart channel-breaking attacks by providing nondeterministic short-time passwords; clients are challenged based on dynamic properties such as behavior patterns and service usages.

In addition to the secure authentication and encryption mechanisms in WebBee, upload security in WebBee is designed for users to be able to block their devices by submitting them to a blacklist when their devices are compromised [10]. However, there is a critical window of time between when a device is stolen, and when either the user submits his device to the blacklist, or when the device needs to be rekeyed. For some applications which require high security, it may be important to reduce this time window. Therefore, detection mechanisms are necessary to enhance the existing security mechanisms, and several systems are proposed [8] [9]. Such detection systems identify anomalous behavior in clients by analyzing their histories, which are stored in a database. That means the detection systems assume that a database is present, and that it stores client behaviors so as to identify what behavior is normal. However, this kind of assumption may not be applied to the WebBee system, in which the whole system infrastructure needs to be able to be deployed in a short period of time in a disaster area, so as to facilitate ad-hoc team coordination [5] [10]. This unique characteristic of the WebBee system will cause high false-positive detection rates, and even makes the detection system malfunction at the beginning state due to the initial lack of a database. Therefore, in addition to user history-based detection mechanisms, the challenge system integrates non history-based detection mechanisms such as time based and application access based authentication. These continuous authentication methods feature more prominently, at first, than the history-based detection mechanisms-at least until enough of the user's history has been established. The challenge system is developed on top of WebBee so that, in addition to the original WebBee security prevention mechanisms, it provides the base structure and deterrent mechanism for prevention and detection methods. As described before, due to the unique characteristics of the WebBee system, the detection may cause high false-positive rates, which results in some important clients or data packets being dropped. Instead of punishing clients who are identified suspects, clients are challenged to render their authentication. Various detection mechanisms can be applied such as deterministic detection (e.g., time-based detection and application-based detection), anomaly-based detection (e.g., GPS-based detection), and misuse-based detection (e.g., known attack signature).

3 Threat Model and Assumption

3.1 Threat Model

Since part of WebBee's mission is to provide security-sensitive services, one of the main concerns for implementing it is security enhancement. Two main security threats have been identified as the primary vulnerabilities for systems like WebBee: 1) online channel-breaking attack (man-in-the-middle attack) and 2) the theft of handheld devices. In the online channel-breaking attack scenario, malicious users may be able to intercept messages between the WebBee server and clients without the knowledge of the legitimate participants. As a result, the intruders could observe authenticated information from the WebBee server, or provide incorrect data to the server to harm the system. Another security issue for WebBee is the inherent physical vulnerability of handheld devices: due to their size and mobility, handheld devices can be stolen easily and, even worse, the theft may be reported much later, resulting in the possibility of crucial information being exploited by malicious users.

3.2 Assumption

We have made several assumptions, described below, in the design of the challenge system for WebBee.

First, we assume that the accumulated data for mobility and usage patterns of a user accurately reflects the normal behavior of the user. Based on a normal behavior data profile, the detection system can classify

the current behavior of a client as normal or abnormal. However, in emergencies, clients may exhibit irregular behaviors-i.e., behaviors that they don't exhibit in non-emergency situations-such as greater usage of certain specific applications, more frequent calls and changes in movement patterns. Therefore, it is necessary to differentiate anomalies caused by emergencies from anomalies generated by compromised devices. In case of an emergency, the challenge system may need to depend more on deterministic challenges (time-based and application type based) rather than on detection methods based on dynamic properties (GPS-based challenge). Such decisions can be tailored adaptively according to the current false-positive rate. The challenge system can adaptively enact the tailoring of such dependencies so that, at the beginning of infrastructure establishment or in emergency, the system may gracefully transition to a dependence on deterministic challenges.

Second, we assume that handhelds are not equipped with tamper-resistant hardware. If a handheld device is compromised, WebBee services are vulnerable to attack or appropriation by malicious users, unless the users are challenged. Therefore, the challenge system needs to minimize the interval between when a device is compromised and when a WebBee service is used by malicious users. Also, it is necessary to be able to define different intervals for the various services, based on a service's individual security level.

