



# **Xeno-ttacks: Near-Future Attack Mutation**

**By Andrew Berkuta**

## Table of Contents

Introduction	3
Examples	4
Prevention Techniques	4
Conclusion	5

# Xeno-ttacks: Near-Future Attack Mutation

- 1. Xeno-ttacks focus on the lowest common denominator in which they can be carried and activated. This could be the next milestone in the evolution of malware.**
- 2. Xeno-ttacks would seek out lowest common denominators in four areas: operating system, program, service, and platform. One of these areas is already being exploited; however, the platform is the most dangerous.**
- 3. Systems not targeted could still become entry points by passing the benign code, which will activate when it lands on the desired targets. Countermeasures must either isolate the threat or prevent it outright.**

According to Gartner researchers and other industry experts, we are starting to see a shift in delivery methods and techniques for malicious code, which will “cross walk” through underlying control and operating systems. A Xeno-threat would seek out the lowest common denominator in various entry points and use it as a transport mechanism. The method it follows differentiates itself from traditional methods employed today. In the Greek language, “Xenos” means “foreign.” In the medical terminology, the definition is “strange or foreign matter.” For the purpose of this paper, we will extract the concept from the medical use to define Xeno-ttacks.

## Introduction

In traditional malware design, a predetermined base is selected. The malicious coder would examine the base and identify vulnerabilities, and then use it to design code to exploit them. An example of this would be the selection of a widely used base, such as a Microsoft® operating system version, and exploring the system for a published or unpublished vulnerability. The optimal vulnerability is one that is not patched yet.

Generically, we will define a base to be an operating system, program, service, or a platform. A vulnerability is defined as a weakness in a process, administration, procedure, or technology that can be exploited. There are numerous

vulnerabilities in programming code, which can be exploited by the traditional methods. However, malicious programmers are not just relying on these methods. The malicious programming communities are at the cusp of creating a new threat generation—the Xeno-ttack.

For review, traditional attempts at breaches and exploitation have used known ports to channel into a system. These ports are published as ones used by a developer to execute back-channel operations and communications. Another way malicious programmers exploit vulnerabilities is to exploit the buffer-overflow conditions found by reverse-engineering the software. The buffer-overflow condition is created when the developer does not account for a program’s memory area growing larger than the allotted space that the developer defined. Exploitation code, after some trial and error by the malicious writer, is designed to identify the buffer, or memory location, and “over stuff” the memory location with more data than it can hold. If the developer did not account for this condition and design safeguards against it, the memory buffer is overrun and the data spills over to an adjacent memory location. If researched correctly by the malicious programmer, a sequence of commands can be placed in this overflow location. The execution of them occurs with the same privilege level of the service or user that owned the overflowed memory location. The Xeno-ttack condition is not yet fully developed; however, there are undertones in the hacking communities of development in this area.

The Xeno terminology is used in the virus and antigen studies within the medical communities. In the medical field, Xeno-science starts with two hosts. The infected host (recipient) contains organisms, which are attacking the host. The second, clean host (donor) contains antibodies that may help eradicate the recipient’s organisms. This sounds standard so far—except Xeno-science would use dissimilar host samples (human versus pig, human versus fish, etc). The focus is to extract similar traits from each host that the offending organisms latch on to. The donor would be resistant to the organisms, and therefore be a prime candidate for providing antibody signatures to combat the organisms affecting the recipient. We use this theory when creating immune-booster medication and antibiotics. This medical example is the basis of our study within the computing field.

## Examples

In security, we have not yet seen cross-infective technology that is mature and widely used. For this type of attack to be successful, there would need to be exploitation at a level that transcends traditional base systems. Examples of this would be the following in donor and recipient format:

1. **Operating system Xeno-ttacks**—malware created on a Microsoft Windows® XP operating system (donor) that is portable and transmutable to a LINUX system running Windows emulation software (recipient).
2. **Program Xeno-ttacks**—Microsoft Internet Explorer web software (donor) containing infection code that transmutes on Firefox web software (recipient). In an orchestrated condition, multiple applications [donors] when run together, activate their respective “benign” Xeno-ttack code in shared space and affect the recipient program or host.
3. **Service Xeno-ttacks**—a LINUX spooler daemon (donor) containing infused code, which starts child services, such as RPC-type daemons and transmutes on to a Windows 2003 server (recipient).
4. **Platform Xeno-ttacks**—an Intel-based PC system (donor) containing overflow or instructions that are portable and transmutable to an Apple system (recipient).

If we examine the four examples above from a severity level, it is very apparent that a jump in malicious programming techniques would cause serious disruption and start the propagation of Xeno-type attacks. Based on severity, one that seems imminent is the fourth example, the platform Xeno-ttacks. A prime example is Apple Corporation. Apple has already announced that it is switching from the PowerPC chip set to the Intel chip set. Another distant but interesting possibility for attack would be from an Intel PC to a RIM BlackBerry using the new Intel processor announced recently. In either case, the operating systems are different; however, the underlying platform would potentially contain similar constructs and instructions where transmutation of code could exist.

