Potential Threats to WAP Enabled Devices

Eric Chien

ABSTRACT

The number of WAP-enabled devices is on the rise. Cellular phones are the first widespread devices to adopt WAP (Wireless Application Protocol) functionality. With WAP, one can access a variety of services from news reports to currency exchange.

New services are being developed everyday and as more robust scripting becomes possible, the chance for malicious code increases.

WAP currently passes WML (Wireless Markup Language) and WMLScript (Wireless Markup Language Script) which is interpreted by the WAP device. Both languages are not robust enough to pose a threat today. However, as new functionality is added, future threats may be possible.

This paper will review WAP, WAP Push, WML, and WMLScript and their ability to host malicious software. Exploits such as buffer overflows, authentication, and encryption schemes are outside the scope of this paper.

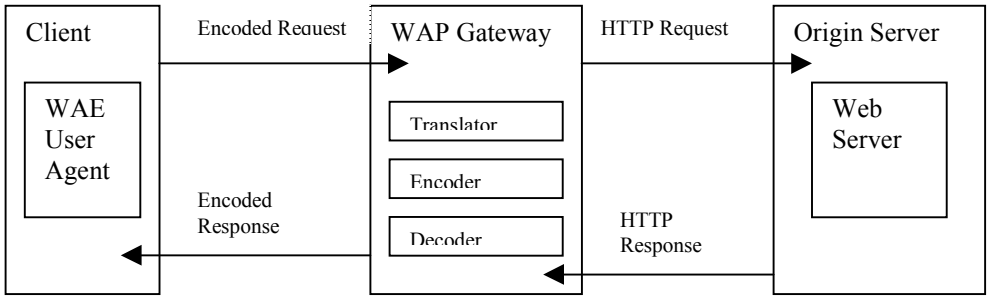
Finally to detect such future threats, potential solution placement will be discussed.

1 Introduction

The growth of media on the Internet is motivated by high-speed bandwidth. Concern for small streamlined content is decreasing with the advent of cheap 24x7 high-speed connections for both home and businesses. However, computing devices are becoming smaller and more portable with a relatively lower bandwidth to Internet content -- thus, the desire for a smaller media format. WAP specifically addresses these needs with WML and WMLScript, which is designed for small display sizes, limited input devices, low resource devices, and narrow-band connectivity.

2 The WAP Model

The WAP programming model is similar to the current HTTP model. WAP consists of a client, gateway, and origin server.



1. The user makes a request for content.
2. The WAE (Wireless Application Environment) User Agent encodes the request.
3. The WAE User Agent sends the request to the Gateway.
4. The Gateway decodes the request.
5. The Gateway translates the request (to HTTP) and sends it to the Origin Server.
6. The Origin Server passes back the appropriate content.
7. If necessary the Gateway translates the content (HTML to WML).
8. The Gateway encodes the content.
9. The Gateway sends the encoded content to the Client
10. The WAE User Agent interprets the encoded content and presents it to the user.

3 Wireless Markup Language

Wireless Markup Language is an XML (Extended Markup Language) language. WML adheres to the Universal Character Set, ISO/IEC-10646 specification. WML syntactic constructs are inherited from XML. For this reason, WML appears similar to HTML and conversion from HTML to WML may be trivial in many cases. WML is a static language and thus, provides limited properties to form malicious content. The current version of WML is 1.3.

WML content is often addressed as WML decks and cards. A deck is analogous to a book in which there are many pages, or cards. On the origin server, a single file represents a deck and may contain multiple cards. Cards are designated with card tag deliminators.

```
 <card id=first>
        <!—First Card Content Here →
</card>

<card id=second>
        <!— Second Card Content Here →
</card>
```

A request for content generally returns the whole file and thus, the deck. This allows the user agent to request and more importantly re-request cards without re-contacting the WML gateway.

