Three Generations of Automated Cyber Attacks

Edward G. Amoroso, TAG Cyber
Reena Choudhry, Shape Security

Version 2.0

May 10, 2019

Abstract
First-generation automated attacks were exemplified by worm programs that self-propagate across networks. Second-generation automated attacks were exemplified by botnets that exploit distribution to create amplified threats. Third-generation automated attacks are now introducing imitation methods to emulate normal, authorized user behavior. Practical cyber security recommendations are offered to reduce the risk of imitation attacks.

Introduction
Most popular views of cyber offense involve an evil hacker – a human being – carefully planning a targeted attack on some unsuspecting victim. The motivations for such malicious action have always varied, but they range from the foolish decisions of youth, to the truly nefarious goals of a nation-state adversary. In all cases, however, this familiar view of cyber offense involves some human actor tangled into the middle of the step-by-step attack plan.

Botnets changed this view for many by introducing automated command and control (C&C). Early Internet Relay chat (IRC) bots emerged in the late 1990s, generally with the goal of causing disruption to a targeted site via a distributed denial of service (DDOS) attack (see [1], for example). Each individual bot would be programmed to take direction from a command node, and would transfer control to a different node if necessary.

With the advent of botnets came recognition that automation might play a significant role in the strategy and tactics involved in cyber offense. While this might have seemed a casual observation, it was anything but – simply because automation enables integration of machine learning, contextual processing, brute force cryptanalysis, and other advanced tactics to the offensive arsenal – even if the controlling human has little skill.

This note explains and illustrates the progression of attack automation in the context of three generations. The first generation, starting in the 1980's, involved *worms* automating the propagation task; the second generation, starting in the 1990's, involved *botnets* automating the attack orchestration task; and the third generation, starting more recently, involves attacks that can effectively *imitate* human actors.

As background to the development of this note, the authors interviewed several cyber security experts from various industrial sectors to solicit their opinion on the relative intensity and consequence of these three generations of attack automation. Their views are sprinkled throughout thus document – albeit with the understanding that their identities remain anonymous. This allowed for more generous sharing of examples and opinions.

First Generation Automated Attack – Worm

We begin our progression through the evolution of automated cyber attacks with a program called a *worm*. The defining characteristic of a worm, in contrast to other forms of malware, is its ability to self-propagation automatically. That is, a worm can cascade based solely on its programmed execution, without need for the type of human interaction one finds with phishing and other software virus methods.

A worm includes three tasks that accomplish the bare bones aspect of its self-propagation. These three tasks become the functional requirements for the skeleton of any worm program, and provide a means for adding other more nefarious attack objectives such as escalating privilege, finding and collecting data, and the like. The three steps of the worm schema are as follows:

Worm:

1. *find_host (h)* *-- finds another system over a network*
2. *copy (worm, h)* *-- copies the worm program to other system*
3. *exec (worm, h)* *-- executes the worm on that system*

An execution trace of the worm from one machine to another shows each line producing some progress in the self-propagation. The first line involves finding some target system on a network; the second line involves copying the worm to that found system; and the third line involves execution of the worm on that computer. This, in turn, starts the process over with that found computer locating another computer, and so on.

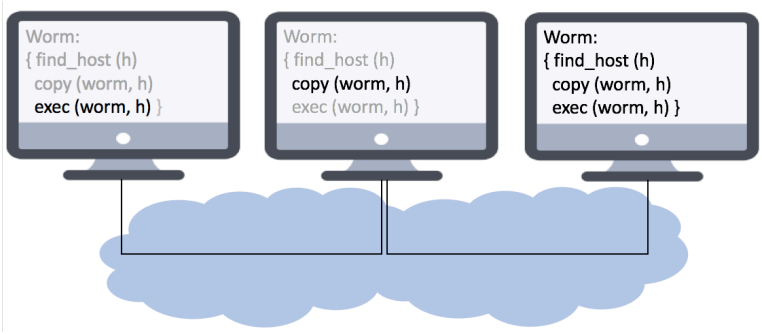

Figure 1. Cascading Self-Propagation of an Automated Worm Attack

Cyber security solutions to worms have been uneven, but include better patching to reduce the attack surface for connection, copying, and execution based on exploitable vulnerabilities. Well-

configured firewalls and network segmentation are also useful means for either stopping or at least slowing down the progression of a worm, especially one that uses a TCP or UDP port that can be temporarily or even permanently disabled.

