**Week3: Attack Prevention Article**

Nate Bachmeier  
CMGT/441: Info Sec Security Management  
June 2nd, 2014  
University of Phoenix  
Facilitator: Alexander Pons

# Week 3: Attack Prevention Article

There are many attack prevention solutions available such as security through obscurity, network isolation, and least privilege accounts. While these mitigations provide some benefit they are often subverted through chaining a series of exploits across the network.

For instance an attacker might first gain access to a web server, from there they have access to the database server. With each hop the access is expanded at an increasing exponential rate (Bluehat 2013). In many complex enterprises it takes less at most 5 hop to gain access to hundreds or thousands of machines.

This raises the question, how do we secure and protect our environments? Throwing more defense in depth is helpful but is addressing the symptom and not the actual problem. To address the root of the issue, we need to look at how attackers are getting in.

According to the Security Information Report (SIR), one of the most frequently used front doors is by exploiting a vulnerability in a software application. If this is their plan of attack then one should fortify the requirements of their exploits working in the first place. By mitigating that risk it is less likely they will gain the initial foothold and penetrate further into the network.

Across the Microsoft ecosystem the “Enhanced Mitigation Experience Toolkit” can be deployed and configured on all nodes. Once deployed additional enforcements can be made to the running applications.

For example a common attack against Address Space Layer Randomization and Data Execution Protection (ASLR/DEP) enabled applications to find a binary that does not have ASLR enabled (Microsoft, 2013). This gives the attacker fixed memory addresses and other constants that can be embedded in the attack. With EMET it is possible to force all binaries to always be in randomized locations and protected from overwrite.

Another common vector is to use exception handler chaining. How this works is the attacker gets a payload inside of the process, where it will live at some unknown location then cause execution of the payload to begin. When the payload starts it sets up a try/catch block then raises an exception. The catch block (attacker code) will begin executing with the exception object. The exception object will contain references to various other objects that can be used to find other interesting memory addresses (Anley, C; Felix, J; Richarte, L, 2007).

With the EMET deployed the Win32 subsystem will perform additional checks before invoking the exception handler. Once such check is that the catch block was defined in a loaded binary not just an arbitrary memory location. This can subvert the attack from working in the first place.

Another protection offered is pre-allocating the NULL page, which provides protections against many kernel exploits. How this vector works is an attacker identifies a kernel mode driver which dereferences a NULL function pointer. The attacker then fills the first few pages with the address of their exploit code, then triggers the bug. Instead of crashing the function will be followed and the attacker will gain system level permission. EMET blocks this attack from being successful.

# Conclusions

The Enhanced Migration Experience Toolkit, provides a great protection strategy by removing the attacker’s ability to gain the initial foot hold into the network. By stopping the initial breach it provides a great protection against many mainstream attacks from success.

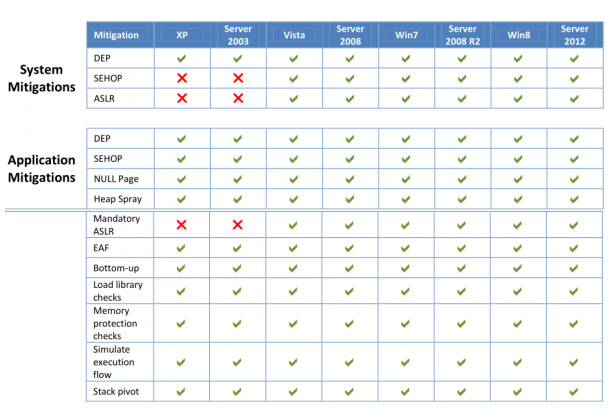


Figure : Protections Offered (Kreb On Security, 2013)

# References

Anley, C; Felix, J; Richarte, L. (2007). *The Shellcoder's Handbook.*

Bluehat 2013. (n.d.).

Kreb On Security. (2013, June). *Windows Security 101 EMET 4.0*. Retrieved from Kreb On Security: http://krebsonsecurity.com/2013/06/windows-security-101-emet-4-0/

Microsoft. (2013). *Security Intelligence Report Volumen 16.* Microsoft Corp.