Christopher Morcom

CS 558

Project Proposal

**Dynamic Data Exchange Exploits for Botnet Entry Points**

**Overview**

The Dynamic Data Exchange (DDE) Protocol is a tool designed for intercommunication between Microsoft Office programs. It is a protocol enabled by default for pre-December 2017 editions of Microsoft Office, but is an exploit that is still available today. DDE run malicious code by exploiting attachments or macros in MS Office programs, which provided an active ingress point for a variety of attacks. Use of this attack occurs through memory corruption vulnerabilities (CVE-2015-1650 and CVE-2016-7231) which allow attackers to install malware at elevated user privilege levels.

**Background**

DDE is a protocol developed in 1987 for Windows 2.0 which exchange string handles or ‘keys’ in applications through a DDE Management Library where requests are broadcasted to other applications in the system. Prior to December 2017, DDE was force-implemented to ensure compatibility across applications. DDE was superseded by Object Linking and Embedding Database (OLE DB) and Open Database Connecting (ODBC) which are generally used for database modeling but are flawed in similar ways. On 12 December 2017, a patch disabling DDE functionality was released, but allowed users to change registry keys to allow DDE requests based on user needs. Similarly, on 20 September 2018, a remote-code execution vulnerability was published regarding an out-of-bounds write in the MS JET Engine through its OLE DB provider and MS Access ODBC. Although these patches have been fixed, they only apply to systems that have installed the MS post-2017 update which is problematic for many Windows 8 users on MS Office 2010 and prior. However, Microsoft has left options in for users to restore DDE along with exploiting OLE DB in modern applications.

**Problem Statement**

By exploiting vulnerabilities in current systems, we propose an experimental approach using a variety of ingress points to try to find optimal attacks that answer the question:

*How can we modify distribute DDE attacks to infect a series of hosts with botnets for exploitation and what are some methods to defend against this?*

**Project Approach**

*Phase I: Understanding and Creating a Sustainable, Experimental Environment*

DDE exploits have been patched by Microsoft, however, because similar memory table vulnerabilities exist in many MS and Windows applications today, understanding how DDE communication works in a stable environment without network access is ideal for analyzing effectiveness and time complexity of a DDE attack.

*Phase II: Performance-Based on Vulnerabilities*

DDE Exploits will have different performance metrics and capabilities based on level of access, which is largely determined by the ingress point used. For instance, exploiting an AllowDDE Registry value of 1 will not allow us to use DDE to execute other programs on the victim. However, exploiting autorun.inf on a hot-swappable or boot drive will allow us to set the Registry value to 2, which will allow a much larger variety of attacks to occur. Exploiting attacks on a virtual machine will yield high asymptotic results than that in a real-world analysis. For the sake of this paper, we will force exploits to be present in some windows systems and use existing tools to create attacks.

*Phase III: DDE Evaluation*

In this phase, we analyze our results and attempt to create an attack on a “modern” machine and dictate parameters that need to be fulfilled in order to successfully launch an attack.

*Phase IV: Attack Distribution*

Based on the analysis in *Phase III*, we will analyze optimal methods of distributing the DDE attack across a network. A common, popular method in past attacks was through malware embedded in MS Office documents.

*Phase V: Defense and Analyses*

Potential defenses for DDE exploits that can be added to existing MS Windows patches can be explored, along with adjustments for screening files for self-executing exploits (where the user may accidentally grant permissions to an attacker).

**Related Works**

Because DDE, ODBC, and OLE DB exploits are rather infrequent and hard to pull off, I was unable to find any studies about these protocols/APIs, however, there are a number of studies about preventing memory error exploits with a decreasing runtime overhead. In 2008, there was a study posted about Write-Integrity-Testing (WIT) to prevent instructions from modifying objects not in the computed static set of objects, and to prevent erroneous control transfers with a 7% runtime overhead. In August 2017, a similar study was published for IoT devices, giving a runtime overhead of 6%.

**References**

“About Dynamic Data Exchange.” *Microsoft*, Microsoft, 30 May 2018, docs.microsoft.com/en-us/windows/desktop/dataxchg/about-dynamic-data-exchange.

“ADV170021 | Microsoft Office Defense in Depth Update.” *Microsoft*, Microsoft, 12 Dec. 2017, portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV170021.

Barker, Sara. “Attackers Exploit Macro-Less Office Documents to Create Havoc.” *SecurityBrief*, TechDay Network, 3 Apr. 2018, securitybrief.co.nz/story/attackers-exploit-macro-less-office-documents-create-havoc/.

Elley. “Alert: A Remote Code Execution Vulnerability Is Discovered in Microsoft Windows Jet Database Engine.” *360 Total Security Blog*, QIHU 360 SOFTWARE CO. LTD., 26 Sept. 2018, blog.360totalsecurity.com/en/alert-a-remote-code-execution-vulnerability-is-discovered-in-microsoft-windows-jet-database-engine/.

Kedem, Migo. “Malware Embedded in Microsoft Office Documents | DDE Exploit (MACROLESS).” *SentinelOne*, SentinelOne, 6 July 2018, www.sentinelone.com/blog/malware-embedded-microsoft-office-documents-dde-exploit-macroless/.

P. Akritidis, C. Cadar, C. Raiciu, M. Costa and M. Castro, "Preventing Memory Error Exploits with WIT," *2008 IEEE Symposium on Security and Privacy (sp 2008)*, Oakland, CA, 2008, pp. 263-277.

doi: 10.1109/SP.2008.30