In 2003, the cyber security community saw a frightening progression of auto-propagating worms such as Slammer, Blaster and Nachi [2, 3, 4]. Surprisingly, in the ensuing years, the frequency and intensity of worm attacks tended to decrease for two reasons: First, the patching programs for most enterprise teams improved considerably; and second, malicious actors began to increase their use of superior attack constructs called *botnets*.

Second Generation Automated Attacks – Botnet

A *botnet* is built from infected, hijacked endpoint systems, usually PCs, but increasingly anything with a CPU. These endpoints, which can be numerous and geographically scattered, are controlled by one or more hijacked systems referred to as the command and control (C&C). Botnets also include servers to store software, a human being called an operator who manages the construct, and a target victim located somewhere on the Internet.

As depicted in Figure 2, the arrangement of a botnet includes bots potentially scattered across the globe, or clumped together in a region – often based on a regionally-interesting lure, such as a phishing email, used to infect these systems. Botnet C&C systems are also generally distributed, as are the botnet software drop locations. The result is a robust, secure distributed system that can deliver lethal volume-based attacks at targets.

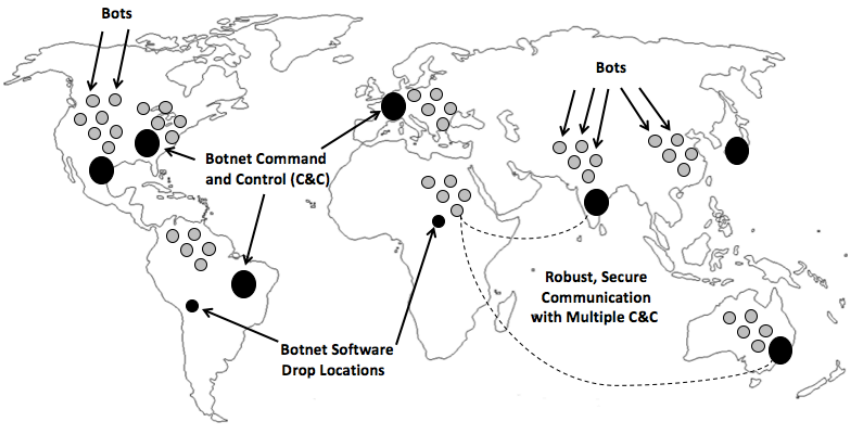

Figure 2. Typical Botnet Geographic Distribution

The power of a botnet comes from four design elements: *Distribution*, *resilience*, *reflection*, and *amplification*. Through distribution, a botnet can access a wide assortment of resources to hijack; through resilience, including clever DNS controls, the botnet can pass-the-baton of C&C to make it hard to take down; through reflection, botnets can obfuscate the sources of attack; and through amplification, botnets can increase their volume and intensity.

In practice, botnets have enabled *distributed denial of service (DDOS)* attacks against targeted websites. Attack volumes were originally manageable in the early 2000's, often in the Mbps and single-digit Gbps range. More recently, however, the occurrence of DDOS attacks approaching 1 Tbps has increased, and might soon become quite common. Such increased size represents a serious unsolved availability risk for network operators.

In addition to DDOS attacks, botnets can also create fake populations of live, human, end-users. For example, botnets are often used to create click-bot attacks, where a crowd generates synthetic interactions with on-line advertising, marketing, or political campaigns. This type of automation continues to be a nagging issue – and it leads to the present generation of more serious automated attacks: *Imitation*.

Third Generation Automated Attack – Imitation

Imitation involves bad actors mimicking the behavior of authorized entities – usually human beings – to accomplish a malicious objective. At small scale, this involves an intruder imitating the predictable actions of a targeted human, such as the owner of a mobile device. In practice, however, most attacks targeting the identity or credentials of a single individual have tended to use PC malware, identity theft, and other traditional offensive methods.

At large scale, however, imitation has become an effective weapon for achieving a variety of broad attack objectives, including account take-overs for on-line banking, Internet retail, consumer credit, and other eCommerce settings. Imitation strategies to attack these types of applications employ simulation of group behavior, often by creating populations of fake users with believable characteristics.

Simulating customer behavior to an eCommerce vendor is accomplished by various attack methods including the following[1]:

- *Account Creation Fraud* – This involves using stolen information about individuals to create fake on-line accounts. When organizations are breached and thus expose the credentials of their customers, employees, or citizens, the primary consequence is the large-scale enablement of account fraud. A common scenario involves use of exposed personally identifiable information (PII) to hack credit or gift card applications.
- *Account Takeover Fraud* – This involves the use of stolen information to hijack real, existing on-line accounts. Traditionally, account takeovers have been done manually, which limited their scope. But with the introduction of automated methods such as credential stuffing, this attack can now scale across large account bases, thus creating significant problems for eCommerce services.
- *Data Harvesting* – This involves automated tools collecting data from Internet-visible sites using a variety of techniques ranging from traditional screen scraping to more sophisticated imitation of authorized requests from potential customers or other interested parties.

