**Lance West – (Conficker Virus)**

Since April 1st is just around the corner at work we have been looking at ways to harden our servers against the Conficker virus, also known as the April Fools virus. It comes around each year, and every year we keep an eye out for it so it does not get a chance to infect anything. Who ever created this virus seems to be upgrading it from year to year. We now know that we have versions A/B/B++ and the new one version C. Over the last month I have been analyzing the virus in case it hits our E-commerce servers. In my analysis I wanted to use an inside-out approach that our book outlines and see if I could answer a few questions I had about the bug. How has it changed since it first showed up on the internet in 2009, and what’s in the new version? The parts of the book on As-Is and To-Be processes helped a lot in diagramming this fun little virus that has caused so many headaches in the IT world.

What are the differences in versions? There are a couple different versions of the Conficker virus now: A/B/B++ and now C. C distinguishes itself as a significant revision to the Conficker B virus.  C incorporates a major restructuring of B's previous thread architecture and program logic, including major functional additions such as a new peer-to-peer (P2P) coordination channel, and a revision of the domain generation algorithm (DGA). From nearly the moment of its initial outbreak in 2009, Conficker C has produced a visible effect on the Internet. Numerous infections of Conficker C's Win32 *dropper* executables have reportedly been delivered into pre-infected Conficker B hosts over the last year or so. The virus just does not go away, it gets re-written into a new form. One interesting and minimally explored aspect of Conficker is its early and sophisticated adoption of binary encryption, digital signatures, and advanced hash algorithms to prevent third-party hijacking of the infected population already on the web.

At its core, the main purpose of Conficker is to provide the authors with a secure binary updating service that effectively allows them instantly control millions of PCs world-wide.  Through the use of these binary encryption methods, Conficker’s authors have taken care to ensure that other hacker groups cannot upload arbitrary binaries to their infected drone population, and these protections cover all Conficker updating services that it uses to communicate across the web.

The primary delivery mechanism used to distribute variant C has been Internet rendezvous points, which had reportedly been blocked. But numerous examples of Conficker C's Win32 *dropper* executables have reportedly been delivered into pre-infected Conficker B hosts. When received by a B host, the *dropper* application performs the following actions: (Master Charts: As-Is Conficker C Virus upgrading Conficker B Hosts – Figure 1)

When the virus was first found in 2009 they saw that the virus used about 250 internal IP addresses to propagate itself on the web. So a group of security experts on the West Coast used their own money to buy the domain IP addresses which made it harder for the virus to spread. C's latest revision of Conficker’s now well-known Internet rendezvous logic may represent a direct retort to the action of the Conficker security group, which recently blocked all domain registrations associated with the A and B strains. C now selects its rendezvous points from a pool of over 50,000 randomly generated domain name candidates each day. (Master Charts: To-Be Domain Generation Algorithm (DGA) – Conficker C Virus – Figure 3).  C further increases Conficker's top-level domain (TLD) spread from five TLDs in Conficker A, to eight TLDs in B, to 110 TLDs that must now be involved in coordination efforts to track and block C's potential DNS queries. With this latest escalation in domain space manipulation, C not only represents a significant challenge to those hoping to track its census, but highlights some weaknesses in the long-term viability of how  Internet address and name space governance is conducted.

One interesting aspect of Conficker is its early and sophisticated adoption of binary encryption, digital signatures, and advanced hash algorithms to prevent third-party hijacking of the infected population.   At its core, the main purpose of Conficker is to provide the authors with a secure binary updating service that effectively allows them instant control of millions of PCs worldwide.  Through the use of these binary encryption methods, Conficker's authors have taken care to ensure that other groups cannot upload arbitrary binaries to their infected drone population, and these protections cover all Conficker updating services: Internet rendezvous point downloads, buffer overflow re-exploitation, and the latest P2P control protocol.

There are three crypto-systems employed by Conficker's authors (RC4, RSA, and MD-6) which all have one underlying commonality.  They were all produced by Dr. Ron Rivest of MIT.  Furthermore, the use of MD-6 is a particularly unusual algorithm selection, as it represents the latest encryption hash algorithm produced to date.  One major implication from the Conficker B and C variants, as well as other now recently emerging malware families, is the sophistication with which they are able to terminate, disable, reconfigure, or black-hole native operating system (OS) and third-party security services. The creators of the Conficker virus have shown advanced cryptographic skills, custom dual-layer code packing and code obfuscation skills, and in-depth knowledge of Windows internals and security products.