Third, we assume that a malicious user cannot duplicate the behavior of the original user. The detection systems identify suspicious clients by comparing the behaviors of clients to their history data. Therefore, if the malicious users can manipulate their behavior patterns to duplicate the original users', the detection systems will not be able to function correctly.

Fourth, we assume that the whole WebBee system (including the challenge system and WebBee applications) is secure, and the only possible point where malicious users can attack the system is the communication point between the WebBee system and clients. The security of the WebBee system itself can be guaranteed by having the whole system installed on a secure machine, or by having several machines with secure channels. We also believe that installed WebBee applications are credible and do not jeopardize the whole system.

4 Challenge System Overview

The challenge system is built as a standalone component that can be integrated into any framework that requires a challenge-response mechanism. The types of challenges are fully customizable, and the challenge protocol can carry any type and size of data. Below, we discuss the challenge system in relation to WebBee. Often, a user must authenticate with a server before he or she can request access to specific services. When authenticated, the user can use these services based on the access privileges granted by the server. Typically the connection and the trust between the server and user are based on pre-established cryptographic keys and information such as user names and passwords. In a mobile environment, this information is stored on the user's personal device, such as a smart phone, and if the device is stolen, a malicious third-party can exploit the stored information to impersonate the real user. An anomaly-based *Intrusion Detection System* (IDS) that monitors the user's access patterns decides whether to issue a challenge to ensure that it is in fact the genuine user who is accessing the server's services. If the IDS determines that the user should be challenged, the original client request is held back until the challenge has been successfully solved. Clearly, there is a trade-off between how often a user should be challenged to provide the desired level of security and the tolerance of the users (too many challenges hinder efficiency and are an annoyance).

It is imperative that a malicious user not be able to explore the challenge space or wait for a specific challenge to come up. That is, if a malicious user keeps sending different requests to the server, the user should always get the same challenge back until it has been solved. This minimizes the chances of a malicious user accessing the server's services to $\frac{1}{n}$ (where n is the number of challenges for the user) if he only knows the solution to a single challenge, and it also makes it harder to explore the challenge space to identify the challenges a specific user has created. Also, every failed response a user submits is recorded in a database, so if a user responds to challenges incorrectly multiple times, that user will be placed in the blacklist until he or she is verified again.

4.1 Adaptive Challenging Rate

Clients in WebBee have their own security levels defined *[I-Min, I-Max]*. *I-Min* represents the minimum interval, which can be either time or the number of accesses, that each client needs to be challenged. At each of these intervals, information which is used for detection is sent to the challenge server, and a challenge decision is made based on the detection system's results. The cost of communication between the challenge server and clients prevents the challenge server from receiving this kind of data too frequently. *I-Min* also can be interpreted as the maximum interval that a WebBee service can be exploited by malicious agents by using stolen devices. *I-Max* represents the maximum amount of time a client can go without being challenged - which comes into effect if all detections pass successfully. *I-Min* cannot exceed *I-Max*. The size of *I-Min* and *I-Max* decides the security level of a client. If a client requires higher security, he or she needs to set those intervals to smaller values, while lower security may be set with larger *I-Min* and *I-Max* values.

The challenging rate of a client represents the rate of challenges in an interval *I-Max* and defined as the following:

$$\textit{C-rate} = \#\text{ of challenges assigned in } \textit{I-Max} \ * \frac{\textit{I-Min}}{\textit{I-Max}}$$

C-rate increases to 1 either when the detection systems have not accumulated enough history for a client in the database, or when it is an emergency. If a client has a high *C-rate*, the challenge system adaptively adjusts to depend to a greater extent on the deterministic challenges (and to a lesser extent on the non-deterministic challenges) by increasing *I-Min*. When the *C-rate* of the client decreases, the value of *I-Min* decreases down to the original value.

