INTERNATIONAL JOURNAL OF ENHANCED RESEARCH IN MANAGEMENT & COMPUTER APPLICATIONS

VOL. 1 ISSUE 3, DEC.-2012 ISSN NO: 2319-7463

Software Security Risks in Mobile E-Commerce

Yeddula Venkatramana Reddy, Member IEEE
vaiviar@ieee.org

INTRODUCTION

Most current e-commerce transactions are conducted by users in fixed locations using workstations and personal computers. Soon, we expect a significant portion of e-commerce will take place via wireless, Internet-enabled devices such as cellular phones and personal digital assistants. Wireless devices provide users mobility to research, communicate, and purchase goods and services from anywhere at anytime without being tethered to the desktop. Using the Internet from wireless devices has come to be known as mobile e-commerce, or simply "m-commerce," and encompasses many more activities than merely online purchasing. One of the major wireless applications is Web access for retrieval of real-time information such as weather reports, sport scores, flight and reservation information, navigational maps, and stock quotes. While email will continue to dominate wireless applications, innovative online applications that, for instance, use location reference information of end users will drive new areas of mobile e-business growth.

Strategy Analytics, among other market research groups, predicts that by 2004 there will be over one billion wireless device users, some 600 million wireless Internet subscribers, and a $200 billion mobile e-commerce market . The Gartner Group estimates that by that same year, 40% of consumer-to-business e-commerce will be conducted over Web-enabled phones . The average cost per minute of wireless usage is expected to drop to two cents per minute in 2004 . It is expected that by 2008, the number of wireless Internet devices will outnumber wired devices .

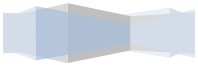
In spite of the seemingly unlimited potential to drive new applications and markets in mobile e-commerce, new security and privacy risks particular to the wireless medium and devices abound in m-commerce applications. Integrating security and privacy into online m-commerce applications will enable a projected $25 billion market in wireless software, content, and commerce. On the other hand, failing to provide a secure system of m-commerce will significantly dampen consumer adoption rates.

Even as the general form factor of mobile computing devices has been drastically reduced, the computing capacity has grown significantly. Today's handheld devices have computing power equivalent to their desktop-computing counterparts of only one generation earlier. This phenomenon, while driving more and more functionality into handheld wireless Internet-enabled devices, is also driving security risks endemic to desktop computing into wireless devices. For example, malicious code—long a problem affecting desktop computing—is likely to be a troubling scourge in m-commerce as the capabilities of handheld wireless devices increase.

New Security and Privacy Risks

In addition to contending with the usual Internet security threats in online applications, wireless devices introduce new hazards specific to their mobility and communication medium. Consider that wireless devices can form ad hoc networks where a collection of peer mobile nodes communicates with each other without assistance from a fixed infrastructure . One implication of ad hoc networks is that network decision-making is decentralized. As a result, network protocols tend to rely on cooperation among all participating nodes. An adversary can exploit this assumed trust to compromise cooperative nodes. For instance, an adversary that compromises a single node can disseminate false routing information to take down the ad hoc network, or worse, instruct all routing to go through the compromised node. Similarly, mobile users will roam through many different cells, ad hoc networks, administrative boundaries, and security domains. As the communication is handed off from one domain to the next, a single malicious or compromised domain can potentially compromise wireless devices through malicious downloads and misinformation or simple denial of service.

Rather than an attacker needing to pursue a target, targets can come to attackers in wireless networks simply by roaming through the attacker's zone. Wireless devices pass through many different, potentially non-trustworthy networks from which service is derived and data



VOL. 1 ISSUE 3, DEC.-2012 ISSN NO: 2319-7463

is exchanged. Information can be stolen or altered without the end user's knowledge. Service can be, and is often, easily denied, whether inadvertently or not. Transactions can be interrupted and then reinstated, often without re-authenticating principals. Simply "refreshing" a browser to reestablish the connection may inadvertently introduce risks. Reestablishing connections and transactions without re-authenticating principals on both sides of the transactions can be dangerous. Requests can be redirected and malicious code surreptitiously downloaded with expected Web data. Most Web sites are not currently configured to deal with intermittent service failures, as is common with wireless connections. Most vendor implementations of the Secure Sockets Layer (SSL) or its wireless counterpart (WTLS) do not re-authenticate principals or recheck certificates once a connection has been established. Attackers can use this vulnerability to their advantage in wireless networks.

