

Cyberwar Surprise Attacks Get a Mathematical Treatment

By [John Bohannon](#) Jan. 13, 2014 , 4:00 PM

```
Last login: Mon Jan 13 14:42:40 on ttys001
Johns-big-mac:~ john$ cd /Users/john/Desktop
Johns-big-mac:Desktop john$ cat stux*
#include "define.h"
#include "8. AssemblyBlock1.h"
#include "9. AssemblyBlock2.h"

INT32 BLOCK4_InjectAndExecuteVirus(PASM_CODE_BLOCKS_HEADER sAS
INT32 BLOCK4_ExecuteLibrary(PASM_CODE_BLOCKS_HEADER sASMCodB1
void BLOCK4_CopyPEHeaderInfo(PGENERAL_INFO_BLOCK sInfoBlock, f
NTSTATUS BLOCK4_AlignAddresses(PIMAGE_DOS_HEADER *pImageDOS);
void BLOCK4_memcpy(void *pDestination, const void *pSource, ur
void BLOCK4_CopyDataIntoMapView(PVOID pVirusModule, PIMAGE_NT
INT32 BLOCK4_InjectCodeIntoNTDLL(PASM_CODE_BLOCKS_HEADER *sASMC
INT32 BLOCK4_LoadVirusModuleInfo(PHARDWARE_ADDRESSSES nHardAdr
```

Surprise. New mathematical model could help predict the launch of cyberattacks, such as the use of the Stuxnet computer worm (code above) to disable Iranian centrifuges.

John Bohannon

Have you been missing the grim mathematical war games that strategists once used to map out possible nuclear confrontations? Don't worry, the games are back—this time applied to computer security. Researchers have now mathematically formalized the strategy of computer hacking, potentially enabling anyone—governments, activist hackers, cybermafia—to determine the optimal timing of attacks.

It's tricky to decide exactly when to try to ruin someone's computers, sabotage the equipment controlled by their computers, or even just harvest their data. For example, if your victim will discover and delete your malware

soon after they detect it, then you should wait patiently for the most damaging opportunity. But if the chances are increasing over time that the victim will patch a vulnerability before you can exploit it, then you'd better hurry. Then again, some malware is so stealthy that you can use it multiple times, so the question is not when, but how often, to use it. On top of all of this, your optimal strategy depends on not only the value of what you stand to gain but also the costs of failing or being caught. The stakes are very different if you are a criminal trying to steal bank passwords from a personal computer, for example, or a spy trying to steal military technology from a foreign country.

So how do you balance all of these parameters to find the best strategy?

Robert Axelrod to the rescue. The University of Michigan, Ann Arbor, political scientist is best known for his work on the [prisoner's dilemma](#), a problem in game theory that has influenced everything from economics to evolutionary biology. Axelrod says he became interested in cyberwar a few years ago when he realized that it involved a similar problem that he first described in a 1979 paper called "[The Rational Timing of Surprise](#)." The core idea is that the element of surprise is itself a strategic resource, and modeling its costs and payoffs can lead you to take counterintuitive actions. A classic example is the British decision in World War II to allow German spies to continue gathering damaging intelligence for Hitler years after the spies' identities were discovered. This allowed the British to wait until the stakes were at their absolute highest: [D-Day](#). By feeding the German spies false information about where the invasion would land, the British sabotaged the German defense.

Hacking a computer network is not so different from storming an enemy beach. In both cases, the optimal time to attack depends on how the risks, costs, and benefits change over time, the target's vulnerabilities, and the element of surprise. So Axelrod teamed up with Rumen Iliev, a psychologist working down the hall from him, to rejigger his 1979 model for cyberattacks.

Then they applied it to several recent examples, such as the [Stuxnet](#) attack, in which an unprecedentedly complex computer worm—allegedly created by the U.S. and Israeli governments—was used to sabotage Iran’s nuclear centrifuges.

The timing of Stuxnet and many other known recent cyberattacks appears to be close to optimal, Axelrod and Iliev report in a [paper published online today in the *Proceedings of the National Academy of Sciences*](#). While the hackers behind them have so far probably relied on intuition, they note that their model could help in the design of future attacks. “The results, however, are equally relevant to a defender who wants to estimate how high the stakes have to be in order for the offense to exploit an unknown vulnerability,” they add.

The model can also be used to help forecast potential patterns in future attacks, they add. For example, there is a large and growing market—both legal and illegal—for information on computer system vulnerabilities known as “zero-day exploits.” As this market grows, the timing of cyberattacks is likely to become more rapid, because potential victims can also buy the same information, helping them strengthen defenses. That will put pressure on hackers to attack sooner rather than wait. (Axelrod says that the U.S. government is the largest buyer in this market.)

“The work provides a solid logical foundation for fresh thinking in the cyber security field,” says John Arquilla, a defense analyst at the Naval Postgraduate School in Monterey, California. He points out that it can lead to some seemingly “perverse” but insightful policy recommendations. For example, cyberdefenders may not want to work too hard at closing every possible security vulnerability as soon as it is identified, because “doing so might unintentionally encourage adversaries to launch attacks, even for relatively low stakes, because their exploits soon won’t be usable at all.” Suffering less frequent attacks on known vulnerabilities may be better than a constant barrage.

“That said, my own view is that ‘risk’ is a multi-dimensional factor that requires further and deeper exploration,” Arquilla says. If all actors blindly pursue their own interests using the model’s logic, the world could become a more dangerous place for everyone, he says. The Stuxnet worm, for instance, was supposed to quietly delete itself after doing its harm, but it was unintentionally released “into the wild, where it is no doubt being tweaked, reverse-engineered, and readied for fresh exploits by others.”