5 Challenge System Architecture

5.1 Overall design

We have integrated the challenge system into the existing WebBee framework. The challenge system with anomaly-based intrusion detection systems and other detection systems function as one entity, which operates separately from the rest of WebBee. The challenge system protocol is illustrated in Figure 1 and works as follows: (1) When a client sends a request to a server application, (2) the WebBee server sends some information about the client, which is used for a challenge decision, to the Challenge server. WebBee applications cooperate with the WebBee server to extract the proper information. (3) Using the client information and based on the current challenge policies, the Challenge server determines whether the client should be challenged, and if so, creates a challenge from the information it has on the user and (4) sends it back to the WebBee server, which in turn (5) stores the original client request in a database and (6) forwards the challenge to the client. The client request is only stored for a limited amount of time to avoid a malicious user depleting the server's resources, and the client must solve the challenge within a narrow time-window that is adjusted according to the environment in which the challenge system is deployed. (7) Once the client sends back a solution to the WebBee server, (8) it is forwarded to the challenge system, which verifies it. (9) The WebBee server is then notified whether the solution was correct, and if it was, (10) the original client request is obtained from the database and (11) is forwarded to the intended application server. After the request is processed by an application server, (12) the result is sent back to the WebBee server and (13) forwarded to the client.

Challenges are persistent across connections to avoid problems in high-latency environments where a client could experience difficulties in solving a particular challenge because the connection (and thus the challenge and the initial server request) was dropped before a response could be sent back. Therefore, we allow challenges to be solved within a limited time window after which the challenge and the client request are invalidated and removed.

Existing client applications need to be modified in order to support the challenge system. However, the challenge itself is decoupled from specific applications, which can choose to interpret and display the data as they see fit. The WebBee protocol, on which the challenge protocol is built, is deterministic in that it does

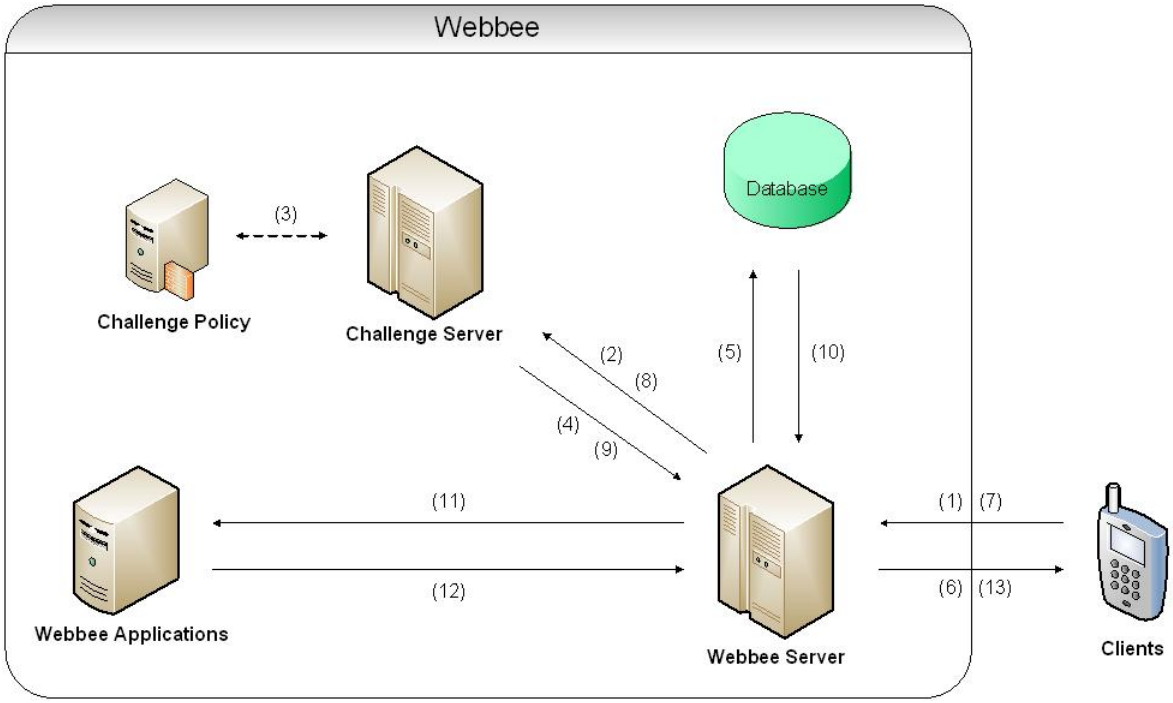


Fig. 1. An overview of the challenge system integrated into WebBee. The arrows represent the sequence of communications involved in the challenge process.