(Master Charts: As-Is Overview of Conficker C Virus – Figure 2)illustrates the Conficker C program structure and logic. When initialized, the DLL performs its setup logic, similar to that of A and B, with extensions. At initialization, it checks for the presence of three mutex values on the target host to avoid re-infection.  If absent, these three mutexes are created:

1. the mutex name "Global\<string>-7";
2. the mutex name "Global\<string>-99; and
3. a mutex named pseudo-randomly generated based on the process ID.  The <string> in the first two mutex is unique per computer name; it is calculated based on the crc32 hash of the computer name and XOR'ed with a constant.  C then installs several in-memory patches to DLLs, and embeds other mechanisms to thwart security applications.

Version C modifies the host domain name service (DNS) APIs to block various security related network connections (Domain Lookup Prevention), and installs a pseudo-patch to repair the 445/TCP vulnerability, while maintaining a backdoor for re-infection (Local Host Patch Logic). This pseudo patch protects the host from buffer overflows by sources other than those performed by the Conficker authors or their infected peers. Like Conficker B, version C incorporates logic to defend itself from security products that would otherwise attempt to detect and remove it. C spawns a security product disablement thread. This thread disables critical host security services, such as Windows defender, as well as Windows services that deliver security patches and software updates. These changes effectively prevent the victim host from receiving automated software updates. The thread disables security update notifications and deactivates safe-boot mode as a future reboot option. This first thread then spawns a new security process termination thread, which continually monitors for and kills processes whose names match a blacklisted set of 23 security products, hot fixes, and security diagnosis tools.

Then Conficker C installs itself into the user file system and configures the registry appropriately to invoke its DLL at host startup. It also inserts a variety of extraneous registry keys that are subsequently unused, presumably to cloak its presence. It copies itself into a randomly named DLL located in either the System32 directory, program files directory, or the user's temporary files folder. It deletes all restore points prior to its infection to thwart rollback. C then performs a simple validation of its DLL size, and suicides if this check fails. It sets the DLL's date to the same date as the local kernel32.dll, and sets NT File System (NTFS) file permissions on its stored file image to prevent write and deletes privileges. Once installed, the DLL spawns a remote thread, which it attaches to the netsvcs.exe or svchost.exe process, depending on the OS version.

The core elements of Conficker C are incorporated into two threads: a P2P communication thread, and the domain generation and Internet rendezvous point thread (Master Charts: Figure 2). The first thread is embodied in a code segment that has undergone an additional layer of code obfuscation, suggesting a desire by the Conficker authors to hinder its analysis, and thereby providing an obvious point for in depth inspection. The Peer-to-Peer logic (P2P) protocol includes an ability to coordinate with peers over TCP and UDP channels, as well as download and run digitally signed Win32 binaries. Incorporated with the P2P thread is anti-tracing logic that will kill the Conficker C process when run under a debugger. Conficker C also incorporates an HTTP date check function – (Master Charts: To-Be Main Peer-to-Peer Network Setup – Conficker C Virus - Figure 4).

Finally, C introduces a substantial modification of the Domain Generation Algorithm (DGA), and query procedure: (Master Charts: To-Be Domain Generation Algorithm (DGA) – Conficker C Virus - Figure 3). The(DGA) will be activated on 1 April, and before April 1st it will enter a loop that sleeps 24 hours and then rechecks the date via getlocaltime function.   Prior to entering the April 1st date check, C will sleep for an initial random interval between 30 and 90 minutes more specifically, if the local hour is after 11 a.m. and before 7 a.m. If the local time is after 8 a.m. and before 11 a.m., the sleep period will be between 2.5 and 3.5 hours.  It will then check for Internet connectivity, and if connected will enter the domain generation logic.   The next section describes this logic in greater detail.

The domain generation algorithm and query procedure have been significantly modified from previous versions of Conficker.  The reasons for these changes may be to address the recent actions of the (Conficker Security Group), as it has moved to block future registrations of Conficker A and B domains by buying the rights to the domains.  Among the key changes, Conficker C increases the number of daily domain names generated, from 250 to 50,000 potential Internet rendezvous points.  Of these 50,000 domains, only 500 are queried, and unlike previous versions, they are queried only once per day.  Furthermore, C provides significantly more filtering of the IP addresses produced by the DNS queries (Master Charts: Figure 3).

To resolve the set of domain names to IP addresses (Master Charts: Figure 3), C uses the standard Windows gethostbyname API function.  If the queried domain produces an IP address that passes C's 4-step filtering test, it will attempt to contact the IP address using port 80/TCP.  Here it uses a single call to the HttpQueryInfo API (i.e., an empty HTTP Get request).  If it succeeds in connecting to an authentic Conficker rendezvous point, the server will immediately send a digitally signed Win32 executable for the client to execute.   This variant does *not* produce the well-known Conficker search URL string (with q= or aq=).  Rather, the empty HTTP request may prove more difficult for IDS and network forensic signatures to identify.