Malicious hackers can compromise wireless connections even without exploiting cooperative ad hoc networks at the transport level. Consider airline passengers who check their stock portfolios in and around airports during wait times and make trades from their mobile phones. A malicious hacker can compromise the closest directory naming services (DNS) server that routes the passenger's Web request to their favorite financial online site such that all requests to, say, Quicken.com, are redirected to the malicious hacker's site. Since secure DNS has not been widely deployed, let alone in wireless networks, it would not be very difficult to implement such a stealthy man-in-the-middle attack. Providing the same look and feel of a Web site is as easy as downloading the target site's Web pages. The insidious part of the attack is in discreetly changing dynamic information—such as stock quotes—to the benefit of the malicious entity or the detriment of the end user.

The wireless medium also provides excellent cover for malicious users. Users of wireless devices can be difficult to trace because wireless devices roam in and out of wireless zones, have no fixed geographic point, and can go online and offline easily. As a result, attacks from wireless devices will likely become the preferred method of operation for launching attacks against fixed networks, especially as the capability of these devices grows.

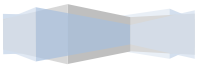
Another risk unique to mobile devices is the risk of loss or theft. Without physical perimeter security provided by buildings, locks and guards, mobile computing devices are at increased risk of theft and loss, particularly given their small size. While the data stored on a misplaced device might be irreplaceable or proprietary, other risks of lost Internet-enabled devices include the ability for finders of lost devices to access proprietary corporate systems, including email servers and file systems. One of the key problems with the current generation of handheld devices is the lack of good mechanisms to authenticate a particular user to a particular device.

In addition to the issue of security, m-commerce applications introduce new and significant privacy risks to end users. First, consider that most users currently own their wireless devices—cellular phones, PDAs, and so forth. As a result, users feel much more comfortable entering personal data on these devices than they would on corporate machines, which could be monitored by their employers. Through a device's Internet connectivity, the privacy of users' data and Web activities is at risk. The public is becoming increasingly aware that major marketing and data collection firms such as DoubleClick and Engage are tracking users' online Web usage via cookies with unique identifiers . In the wireless Web, we will certainly still have tracking of Web usage and even more profiling is possible. In March 2000, it was disclosed that AT&T Wireless and Sprint PCS were sending users' phone numbers to the Web sites they accessed from their Web-enabled wireless phones. As a result, these Web sites can now track end users by personal identifying information such as phone numbers as well as potentially using those phone numbers for offline direct telemarketing.

Other privacy risks of Internet-enabled wireless devices abound. Many m-commerce applications use personal preferences to provide value-added services. By specifying your favorite cuisines, for example, an online service can download information (including directions) for restaurants in an unfamiliar city to a handheld device. Location-oriented services provide the ability for online Web sites, including marketers, to determine your geographic location with a certain degree of precision. Combining this data, an online marketer will know not only a user's personal preferences such as cuisine and music, but also the user's precise geographic location at all times. This data can be used, perhaps, to send coupons for a participating merchant to the user while passing the merchant. Many consider these value-added services, while others consider collection of this information to be invasive.

Finally, consider that the physical limitations of handheld devices make it unlikely users will spend their limited time and bandwidth downloading and reading the legal jargon written in privacy policies on their small displays. Therefore, posting privacy policies for wireless Web sites will be insufficient, and perhaps make a viable case for P3P-type solutions to automatically reach agreements on privacy policies and preferences or to notify users of disagreements before proceeding .

In summary, mobile e-commerce systems will introduce new security and privacy risks beyond those currently found in desktop e-commerce systems. Using wireless devices for m-commerce will result in new vulnerabilities and potentially represent a new weak link in e-commerce. Since attackers tend to exploit the weakest link in a chain, the security risks of wireless devices must be carefully analyzed and addressed.



VOL. 1 ISSUE 3, DEC.-2012 ISSN NO: 2319-7463

Addressing the Software Risks

Much ado has been made about the security of wireless transport protocols such as the Wireless Application Protocol (WAP). The WAP advocates argue that the Wireless Transport Security Layer (WTLS) provides a secure infrastructure for m-commerce applications. Critics have decried the infamous "WAP gap" where wireless requests to Web pages are translated at the WAP gateway from the WTLS protocol to the standard SSL protocol widely used in secure HTTP requests. In the process of translating one protocol to another, the data is decrypted and then re-encrypted. If an attacker is able to compromise the WAP gateway, then simply capturing the data when it is decrypted can compromise the secure session. In reality, these issues are red herrings that draw attention away from the more substantive vulnerabilities in m-commerce systems: the software systems that run on both ends of the session.