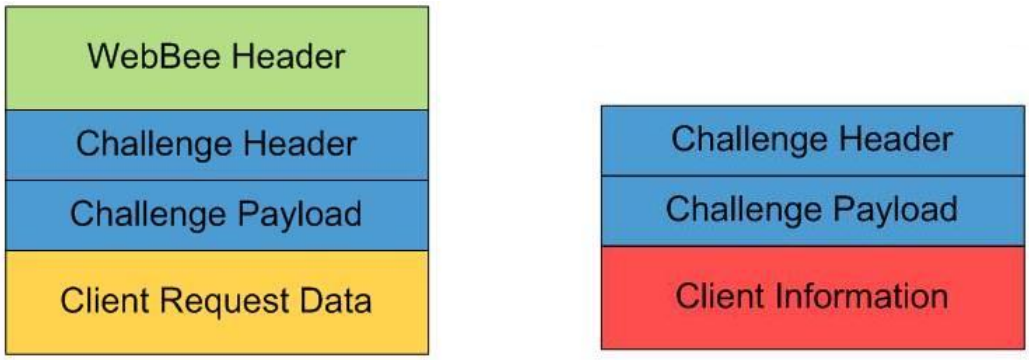
not support sporadic messages sent from the server to the client. That is, when a client sends a request to the server it expects a reply and the server cannot contact the client directly without the latter first initiating the conversation. However, since the challenge data is piggybacked on server replies, the client, which is waiting for a reply to an earlier request, can solve the challenge and then go back to waiting. The challenge system only needs to challenge users when they are accessing services on the server, so this somewhat limited communication design does not impact the effectiveness of the challenge system.

5.2 Challenge Protocol

In order to allow the challenge system to communicate with the client, we augmented the original WebBee communications protocol to piggyback the challenge data, which consists of two parts: a fixed-sized header and a variable-sized payload as shown in Figure 2(a). The header is always present in packets sent between the server and a client, and includes information such as the identifier of the challenge, the type of message, and the length of the payload. It is this header that is passed between the WebBee server and the Challenge server, and thus the Challenge system does not see actual client requests. However, some client information needs to be used to make a challenge decision. Such information is extracted with help of WebBee applications which understand structures of client request data, and is included in the challenge payload in Figure 2(b).

5.3 Client Information

Client request data includes information which is used to make a challenge decision for clients (for example, GPS information is required for the GPS-based challenge policy, and request time is required for the time-based challenge policy). Hence, one simple implementation is to have a client request data by sending it to the Challenge server, and make the Challenge server parse the necessary information from data received. However, there are two main issues: 1) Depending on the WebBee applications with which the client is



(a) Protocol between clients and WebBee server (b) Protocol between WebBee server and Challenge server

Fig. 2. Communication Protocols within the WebBee system

trying to transact, the size of the request data may be large, resulting in significant data transfer overhead between the WebBee server and the Challenge server. Some requests contain images or audio data which may exceed several hundred kilobytes. 2) To extract the necessary information for challenge decisions from the request data, the Challenge server should understand the structure of the request data which varies depending on the application. Also, in case a new application is added to the WebBee system, it may be necessary to modify the Challenge system code, which violates the decoupling of the Challenge system from the other subsystems. In the current implementation as shown in Figure 3, when the Webbee server receives request data from clients, it requests structural information about the request data from the application servers based on application identities, and with such structure information, the Challenge server selects only the necessary data, depending on the current challenge policies applied. In this way, data transfer to the Challenge server is limited to only data used for challenge decisions. Furthermore, the Challenge server and the WebBee server do not need to understand the request data structure and the current challenge policies, resulting in better decoupling.

