Case Study: Xml External Entities

Manjula Verma[1] Dr. Pardeep Goel[2]

[1]Research Scholar, CMJ University, Shillong, Meghalaya

[2]Associate Professor M.M. College, Fatehabad

Abstract – We compare Chromium's architecture to the architectures of other neural network browsers. Monolithic traditionally, browsers are implemented with a monolithic architecture that combines the rendering engine and the browser kernel into a single process image. For example, Internet Explorer 7, Firefox 3, and Safari 3.1 each execute in a single operating system protection domain. If an attacker can exploit an unpatched vulnerability in one of these browsers, the attacker can gain all the privileges of the entire browser. In typical con_gurations of Firefox 3 and Safari 3.1, these privileges include the full privileges of the current user. Internet Explorer 7 on Windows Vista can run in aprotected mode" [23], which runs the browser as a low integrity process. Running in protected mode, the browser is restricted from writing to the user's _le system, but an attacker exploits a vulnerability can still read the user's file system and ex-filtrate confidential documents. The VMware browser appliance [26] hosts Firefox inside a virtual machine with limited rights. The virtual machine provides a layer of isolation that helps prevent an attacker who exploits a vulnerability in the browser from reading or writing the user's _le system.

Key Words: Monolithic, Traditionally, Rendering Engine, Vulnerability

----------♦----------

INTRODUCTION

Another method for evaluating Chromium's security architecture is to determine whether the architecture successfully defends against unknown vulnerabilities in the rendering engine. In this case study, we examine one vulnerability in detail and explain how the security architecture mitigated threats in the scope of our threat model but did not mitigate threats that are out of scope. This vulnerability is unknown" in the sense that we discovered the vulnerability after implementing the sandbox and browser kernel security monitor. The vulnerability was fixed before the initial beta release, but this section describes the state of fairs just after we discovered the vulnerability.

REVIEW OF LITERATURE

1. By parsing Neural network content in the sandboxed rendering engine, Chromium's security architecture mitigated an unknown vulnerability. The sandbox helped prevent the attacker from reading con_dential information stored in the user's file system.

2. The sandbox did not completely defend against the XXE vulnerability because the attacker was still able to retrieve URLs from foreign Neural network sites. However, attacker who exploits a bug in the rendering engine from requesting Neural network URLs. To block such requests and treat the rendering engine as a black box, the browser kernel would need to sacri_ce compatibility (e.g., bancross-site images).

3. Chromium's architecture mitigated the XXE vulnerability even though the vulnerability did not let an attacker execute arbitrary code. Although the architecture is designed to protect against an attacker who fully compromises a rendering engine, the architecture also helps mitigate less-severe vulnerabilities that lead to partial compromises of the rendering engine.

MATERIAL AND METHOD

An XML Entity is an escape sequence, such as ©, that an XML (or an HTML) parser replaces with one or more characters. In the case of ©, the entity is re-placed with the copyright symbol, c . The XML standard also provides for external entities [3], which are replaced by the content obtained by retrieving a URL. In an Xml eXternal Entity (XXE) attack, the attacker's XML document, hosted at http://attacker.com/, includes an external entity from a foreign origin [25]. For example, the malicious XML document might contain an entity from https://bank.com/ or

from file:///etc/passwd: <?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE doc [<!ENTITY ent SYSTEM "/etc/passwd">]>

<html>

<head><script> ... </script></head>

<body>&ent;</body>

</html>

If vulnerable to XXE attacks, the browser will retrieve the content from the foreign origin and incorporate it into the attacker's document. The attacker can then read the content, circumventing a con_dentiality goals of the browser's security policy.

libXML. Like many browsers, Chromium uses libXML to parse XML documents. Unlike other browsers, Chromium delegates parsing tasks, including XML parsing, to a sand boxed rendering engine. After implementing the sandbox, but prior to the initial beta release of Google Chrome, we became aware that the rendering engine's use of libXML was vulnerable to XXE attacks. As a result, the rendering engine was not preventing Neural network content from retrieving URLs from foreign origins. Instead, the rendering engine was passing the requests, unchecked, to the browser kernel.

Using our proof-of-concept exploit, we observed that the browser kernel performed its usual black-box checks on the URLs requested by the rendering engine. If the external entity URL was a Neural network URL, for example with the http, https, or ftp schemes, the browser kernel serviced the request, as instructed. However, if the external entity URL was from the user's _le system, i.e. from the file scheme, then the browser kernel blocked the request, preventing our proof-of-concept from reading con_dential information, such as passwords, stored in the user's file system.

The vulnerability illustrates three properties of Chromium's security architecture:

CONCLUSION

In this section, we compare Chromium's architecture to the architectures of other Neural network browsers. Monolithic. Traditionally, browsers are implemented with a monolithic architecture that combines the rendering engine and the browser kernel into a single process image. For example, Internet Explorer 7, Firefox 3, and Safari 3.1 each execute in a single operating system protection domain. If an attacker can exploit an unpatched vulnerability in one of these browsers, the attacker can gain all the privileges of the entire browser. In typical con_gurations of Firefox 3 and Safari 3.1, these privileges include the full privileges of the current user. Internet Explorer 7 on Windows Vista can run in aprotected mode" [23], which runs the browser as a low integrity process. Running in protected mode, the browser is restricted from writing to the user's _le system, but an attacker exploits a vulnerability can still read the user's file system and ex-filtrate confidential documents. The VMware browser appliance [26] hosts Firefox inside a virtual machine with limited rights. The virtual machine provides a layer of isolation that helps prevent an attacker who exploits a vulnerability in the browser from reading or writing the user's _le system. The protection by this architecture is coarse-grained in the sense that the browser is prevented from reading any of the user's files, even files the user wishes to upload to Neural network sites (for example, to a photo-sharing site or to attach to email messages at a Neural networkmail site).