Here is an example of a sample WML file or deck.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
        <card id="card1" ontimer="#card2" title="Multiple Card Demo">
                <timer value="100"/>
                <p align="center">
                        Card 1
                </p>
        </card>

        <card id="card2" title="Multiple Card Demo">
                <p align="center">
                        Card 2
                </p>
        </card>
</wml>
```

4 Wireless Markup Language Script

Wireless Markup Language Script is an extended subset of JavaScript. WMLScript allows scripting ability with the WAP framework and is based loosely on ECMAScript. The purpose of WMLScript is to extend the capabilities of WML. WML is static and thus, there is no way to modify or extend WML without modifying the source itself. For example, WMLScript could be utilized to check the validity of user input locally, eliminating a round-trip to the WML gateway. In addition, in the future, WMLScript may be utilized to access facilities of the device, such as making phone calls, sending messages, and accessing the address book or SIM card.

WMLScript provides the framework. WMLScript Standard Libraries provide the extended functionality. The current version of WMLScript is 1.2. The current version of the WMLScript Standard Libraries is 1.3.

Here is an example of a sample WMLScript file.

```
extern function foobar(foo,bar) {

        if (foo == bar) {
                Dialogs.alert("Foo is equal to bar!");
        } else {
                Dialogs.alert("Foo does not equal bar!");
        }
}
```

5 WAP Code Threats

By the strictest definitions there can not be WAP viruses. WAP is simply the protocol like SMTP (email). However, there potentially could be WML or WMLScript viruses or related malware.

5.1 WML

The latest specifications of WML rule out the possibility of a virus. However, there is the possibility for joke or trojan type content.

For example, a WML card named X could auto-load another WML card named Y. The WML card Y, could in return auto-load X leaving one in an infinite loop. This type of 'trick' depending on the device can be canceled or one can quit browsing. Upon restart of the browser, one can start back at the Home page and then avoid visiting WML cards X or Y.

If the device resources become overloaded or if cancel or quit features are unavailable, one may have to reset the device.

Another example could be a WML page that appears to contain legitimate information. Unexpectedly, the page could redirect one to a screen, which prompts one for their PIN code, which appears to be generated from the phone. However, this screen instead is a form that sends the PIN code to an anonymous web site.

Both scenarios while possible are extremely low risk. Both require the user to specifically visit the pages and the resultant imposition is negligible.

Future revisions of WML probably will remain generally unexploitable. WML by nature is static limiting the possibility for replicating malicious code. Currently, there are no HTML viruses and thus, WML viruses are highly unlikely.

5.2 WMLScript

The latest specifications of WMLScript rule out the possibility of viruses. However, similarly to WML, there is the possibility of joke or trojan type content. Both previous examples of infinite loops or fake content can be performed with WMLScript.

Future specifications alluded to by current documentation of WMLScript may allow viruses or more specifically worms and potentially more dangerous trojans. The documentation clearly states:

> *The following list contains some capabilities that are not supported by WML:*
> …
> *Access to facilities of the device. For example, on a phone, allow the programmer to make phone calls, send messages, add phone numbers to the address book, access the SIM card etc.*

...

WMLScript was designed to overcome these limitations and to provide programmable functionality that can be used over narrowband communication links in clients with limited capabilities.