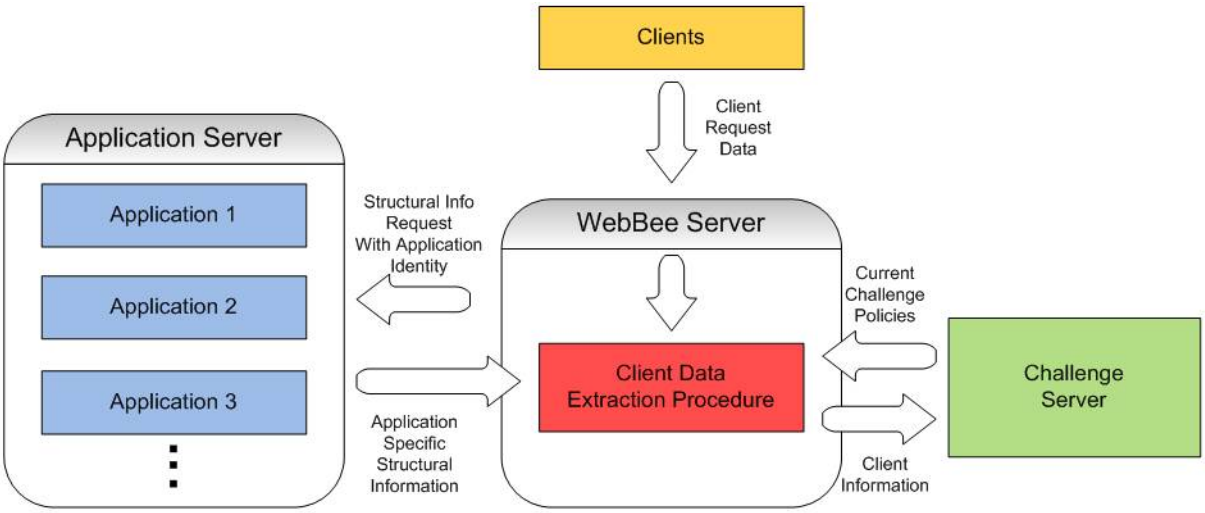


Fig. 3. Client information for challenge decisions is extracted at the WebBee server with structural information provided by WebBee applications based on current challenge policies and sent to Challenge Server

5.4 Challenge Policies

Several metrics can be used by the Challenge server to make the decision of whether or not to challenge a client. Simple metrics include periodic challenges, which require a client to authenticate every so often, and low credibility challenges, which involve challenging clients that have a significant history of unsuccessfully solved challenges. More advanced metrics use other information, such as behavior of clients, current positions, and application types. As shown in Figure 4, the Challenge Policy Proxy cooperates with the Challenge server to analyze various pieces of interest in the client information, which is provided by the Challenge server, and makes a challenge decision based on the current challenge policies instituted, and on the security setup of an individual client. In the current system, the application type-based challenge policy and the GPS-based challenge policy are implemented on top of the periodic challenge policy.

