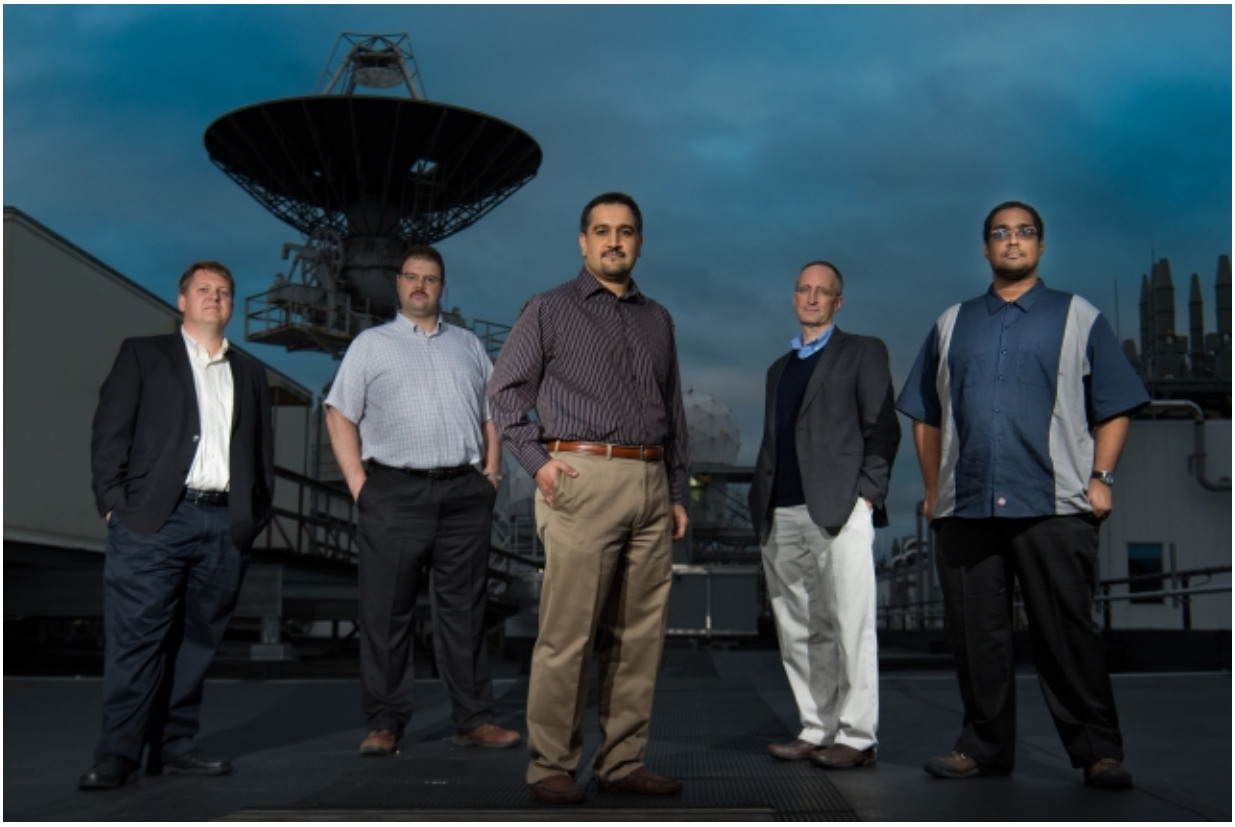
A unique moving target technique combats information leakage attacks

December 12 2016, by Dorothy Ryan



At each output, the Timely Address Space Randomization (TASR) software rerandomizes code so that cyberattackers are constantly faced with a moving target on which to utilize the "leaked" memory information they acquired via a network intrusion. The rerandomization does not affect the computer's memory storage (stack and heap). Credit: Massachusetts Institute of Technology

When it comes to protecting data from cyberattacks, information technology (IT) specialists who defend computer networks face attackers armed with some advantages. For one, while attackers need only find one vulnerability in a system to gain network access and disrupt, corrupt, or steal data, the IT personnel must constantly guard against and work to mitigate varied and myriad network intrusion attempts.

The homogeneity and uniformity of software applications have traditionally created another advantage for cyber attackers. "Attackers can develop a single exploit against a software application and use it to compromise millions of instances of that application because all instances look alike internally," says Hamed Okhravi, a senior staff member in the Cyber Security and Information Sciences Division at MIT Lincoln Laboratory. To counter this problem, cybersecurity practitioners have implemented randomization techniques in operating systems. These techniques, notably address space layout randomization (ASLR), diversify the memory locations used by each instance of the application at the point at which the application is loaded into memory.

In response to randomization approaches like ASLR, attackers developed information leakage attacks, also called memory disclosure attacks. Through these software assaults, attackers can make the application disclose how its internals have been randomized while the application is running. Attackers then adjust their exploits to the application's randomization and successfully hijack control of vulnerable programs. "The power of such attacks has ensured their prevalence in many modern exploit campaigns, including those network infiltrations in which an attacker remains undetected and continues to steal data in the network for a long time," explains Okhravi, who adds that methods for bypassing ASLR, which is currently deployed in most modern operating systems, and similar defenses can be readily found on the Internet.

Okhravi and colleagues David Bigelow, Robert Rudd, James Landry, and

William Streilein, and former staff member Thomas Hobson, have developed a unique randomization technique, timely address space randomization (TASR), to counter information leakage attacks that may thwart ASLR protections. "TASR is the first technology that mitigates an attacker's ability to leverage information leakage against ASLR, irrespective of the mechanism used to leak information," says Rudd.

To disallow an information leakage attack, TASR immediately rerandomizes the memory's layout every time it observes an application processing an output and input pair. "Information may leak to the attacker on any given program output without anybody being able to detect it, but TASR ensures that the memory layout is rerandomized before the attacker has an opportunity to act on that stolen information, and hence denies them the opportunity to use it to bypass operating system defenses," says Bigelow. Because TASR's rerandomization is based upon application activity and not upon a set timing (say every so many minutes), an attacker cannot anticipate the interval during which the leaked information might be used to send an exploit to the application before randomization recurs.

When TASR determines that the rerandomization must be performed, it pauses the running application, injects a randomizer component that performs the actual rewriting of code, then deletes the randomizer component from the application's memory, and resumes the application. This process protects the randomizer from infiltration. To change the memory layout of a running application without causing a crash, TASR updates all memory addresses stored in the application during rerandomization.

TASR has several advantages over other randomization techniques. It protects against all existing types of information leaks for memory corruption attacks, regardless of the specific method of attack (e.g., viruses, email phishing, access via the Internet) or type of vulnerability

(e.g., logic flaws, race conditions, buffer overflows). TASR is flexible: it is compatible with full standard C language, does not require additional hardware, and is backward-compatible with legacy systems. Finally, performance evaluations carried out by the research team showed that the fully automated TASR technique incurs a low execution overhead of only about 2.1 percent.

The research team has published details on the development of the TASR technique and their assessments of its effectiveness in the Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. They are now pursuing opportunities to transition the technology into operational systems and have already taken some of the foundational concepts behind TASR and applied them to additional prototypes. They also plan to release portions of the TASR system under an open-source license. Researchers seeking more information on this technology may contact Okhravi.

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology