# Introduction

Since as early as 1949, the idea of a self-replicating computer program has been theorized (von Neumann, 1966). Today we see this theory put into practice by the number of computer viruses, trojans, and worms in existence. The usual purpose of these self-replicating programs is to infect vulnerable systems and execute a malicious payload such as file deletion, e-mail spamming, denial of service attacks, login credential stealing, and theft of money via online banking compromise (FoxNews, 2009). While in the past not all had a malicious payload, affecting the infected system and surrounding systems, the last few years have seen much more malicious and criminal intent in the programs' design.

Over the last few decades the traditional response to these malicious programs has been "anti-virus" software. This software relies on obtaining signatures or patterns of the malicious programs so they can be blocked from infecting a machine or removed from one. This had led to a cat-and-mouse game between the anti-virus vendors and the malicious program writers. The growth of the Internet has interconnected millions of computing devices on a large shared network. This has led to the adaptation by malicious program writers to move from the traditional virus model of attaching to an executable or disk file the user executes or opens to full self-replicating computer worms. These programs start by infecting a machine on a network and then spread themselves automatically to other machines through unpatched software on the other machines. The success of these worms comes primarily from the fact that the larger user base of the Internet has users who do not regularly patch their computer software. This has allowed malicious persons to gain control of tens of thousands of computing devices to execute code to steal data or attack other networks at their desire (Schneier, 2007). In some cases the victim of the infection doesn't even know their machine has been taken over or is susceptible to being taken over.

Anti-virus companies, operating system companies, and IT organizations have attempted to keep up with the growing threat, but their efforts can still not fully prevent infection on every machine. In some cases the malicious program even has the ability to disable anti-virus and software update programs so as to retain control of the infected machine (Leung, 2009). Some newer emergent technologies are being actively developed and slowly deployed that, in some cases, can detect a machine that has been infected so as to disconnect it from the network to reduce infection of other machines, but these have not gained wide popularity nor been thoroughly evaluated for effectiveness (Pauli, 2010).

Yet another method that has been proposed multiple times before but not accepted by the security community at large as plausible is to write a computer worm that is self-replicating but instead of infecting and taking control of machines it would patch them to protect them instead. This type of program is known as a "benevolent worm" (Schneier, 2003). The major reason until now that the industry and academic community as a whole has rejected this idea is that computer programs by human nature are likely to have coding errors or bugs which can cause unexpected behavior (Schneier, 2003). The risk of a benevolent worm causing more damage when it was intended to repair a vulnerable system was considered not worth the benefit of the protection. Major questions such as what if it unknowingly brought down a major power-grid system or caused a critical hospital system to crash have been asked. The most recently opinion, however, is that the large number of infected systems under the control of persons with unknown or malicious intent is more dangerous now than these possible risks from a benevolent worm (Schneier, 2011). Also, the possibility of critical systems such as power plant control systems being vulnerable in the first place and going unnoticed by the operations personnel for that system poses more of a risk to being susceptible to unknown and malicious persons versus an out of control benevolent worm. This paper will address the risks of benevolent worms for use as a way to combat malicious software and patch at-risk systems.

## Related Work

TODO

## Research Questions

The purpose of this research is to evaluate the characteristics of computer worms, both benevolent and malicious, and access those qualities that would be required to develop a successful benevolent worm. Some questions that must be addressed are:

* What are the possible risks with releasing a benevolent worm into the wild?
* What are the benefits?
* Can a benevolent worm be developed with sufficient quality such that the risks presented by such a program be mitigated or heavily out-weigh the risks of having infected systems on the network?

## Research Methodology

An analysis of past computer worms (both malicious and benevolent) provides the characteristics that a benevolent worm should emulate or avoid. Since many malicious computer worms do not have source code readily available much of the research is based on analysis by other computer security researchers and companies of binaries or running instances seen in the wild. The following questions are addressed for each worm:

1. What was the worm’s purpose (benevolent or malicious)?
2. What methods of propagation were used?
   1. Human empathies exploited?
   2. Automatic methods?
3. Did the worm achieve its purpose?
   1. What worked correctly?
   2. What went wrong?
   3. What could be corrected?

The second stage of research was to develop small programs that function as benevolent worms. These are targeted at specific vulnerabilities using specific exploit vectors versus one large program that can exploit with many different vectors. An isolated environment of virtual machines was used to do the testing on a network. Data on how many machines were successful infected, patched, and working afterwards was recorded. Each test had machines with various levels of patches applied already as well as some machines left untouched as a control group. In addition one test had some machines with specific OS corruption that allowed the OS to continue to run and be usable, but represents possible unexpected issues when the worm attempts to infect and patch it. This testing allowed for the collection of information on how a benevolent worm could perform under controlled expected conditions as well as random unexpected conditions of the systems it was attempting to patch.

The final stage of research involved the study of the ethical and legal issues with releasing computer worms. Proposals for methods of release are made based on the previous stages and this study.

# Works Cited

FoxNews. “Conficker Worm Hits University of Utah Computers.” FoxNews.com – SciTech. 12 Apr. 2009. 7 Sep. 2011. <http://www.foxnews.com/story/0,2933,514681,00.html>

Leung, Ka Chun and Sean Kiernan. “W32.Downadup.C.” Threats and Risks. 6 Apr. 2009. Symantec. 7 Sep. 2011. <http://www.symantec.com/security\_response/writeup.jsp?docid=2009-030614-5852-99>

Pauli, Darren. “Govt stays ISP zombie laws.” Security. 26 Nov. 2010. ZDNet. 7 Sep. 2011. <http://www.zdnet.com.au/govt-stays-isp-zombie-laws-339307564.htm>

Schneier, Bruce. “Benevolent Worms.” Crypto-Gram Newsletter. 15 Sep. 2003. 26 Aug. 2011. <http://www.schneier.com/crypto-gram-0309.html>

Schneier, Bruce. “The Storm Worm.” Schneier on Security. 4 Oct. 2007. 26 Aug. 2011. <http://www.schneier.com/blog/archives/2007/10/the\_storm\_worm.html>

Schneier, Bruce. “Hijacking the Core flood Botnet.” Schneier on Security. 2 May. 2011. 26 Aug. 2011. <http://www.schneier.com/blog/archives/2011/05/hijacking\_the\_c.html>

von Neumann, John and Arthur W. Burks, ed. Theory of Self-Reproducing Automata. Urbana: University of Illinois Press, 1966.