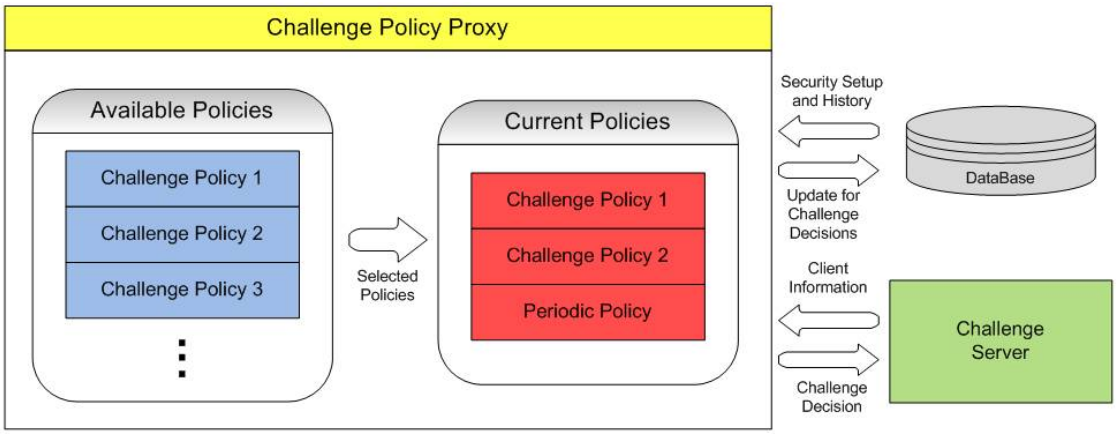


Fig. 4. Challenge Policy Proxy cooperates with Challenge Server to make a challenge decision

Application type-based Challenge Policy The application type-based challenge policy challenges based on what application(s) a client is running, since some applications do not warrant a high level of security. The list of high-security applications is marked in the database so that when a client requests a service for high-security applications, the client is challenged. Since some clients may want to have their own lists of applications to be challenged, the system looks up the security setups of clients in database, or it uses default lists.

GPS-based Challenge Policy The GPS-based challenge policy decides whether to challenge clients based on geographical information when clients request secure services. When a challenge is solved by a client correctly, the current GPS coordinates of the client are recognized as VALID and saved in database as part of the VALID GPS history for the client. Each VALID GPS entry represents a geo-positioned circle with a client-defined radius, and the combination of a set of these circles forms a zone within which client is not challenged, until he or she leaves the zone. VALID GPS entries are designed to expire after a certain amount of time, and such entries are no longer VALID and are excluded from the database. Therefore, if a client who moves to another area returns to his or her original area before the expiration, the client is not challenged. Just as with the application type-based policy, each client may set his or her own radius and expiration according to the security level he or she wants to establish.

6 Evaluation

The challenge data is included in the WebBee communication protocol, and it varies in size depending on the current challenge policies and the WebBee applications used. The current implementation has two distinct challenge policies applied: a GPS-based policy and an application type-based policy. For each client request, a fixed-size challenge header and additional challenge information incur less than 40 bytes overhead on packets from the WebBee server to the Challenge server, and also there is structural information of 30 bytes transferred from the Application server to the WebBee server. This overhead is likely to increase as more policies are applied. If a challenge is assigned to a client, variable-sized challenge and challenge response payloads are present depending on challenge types, but it does not impose extra overhead during normal communication. The WebBee and challenge system are integrated in two machines: Star and Samba. The Samba server is a database server which is connected to Star with wired LAN (100Mbps Ethernet), and the Star server has the rest of components of the WebBee and challenge systems. Both machines consist of AMD Atholon 64 X2 Dual Core 3800+ CPU and Western Distal 40GB 5400RPM hard disk, but Samba has 2G memory while Star has 1G.

6.1 Basic Performance Evaluation

Round-trip times from when a new client issues a request to when the client receives a response from the WebBee server have been measured so as to evaluate the current system, as shown in Table 1. The first column represents the average round-trip time taken when only the WebBee server is used to process client requests. Other results are compared to the WebBee Server-Only time. The second column shows the average time taken when the Challenge server is present but no challenge policy is applied. This result shows that there is about 8.5 ms difference in challenge data communication overhead between the WebBee server and the Challenge server. The third column is the average round-trip time when a client is challenged and responds to the challenge. Since the result includes a second round of communication between a client and the server to answer a challenge, the actual overhead of the Challenge server in this case is 88 ms and 59% of total roundtrip time. Current challenges are simple text-based challenges which are typically quite short in length, and so are suitable for challenging users on high-latency connections. Though the overhead for the Challenge server when a challenge is assigned seems significant, it does not affect the overall performance of the WebBee system due to the fact that clients are rarely challenged. The last column represents the average time taken when both GPS-based and application-based challenge policies are applied. 56 ms of additional overhead is rooted mostly from structure data transfer between the Application server and the Challenge server, and from the database access for the GPS history of a client. In the current implementation, GPS-based and application type-based challenge policies are applied on top of the periodic policy, meaning that once a client answers a challenge correctly, neither policy will be applied to the client for a certain length of time. Therefore, most round-trip times for when a client accesses the system are 39.5 ms.