The "WAP gap" problem will likely be solved in the near future by simple modifications to existing protocols. Traditionally speaking, data encryption over communication channels has been the strongest perceived security element in the system. As a result, malicious hackers tend to ignore the security provided by encryption protocols and simply attack the weakest links in the system, such as servers and clients. The problem of providing server-side security for wireless Web access closely mirrors the problem of providing server-side security in fixed-wire e-commerce and is well understood and documented . One exacerbating factor in wireless Web server security, however, is the fact that there are currently few wireless gateways or portals to the wired Web. Thus, those few gateways present ideal targets of opportunity or single points of failure for an attacker to bring down a significant portion of the wireless Web by selective denial-of-service attacks.

In the remainder of this article, we focus on the software security risks of wireless devices that present the most significant and least-understood security and privacy risks in m-commerce applications.

Platform Risks

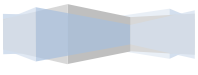
The platform or operating system that the device uses provides the basic infrastructure for running m-commerce applications. Without a secure infrastructure for computing on the device, achieving secure m-commerce may not be possible. Unfortunately, the manufacturers of many wireless devices have largely ignored the lessons learned from the past and have failed to include basic operating system features necessary to enable any kind of secure computing. Many manufacturers have failed to provide:

- memory protection for processes;
- protected kernel rings;
- file access control;
- authentication of principals to resources;
- differentiated user and process privileges;
- sandboxes for un-trusted code; or
- biometric authentication.

For example, one of the most popular PDA devices on the market does not provide memory protection for its applications. This failing poses serious threats for each application's own security and privacy. For instance, a trusted application that uses a private key for signing documents can be attacked by a rogue application. The rogue application can attempt to steal the decrypted key in the signing application's memory by interrupting it at just the right movement .

To address these platform risks, the wireless device operating system needs to enforce memory protection between applications to prevent one application from spying on another. The second fundamental protection necessary is access control for principals and objects to prevent unauthorized programs and users from accessing confidential data such as private keys or confidential information in databases. Wireless PDA platforms should also support encrypted tunnels or virtual private networks to provide confidential access over insecure wireless links to corporate systems. Finally, strong authentication mechanisms such as fingerprint recognition systems should be built into the devices to authenticate the user to the device. Similarly, software certificates should be used to authenticate software to the user before installing and running the software.

Software Application Risks



While the operating system provides the basic platform for wireless applications, the software applications that run on the device are equally important. Assuming that basic platform services listed previously are provided to applications, it is possible to design and develop secure wireless applications using good software engineering and assurance methods. The relation between software flaws and security vulnerabilities is well understood . The daily software bug postings to the BugTraq[1] list provide ample evidence of security holes introduced by software flaws.

Software development for wireless devices will be no different in this respect. Flaws in the logic and implementation can certainly result in security holes that will be exploited by attackers or malicious Web sites. Low-level languages typically used for development in handheld devices will ensure the continuation of basic flaws such as buffer overflow flaws . Furthermore, the physical limitations of a device often force application developers to make security and performance trade-offs. For instance, limited power, processing cycles, memory, and bandwidth will force application developers to forgo security features such as encryption in an effort to improve online performance. Security features in advanced languages such as Java may be omitted in vendors' JVM implementations. For instance, runtime checking of type safety is expensive, as is implementing fine-grained sandboxes and stack-introspection—security features implemented in current desktop JVMs.

One of the most interesting software-related developments in wireless devices is the ability to send and execute mobile code. In the wired world, mobile code is used pervasively in Web pages. Though Java applets brought mobile code to the forefront, by far the most common form of mobile code is JavaScript or Microsoft JScript. Scripting is used extensively in Web pages to validate forms and create the look and feel of Web pages.

For several reasons, scripting will also have ample uses in the wireless Web. First, client-side processing is attractive for reducing the number of communication hits necessary on extremely bandwidth-limited wireless links. For instance, client-side form validation reduces unnecessary server-side error reports and reentry messages. Second, some server-side processing can be off-loaded to clients using mobile code that will increase the availability of servers to more simultaneous connections. Third, scripting will be pervasive in wireless Web computing for the same reason it has become ubiquitous in wired Web pages: Web page development heavily leverages JavaScript for display functions and client-side transaction processing.

WML Script

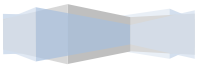
WML Script is the WAP equivalent of JavaScript. More accurately, WML Script is based on JavaScript, uses similar syntax and constructs, and provides semantically equivalent functions . However, WML Script was specifically developed for capability-limited WAP-enabled wireless devices. We use WAP-enabled mobile phones as our case study and WML Script as our example scripting language. We believe the malicious scriptware threat to wireless devices will become increasingly important and destructive.