Having programmable access to the device raises the possibility for worms. For example, one may view WML content that secretly sends an SMS with the URL to all your contacts. They, in turn, believing you sent the URL, view the page and likewise unknowingly send all their contacts the URL. In addition, having access to information on the phone may allow Trojans to gather information such as contact phone numbers and names and send them to anonymous websites.

5.3 Injectors

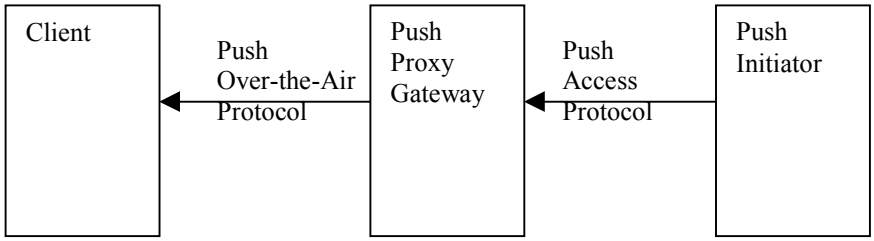
There is a possibility of an injector. Most WML files are stored on a traditional web server. This web server could be infected with malware, which injects malicious code into WML(Script) files. This allows the malicoius WML(Script) code to spread to many WML(Script) hosting sites via a traditional virus.

For example, a Windows worm could be constructed to inject the above code examples into any WML(Script) files that are found. This would allow the malicious code to spread via the Windows virus.

However, a true cross infector would not be possible. That is, the modified WML(Script) files would not be able to recreate the Windows virus.

6 WAP Push

The WAP specification defines a push architecture, which allows information to be transmitted to a device without an explicit request from the device. The following illustrates the WAP push architecture.



A real-life example may be a real-time new mail indicator. The Push Initiator may be an e-mail server. The e-mail server via PAP (Push Access Protocol) notifies the PPG (Push Proxy Gateway) that there is new mail for the Client. The PPG verifies identity of the e-mail server and that the client has requested such services. Then, via PushOTA (Push Over-the-Air Protocol), the PPG passes a Service Indicator or other content to the Client. The Client then processes the Service Indicator (which may include ignore and identity verification) and potentially launches a service, such as a request for e-mail.

The WAP Push architecture allows for any content-type delivery although, the Client and PPG may limit the acceptable content-type. Clearly, the acceptance of executable content that may be malicious provides a potential vector of delivery. However, today the majority of Push implementations are limited to Service Indicators described below.

6.1 Push Proxy Gateway

The PPG is responsible for authentication of the Push Initiator including verification that the Client requested such services. This prevents potential spamming and denial-of-service attempts on the Client. In addition, this reduces a vector of delivery of potentially malicious code to the Client. Should there be a bug

in the verification scheme of the PPG, clearly this presents a major vector of delivery to the Client. A rogue Push Initiator could transmit malicious code to a Client unbeknownst to the client user.

6.2 Push Content and Service Indicators

The WAP Push architecture allows for any content to be passed. However, current implementations generally restrict content to Service Indicators. A Service Indicator is an application of XML 1.0. A Service Indicator itself can not hold a virus. A Service Indicator provides a small amount of information to the client such as notification that new mail has arrived. In addition, the Service Indicator may provide additional information such as the mail server to contact for receipt of the new mail.

6.3 Client Interaction

The Client may also ignore push content even if already authenticated by the PPG. Service Indicators are generally the majority of push content. In addition to providing notification, a Service Indicator can request the Client invoke a service such as a mail reader.

Being able to remotely activate services on a Client obviously provides for potential abuse. However, in addition to the authentication of the Push Initiator, any invocation of Client services must be preceded by user interaction, according to the specification. This eliminates the possibility of rogue Push Initiators executing code remotely via WAP Push. Obviously, any bugs in this scheme could lead to potential abuse.

7 Solution Placement