Table 1. Performance of the Challenge System

	Webbee Server Only	Challenge System on but no challenge assigned	Challenge System without any policy	Challenge System with policies
Average time (ms)	30.920	39.502	150.302	206.145
Standard Deviation (ms)	0.986	2.610	11.446	10.957
Overhead (ms)		8.581	88.462	144.305
Fraction (%)		21.724	58.856	70.002

6.2 The Number of Challenges

We have evaluated the relationship between the number of challenges and the effect on system performance. The larger the number of challenges for each client is, the less likely a malicious user can use the client's system when an answer is known. However, the large number of challenges decreases the system performance as in Figure 5.

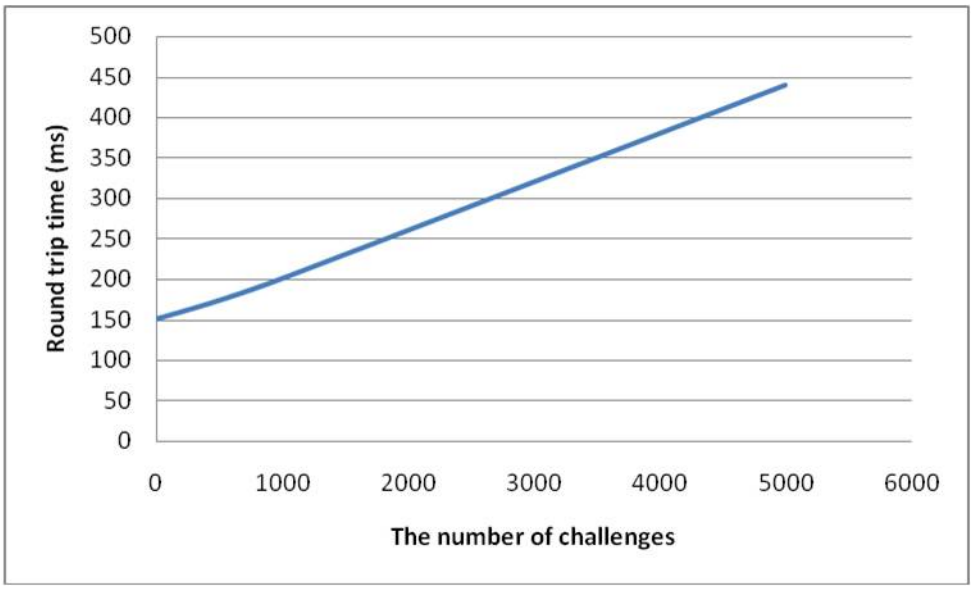


Fig. 5. The round trip time increases as the number of challenges increases.

7 Discussion and Future Work

7.1 Security

There are a couple of security issues in the Challenge system. First, when malicious users try to use a stolen device, they may not be challenged until a certain amount of time (*I-Min*) has elapsed. Users are able to manage their own settings about how frequently they are challenged so that challenges can be assigned every time users access client applications that require a high security level. However, it may annoy users, especially if responding to a challenge is a very complex procedure. One possible solution may be to have a challenge policy that recognizes changes in users' behaviors immediately. For example, a constant voice recognition or a fingerprint reading system will be able to detect the change of a user at any moment. However, this will require constant voice or fingerprint data transferring to servers, resulting in significant system overhead.

The second issue is that malicious users may be able to send specious data to the server to avoid challenges. Code Obfuscation [2] can be used to encrypt client applications so that even if a device is stolen, a malicious user will not be able to understand and modify applications. However, some of the data in client applications come from external devices, such as GPS units and clocks; bogus devices may be maliciously substituted to generate spurious information. Some devices have unique device identifiers (for example, Bluetooth GPS devices), so client applications can be programmed to accept data from only registered devices. However, others may not have such identifiers, and it will be more difficult to distinguish these kinds of unauthorized devices from authorized ones.