The infected node is capable of acting like a server.  In this mode it will interact with the Conficker P2P network and distribute digitally signed files to other P2P clients.  There is a unique mapping from IP address to the two TCP and UDP listen ports in each host. This helps C avoid the need for super-nodes or a peer list (i.e., C requires no embedded hit-list to locate peers). DnsQuery patching allows Conficker to suppress connections to security companies that may patch and install tools that would otherwise remove Conficker from the host.

Domain Lookup Prevention at each process initialization, Conficker C applies an in-memory patch to dnsapi.dll (Windows XP, 2K) or dnsrslvr.dll (Vista).   It does not patch the DLL files on the filesystem, only their in-memory instances.  These DLLs contain the standard Windows APIs for domain name resolution and caching.  Conficker modifies Window's DNS lookup and cache services to prevent successful communications with various security product vendors and research sites.

Security Product Disablement(Master Charts: Figure 5)Upon first installation,  C installs itself and obfuscates its presence on the victim's host,. These steps allow it to avoid easy diagnosis and removal by an attentive user. It deletes all restore points prior to its infection to thwart rollback, and sets NTFS file permissions on its stored file image to prevent write and delete privileges.  Most of this logic also appeared in prior version, but here we find some extensions and updates.

In addition to disabling Windows Defender and the Windows error reporting service, this logic disables BITS *(Background Intelligent Transfer Service).*The BITS service is used to prioritize, throttle, and control asynchronous file transfers between machines using idle network bandwidth.  It is used by the Windows Update services and other software updater’s to stay current with the latest patches and security hot fixes.The disable security service (Master Charts: Security Product Disablement - Cause & Effect Chart – Figure 5**).**  This diagram illustrates the actual logic used to disable the security services within Windows.

Conclusion**:** At present the analysis of Conficker Variant C, which emerged on the Internet at roughly around 2009.  This variant incorporates significant new functionality, including a new domain generation algorithm and a new peer-to-peer file sharing service.   Absent from this discussion has been any reference to the well-known attack propagation vectors (RCP buffer overflow, USB, and NetBios Scans) that have allowed C's predecessors to saturate so much of the Internet.  Although not present in C, these attack propagation services are but one peer upload away from any C infected host, and may appear at any time and could be included in the next round of updates coming in April 1, 2012.

C is, in fact, a robust and secure distribution utility for distributing malicious content and binaries to millions of computers across the Internet. This utility incorporates a potent arsenal of methods to defend itself from security products, updates, and diagnosis tools.  It further demonstrates the rapid development pace at which Conficker's authors are maintaining their current foothold on a large number of Internet-connected hosts.  Further, if organized into a coordinated offensive weapon, this multimillion-node botnet poses a serious threat to the Internet user in the coming months.

 **Lance West**

**References:**

Sophos Anti-Virus (2012) : Tracking and finding Conficker infections: <http://www.sophos.com/support/knowledgebase/article/61259.html>

Eric Chien, "Downadup: Peer-to-Peer Payload Distribution," 2009.
[http://myitforum.com/cs2/blogs/cmosby/archive/2009/01/22/downadup-peer-to-peer-payload- distribution-symantec-security-response-blog.aspx](http://myitforum.com/cs2/blogs/cmosby/archive/2009/01/22/downadup-peer-to-peer-payload-distribution-symantec-security-response-blog.aspx)

Jose Nazario, "The Conficker Cabal Announced," Arbor Networks, 12 February 2009.
<http://asert.arbornetworks.com/2009/02/the-conficker-cabal-announced/>

SRI International, "A Comparative Assessment of Conficker B++ vs Conficker C," 06 March 2008. [http:/mtc.sri.com/Conficker/addendumC/HMA\_Compare\_ConfB2\_ConfC/](http://mtc.sri.com/Conficker/addendumC/HMA_Compare_ConfB2_ConfC/)

CAIDA, "Conficker/Conflicker/Downadup as seen from the UCSD Network Telescope," February 2009.
<http://www.caida.org/research/security/ms08-067/conficker.xml>

Conficker Working Group (2012) : DDoS and Security Reports: The Arbor Networks Security Blog **C:\AAA\_Spring\_Classes\AAA\_683 Class\Assignment 3\conflicker Virus\Conficker Working Group Lessons Learned Document \_ DDoS and Security Reports \_ Arbor Networks Security Blog.htm**

The Conficker Working Group (2012) : Lessons Learned Document <http://www.confickerworkinggroup.org/wiki/>