This section does not discuss the technical solutions of how to detect such threats. Products with the ability to detect current JavaScript or Visual Basic Script threats should require little if any modification to detect future WML and WMLScript threats. This section only discusses the advantages and disadvantages of solution placement.

7.1 On-Device Solution

Placing a scanner on the device can be effective in blocking content, but may be resource intensive and difficult in regards to development. WAP devices vary from phones to personal digital assistants (PDAs) and each of the devices may be running on non-updatable and proprietary operating systems. One could not create a single solution that could be placed on all devices. In addition, updating the firmware or software may be problematic.

While possible, on-device solutions do not seem practically viable.

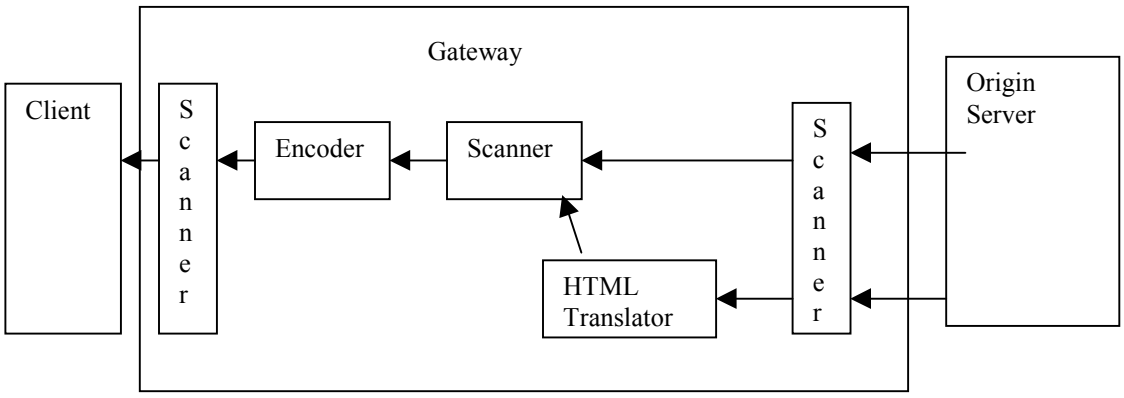
7.2 Origin Server Solution

Origin servers are generally run on popular operating systems such as Unix or NT. Existing products can scan these servers. The existing products would need to be updated with new signature files to detect malicious WML(Script) code, but this would be trivial for any scanner that can currently handle JavaScript or Visual Basic Script threats.

However, limiting the solution to the origin server does not adequately protect the end user. While the origin server administrator could assure his systems are free from malicious content, the end user could not be assured any origin server visited is free from malicious content.

7.3 WAP Gateway Solution

A solution at the WAP gateway seems ideal. All WAP content must pass through a WAP gateway before reaching the device providing for a single point of protection. At the gateway a scanner could be placed in multiple places. A scanner could be placed after the receipt of WML or HTML content from the origin server or after the conversion of WML or HTML to binary representation. The following diagram represents the potential placement locations at the gateway.



Most anti-virus products currently have the technology to detect WML or WMLScript threats. These engines would simply need to be wrapped to interact with the WAP gateway components.

8 Summary

While currently, WML(Script) does not pose a great threat for malicious content today, it may in the future. Before such threats are possible, the WML(Script) specifications must provide more robust functionality. Once the functionality is possible, WML(Script) malware still may not be a large threat. Other environmental factors from market penetration to social engineering play a role in the ability of threats to spread. The role of these factors is outside the scope of this document.

In general, security vendors should already have the technology to detect such threats. Security product engines need to be wrapped to interact with the appropriate components or run on the appropriate operating systems.

Finally, users should extend their safe computing practices to such devices and avoid questionable executable content.

REFERENCES

1. WAP Architecture, Version 30-Apr-1998, Wireless Application Protocol Forum, Ltd.
2. WAP-193-WMLScript Language Specification, June-2000, Wireless Application Protocol Forum, Ltd.
3. WAP-194-WMLScript Standard Libraries Specification, June-2000, Wireless Application Protocol Forum, Ltd.
4. WAP WML, WAP-191-WML, 19 February 2000, Wireless Application Protocol Forum, Ltd.
5. WAP Service Indication, Version 08-Nov-1999, Wireless Application Protocol Forum, Ltd.
6. WAP Push Architectural Overview, Version 08-Nov-1999, Wireless Application Protocol Forum, Ltd.
7. WAP PPG Service, Version 16-August-1999, Wireless Application Protocol Forum, Ltd.
8. WAP Push Access Protocol, Version 08-Nov-1999, Wireless Application Protocol Forum, Ltd.
9. WAP Push OTA Protocol, WAP-189-PushOTA, Version 17-Feb-2000, Wireless Application Protocol Forum, Ltd.
10. WAP Service Loading, Version 08-Nov-1999, Wireless Application Protocol
11. http://www.wapforum.org
12. Nokia Toolkit 2.0
13. UP.SDK, Phone.com Software Development Kit
14. http://developer.phone.com
15. WML Reference, Version 1.1, Nokia
16. WMLScript Reference, Version 1.1, Nokia
17. Phone.com emulator