WML Script is called from Wireless Markup Language (WML) pages, the wireless surrogate for HTML derived from XML. The primary reason WML and WML Script were developed was to work with the limited display, bandwidth, processing and storage capacities of wireless handheld devices such as cellular phones. While WML Script is optimized to work with the bandwidth- and capability-limited facilities of WAP-enabled devices, it is designed to attempt to replicate JavaScript functionality as much as possible. In addition, specific libraries are distributed with WML Script engines to provide access to device-specific facilities, such as telephony functions .

One of the main reasons for using WML Script is to provide a uniform interface to wireless applications and functions that is independent of the device brand . Until recently, most functions available on cellular phones were native and built-in by the manufacturer. For instance, voice mail, call management and personal address books, among other myriad functions, were device-specific and varied by manufacturer. Given the number of different phone manufacturers, these differences provided significant challenges to wireless phone service providers in terms of both compatibility and usability for wireless customers. The industry is now quickly moving to provide these features via WML Script using the WML Script interpreter as the standard development platform across different manufacturers' devices. The idea is that the interface and functionality for these different wireless services will be the same regardless of which particular brand of wireless device a customer chooses. This would also make it possible to use different phones without requiring a user to reprogram personal preferences, data, and functionality.

Security Risks of WML Script

Like the developers of the wireless device platforms, the developers of WML Script have ignored the lessons learned from past security problems with JavaScript and other mobile code technologies. The security risks associated with WML Script are based on a fundamental lack of a model for secure computation. The WML Script specification does not call for the distinction most JavaScript interpreters make between trusted local code and untrusted JavaScript downloaded from the Internet . As a result, WML Script is given the same amount of



VOL. 1 ISSUE 3, DEC.-2012 ISSN NO: 2319-7463

access whether it is downloaded from a trusted service provider, is built into the phone, or is downloaded from untrusted Internet WAP content providers. The lack of access control for WML Script means the types of attacks that can be launched using WML Script will be limited only by the imagination of malicious script writers.

For purposes of efficiency, WML Script is compiled into a WML Script byte-code, which is downloaded by the client, and run on a WML Script virtual machine or interpreter. Though the wireless industry uses the terms "byte-code" and "virtual machine," these should not be misconstrued to embody the same safety properties of Java or to enforce a sandboxing mechanism.

Indeed, WML Script is not a type-safe language, nor does the VM or interpreter appear to enforce any kind of sandboxing mechanism to prevent WML Script from accessing persistent storage or making network accesses . In fact, a WML Script can download remote WML Scripts using standard URL requests. Though URL domain/path checking is performed before a script can be downloaded and run, the check is a server-side check and will not prevent client-side damage from a malicious script. Furthermore, WML Script can be pushed to a device—using scheduled pulls from Web pages or other WML scripts—without the owner's knowledge. Unlike the JavaScript interpreters found in most desktop browsers, the WML Script virtual machine does not appear to have a mechanism for preventing access to persistent storage on a device from untrusted scripts . As a result, personal identifying information kept on a device is susceptible to unauthorized disclosure from malicious WML scripts that download and read the personal information and then ship it off to other sites. However, given the limited amount of current storage capability on phones, this may be the least interesting of attacks.

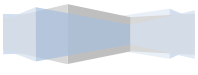
More interesting attacks will involve online application duplicity and email virus attacks. Consider that many online applications require users to authenticate themselves using an account identifier and password. Using WML Script, an attacker can send the user an email message with the account holder's favorite banking page. The script can be used to rewrite all the links on the page to redirect the user to a page under their control where they can ask the user to re-authenticate, grab the user name and password, then forward the user on to the real bank page. The malicious site can then use captured user accounts and PINs for its own malicious purposes. Another example of up-and-coming WAP applications will be the ability to charge purchases in real time (for example, purchasing movie tickets) to a wireless phone bill, to e-cash stored on the phone's smartcard, or to an online bank account. Scripts will be used heavily for these types of online transactions for processing the client-side portion of the transaction. Malicious scripts will have the ability to falsely ring up charges or potentially off-load money from smartcards or bank accounts.

Since WML Script can access persistent stores, it will be fairly trivial to write WML scripts that can destroy user data stored on a device. As the persistent store sizes grow and more applications are ported to Web-enabled devices (for example, document-processing applications), the value of the data stored in persistent storage will grow, providing virus writers more incentive to write damaging scripts. Script worms will also be written to leverage the Web, chat rooms, and email capabilities to propagate to wireless devices. Initially, these worms will simply propagate from phone to phone, consuming precious wireless bandwidth and possibly running up end users' phone charges. However, since access to device storage is not well constrained, future scripts will behave maliciously, shipping off or destroying valuable personal data, for instance.