A second possible example of “fertile ground” would be both the service and the operating system Xeno-ttacks, examples three and one, respectively. First, the service attack is possible as we further use cross-functionality with different operating systems. We have the ability to emulate stacks and buffers closely to the native system we are emulating. Certainly, if we view the original code as one to emulate, would we not “inherit” some possible unknown vulnerabilities as well? If so, then we could, under the pretense of compatibility, cause a duplication of the weakness in the original code. Why not then have the ability to design portable system calls within a service that

is posed as benign (ignored) on the donor, and once entered into the target recipient, then activates, spawns, or calls other services, and comes in under traditional protective barricades? From a malicious mindset, this is what is being discussed and explored.

The second case of operating system attacks contains elements that would exploit system calls and constructs (usually backward-compatibility ones are low-hanging fruits) that have the ability to read and execute code designed for other systems. A prime example is Microsoft Windows and the ability to run POSIX commands, or, with configuration, Apple services to reach out to Apple systems and provide resources to them. It is not a stretch of the imagination to determine that these pre-existing conduits are exploitable, given enough time and talent.

Program Xeno-ttacks are already here in their fundamental construct and therefore, have the lowest category on the severity level. Today, certain portability vehicles exist that can pass on through various hosts. Java, ActiveX, Unicode, common APIs, and applets are great examples in which portability is built into the design and an easier ability to walk across base levels can occur. In many respects, this is already happening on a controlled scale. The good news is that there is a security industry that caters to the prevention of these threats. What would make a program Xeno-ttack increase drastically in its severity level would be the orchestration of multiple programs working in concert via their snippets of code. In those instances, shared memory locations, data, and resources could be the catalyst in which the benign pieces on the donors are assembled and activated on the targeted recipient host or program. This threat would be difficult to protect against, and would be one of the more disruptive in nature.

## Prevention Techniques

As previously mentioned, there are programming conditions and certain buffer-overflow conditions that are preventing the programming example of Xeno-ttacks. For the others, it is too soon to see countermeasures. The permanent correction to prevent these Xeno-ttacks would be to abandon legacy compatibility and control cross-functionality within systems.

In the computing industry, it has been a long-standing belief that manufacturers and software should break away from backward compatibility to older systems. Intel, for example still produces its chips with compatibility to 386 instruction sets. This compatibility causes penalty to the newest chipsets by enabling the extended code base and translation. Furthermore, the memory locations are still compatible, which is one cause for buffer-overflow conditions and base memory leakage.

Microsoft is a large software manufacturer whose products still contain compatibility to the platform code, and Microsoft has itself put in operating-system compatibility threads and hooks for interoperability. A good example is the use of the Apple file system extensions, which are obscured from the purview of native Windows operation controls. Many supplemental countermeasures are available that will scan these areas, but nothing exists within the base operating system to indicate that someone has placed malicious code into these hidden file systems. Also, there is still the compatibility with POSIX in the operating systems. Under the interoperability banner, these hooks are placed to enable UNIX systems to hook into the NTFS system. In both Microsoft examples, a need for administrative controls for these would be in order. The most preferred would be the necessary and out-of-the-box disabled state of these “enhancements.”

Prevention for all of these types of conditions would force manufacturers to go back to the drawing board and redesign the systems. This proposal would be too cost prohibitive and would cause too much complexity in a redesigned system, requiring administration and operations to consciously think through enablement of services prior to or during installation of a system. It would also be more complex in management and repurposing the system when needing new features.

Another solution would be for manufacturers to abandon legacy compatibility in the next generation of product, and schedule obsolescence within their legacy product lines in favor of a better system. This, too, would potentially lose market share for the manufacturers from entities that have heavily developed systems and code under the old architecture. No manufacturer wants to lose market share by doing the right thing.

The reality will be that the brunt of the protection from these Xeno-ttacks will lie in the hands of security countermeasure manufacturers. For so long, the security industry has been answering the call of the computing industry to store up defenses and provide layers of protection for systems, services, processes, and technologies. The security industry will further develop the appropriate countermeasures when new attack methods are created. With the severity and quick proliferation of threats, the security industry has its hands full.

There are boundaries that have not been crossed yet and, which act as natural barriers to most scenarios happening today. Efforts on the part of the malware design communities are actively working on these concepts and are gaining additional intelligence to apply to their trade. One such barrier is user intervention. In many cases, user intervention is necessary to apply or authorize the execution of the malware in this context. Strong efforts are being put forth to automate and remove the user from the equation to run this transmutable code.

A second barrier is the efforts of the security industry to rebuild traditional countermeasures to incorporate technologies that observe and protect against complementary vectors of attack. An example would be the traditional anti-virus, which has been rebuilt to include buffer-overflow protection and firewall countermeasures. Host-based intrusion detection has been rebuilt to actively protect and combine firewall technology. Various other security technologies are redefining their single-purpose task and are following the “Four Cs of Security,” consolidation, collaboration, correlation, and compliance. The only solid example of this trend in the security industry currently is McAfee,® Inc. Throughout its products, McAfee has continually infused these cross-pollination techniques to ensure survivability within systems and the enterprise.

## Conclusion

Although not fully prevalent, it is predicted that within a two- to three-year period, Xeno-type attacks (“Xeno-ttacks”) will be built in sufficient number and will overcome the barricades with development techniques that are being explored today. The security industry is always seeking these techniques to provide valid countermeasures against the threats, and is always monitoring the venues that are being used as code-exchange and discussion forums.