7.2 Additional Policies

The Challenge server is structured in such a way that an individual challenge policy module can be developed separately, and can be applied regardless what other current challenge policies are in place. Also, challenge policies can be applied to users in various ways. Some users may have multiple policies applied with the highest security level, and some may not have any policies applied. One issue is that in the current Challenge system, challenge policies are applied only statically. In certain circumstances, dynamic policy applications may increase performance or help to make better challenge decisions. For example, if a user is on a trip and is moving very quickly or erratically, GPS-based challenges will be issued to the client every time he or she accesses the system. However, in a dynamic policy application, the user may be able to inform the server about his or her current situation, and the Challenge system would make a challenge decision based less on GPS information, but more on other information.

7.3 Challenge Response

In the current Challenge system, challenges and responses are in plaintext, which is beneficial in reducing the amount of data transfer between client applications and servers. However, text responses can be easily stolen, and there is no way to identify untrustworthy responses. The Challenge system allows users to create a large set of challenges and responses. With n challenge-response pairs, an attacker with one stolen challenge-response pair has a $\frac{1}{n}$ chance of defeating the system. Thus, increasing the number of challenge-response pairs will tighten security. On the other hand, the increase may also confuse users, because they are required to remember more. One possible solution to this is to employ a biometrics-based response system such as voice, face, and fingerprint identifications [3]. Due to the fact that it requires special, potentially very costly sensors, and that it also generates a significant amount of additional data transfer, it may be employed for only security-critical applications. Another possibility is to use Smart Cards to produce short-lived challenge responses [1], but these are susceptible to theft, just like hand-held devices.

8 Conclusion

In this paper, we proposed a challenge-response system to provide better security to our existing project - WebBee. It supports various unique characteristics of WebBee, and as a standalone component, it minimizes coupling with the WebBee server. The challenge system provides an infrastructure to integrate various static or dynamic detection systems, and different sets of challenge policies can be applied to each user by the application of user-specific security settings. Though the overhead of the challenge system is relatively large, the overall performance of the WebBee server does not decrease due to the small number of challenges issued to a client in a given interval.

References

1. Hiltgen, A., Kramp, T., Weigold, T.: Secure Internet Banking Authentication. Security & Privacy Magazine, IEEE, vol. 4, pp. 21 - 29 (2006)
2. Greg Travis.: How to Lock down Your Java Code. http://www.ibm.com/developerworks/java/library/j-obfus (2001)
3. Ratha N. K., Connell J. H., Bolle R. M.: Enhancing Security and Privacy in Biometrics-based Authentication Systems. IBM System Journal, vol. 40, pp. 614-634 (2001)
4. Krian Upatkoon, Wenjie Wang, Sugih Jamin.: WebBee: An Architecture for Web Accessibility for Mobile Devices. Proc. of the 10th IFIP International Conference on Personal Wireless Communications, Colmar, France (2005)
5. Sugih Jamin.: Secure Coordination and Communication in a Crisis Using Hand-held Devices. DHS S&T Cyber Security Project USSS Technology Day
6. Sun B., Yu F., Xiao Y., Wu K., Leung V.C.M.: Enhancing Security Using Mobility-Based Anomaly Detection in Cellular Mobile Networks. IEEE Trans. Vehicular Technology, vol. 55, pp. 1385-1396 (2006)
7. Debar H., Dacier M., Wespi A.: A Revised Taxonomy for Intrusiondetection Systems. Ann. Telecommun., vol. 55, pp. 361-378 (2000)
8. Zhang Y., Lee W.: Intrusion Detection Techniques for Mobile Wireless Networks. Wireless Networks 2003, vol. 9, pp. 545-56 (2003)
9. Sundaram A.: An Introduction to intrusion Detection. ACM Crossroads, vol. 2, pp. 3-7 (1996)
10. The WebBee Team.: Webbee: a Secure Coordination and Communication Application Infrastructure for Handheld Environments in Crisis Scenarios (2008)