Telephony applications present another interesting area subject to exploitation from malicious scripts. WML Script provides access to telephony functions through the Wireless Telephony Application Interface (WTAI) . Access to a phone's telephony facilities allows online service providers to:

- accept/initiate calls;
- send/receive text messages;
- add/search/remove phonebook entries;
- examine call logs;
- send tones during calls; and press keys on keypad during a call.

Realizing the potential security risks of WML Script accessing a telephone's telephony functions, the Wireless Telephony Application (WTA) services rely on two assumptions to provide security . First, it is assumed the user will only visit trusted WAP gateways wherein a WTA server may run. The WTA server is simply a server that delivers WML Script over WAP connections that can access a phone's WTA functions. If the user hits an un-trusted WAP content provider that sends WML Script with WTA functionality, it is possible for the WML script to make random phone calls, send personal data through a phonebook, or even erase a phonebook. Hence, the security model here holds that a user should hit only trusted WAP sites. If WAP-enabled devices and services grow as predicted, this assumption will rapidly become untenable, as many vendors will launch their own WAP gateways.



VOL. 1 ISSUE 3, DEC.-2012 ISSN NO: 2319-7463

The second assumption made for secure functionality is that a user will securely configure his or her device to prevent giving blanket permission for any WML script to access WTA functions. In fact, there are three permission types available for accessing WTA functionality: blanket permission to access all functions through the WTAI, context permission to run a given WTA function within a current execution context, and single-action permission for one-time access to a WTA function .

Finally, it is worth mentioning that the WTA specifications do not specify any default permission settings. Rather, mobile service providers determine these settings. If history is any indication, service providers will preconfigure devices with liberal permissions to permit access to their own scripts without regard for other potentially malicious scripts.

In short, the WML Scripting capability built into WAP 1.2 compliant devices will provide a fertile breeding ground for the current generation of malicious scripts that run unabated in desktop platforms. Furthermore, without any foundational security model, the severity of attacks against wireless devices will increase as these devices become more critical to users and businesses for both storage and processing of confidential information.

CONCLUSION

While many of the risks of desktop Internet-based commerce will pervade m-commerce, m-commerce itself presents new risks. The nature of the medium requires a degree of trust and cooperation between member nodes in networks that can be exploited by malicious entities to deny service as well as collect confidential information and disseminate false information. Furthermore, the platforms and languages being developed for wireless devices have failed to adopt fundamental security concepts employed in the current generation of desktop machines.

Encrypted communication protocols are necessary to provide confidentiality, integrity, and authentication services for m-commerce applications. Perhaps the greatest risk of encrypted communication links, though, is the false sense of security they provide wireless users and purveyors of m-commerce.

Probably the most significant risk to m-commerce systems will be from malicious code that is beginning to penetrate wireless networks. Malicious code has the ability to undermine other security technologies such as signing, authentication, and encryption because it runs resident to the device with all the privileges of the owner.

The risks presented by malicious mobile scripts to wireless devices are significant. As illustrated here, wireless device manufacturers and language developers have ignored past lessons learned with regard to security and privacy risks in mobile code. Our goal here is to highlight the key security and privacy risks already apparent in these devices and their language platforms in order to influence device and platform manufacturers to build more robust and secure systems. It is important to note here that though the current version of WAP, 1.2 uses WML and WML Script as its language and partner script, WAP version 2.0 is scheduled to be released within the next year. Version 2.0 may retire WML and WML Script in favor of a more robust language such as XHTML and a similar partner script, possibly JavaScript. While we expect many of the same security risks to apply to the scripting language of choice for version 2.0, we hope the specification addresses the weaknesses highlighted in this article.

The best strategy for addressing the security and privacy risks of Internet-based content is to build security into the platform and applications themselves, rather than attempt to introduce security patches afterward. For instance, Java provides type safety, memory protection, and sandboxing for un-trusted content. While history has shown that various implementations of the Java virtual machine have not been perfect, its model of secure computation is relatively good. The device manufacturers and the language developers for wireless applications should leverage the decades of progress in secure operating system models and secure models of computation before going forward with business-critical and privacy-related wireless applications. Otherwise, we are doomed to repeat the mistakes of the past, and potentially take two steps backward as we move one step forward.

About Author

Yeddula Venkatramana Reddy received his Master degree in Computer Applications from Sri Venkateswara University, Tirupathi, India and M. Tech from Andhra University, Visakhapatnam, India.. Now he is in Nalla Malla Reddy Engineering College, Hyderabad, India. He is an IEEE member and many paper published in various journals/conferences.