[1] Information was adapted here from materials on the Shape Security website (https://www.shapesecurity.com/).

The root cause behind all imitation attacks is that modern automation from unauthorized bad actors has improved to the point where they can accurately mimic authorized good actors – regardless of whether they are humans, devices, or other endpoint systems. Such mimicking is observed by human bad actors and programmed into the imitation user's functionality. At some point in the future, machine learning might be used to translate normal behavior to fake users.

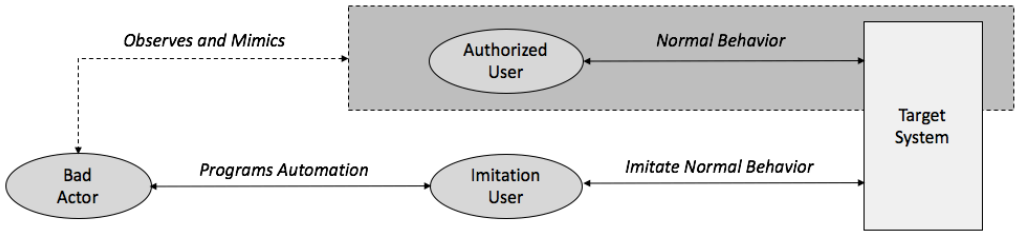

Figure 3. Automation as Root Cause for Imitation Attacks

Any type of entity, in any type of environment, can be vulnerable to imitation attacks. Large-scale industrial control systems (ICS), for example, might rely on human actions as part of a maintenance or other operational procedure. If a bad actor uses automation to mimic authorized human actions such as approving some potentially dangerous operation such as power transformation, then bad consequences might result.

Recommendations for Mitigating Automated Attacks

Conventional cyber attacks are mitigated by a combination of security policies, administrative procedures, and functional measures. Reducing the risk of automated attacks is no different – and certainly good security policies, proper user and administrative procedures, and a modern cyber security functional environment are pre-requisites to solving the problem. Every working security solution starts with attention to the basics.

That said, to achieve meaningful reduction of cyber risk in any enterprise, eCommerce, or infrastructure setting from imitation attacks requires functional controls that can match the speed, flexibility, and scale of any automated threat campaign. Stated simply: One needs automated defense to stop automated offense. The functional requirements for the detection, mitigation, and response to imitation attacks are as follows:

Real-Time Visibility – Application requests from potentially fraudulent imitation sources must be visible as they occur. This implies the need for real-time detection of application accesses either in-line or via some other means for observation. Generally, such imitation detection would be invisible to both authorized and fraudulent users. In cases where access is encrypted, visibility will be confined to clear text metadata.

Advanced Data Analytics – Information collected about application accesses must be analyzed quickly to determine if an imitation attack is underway. This might require advanced machine

learning based on accurate threat intelligence. Such capability will generally require coordination with cloud-resident resources. Accuracy of analytics improves with the amount of historical data available about both normal and imitation attack behavior.

Imitation Detection Algorithms – Determination of what exactly constitutes an imitation attack is obviously the greatest technical challenge in addressing this threat. Algorithmic techniques will typically take the following into account:

- *Correlation* – Many different factors will be correlated based on available signatures to determine whether an automated attack is underway (as in traditional Turing algorithms). This can include measurements of timing, syntax, semantics, and other attributes of application accesses by users.
- *Machine Learning* – Machine learning can be used with training data to support auto-learning. This can be done off-line or live (as with deep learning) to categorize examples of what is imitation and what is normal. Machine learning accuracy improves with additional training data.
- *Behavioral Analytics* – Behavioral analytics can be performed to review patterns of activity based on heuristics about what constitutes real versus imitation behavior. This can complement signature and machine learning methods.

Automatic Mitigation – The recommended decision to block, redirect, or notify operators about an imitation attack must be done in real-time, based on the collected data and supporting analysis. Whether the actual mitigation is done automatically is a local decision, but reliance on manual control of mitigation will greatly increase risk during automated imitation attacks.

References

[1] https://en.wikipedia.org/wiki/Sub7
[2] http://malware.wikia.com/wiki/Slammer
[3] https://www.techopedia.com/definition/27295/blaster-worm
[4] https://www.upi.com/Nachi-worm-tries-to-fix-computers/13781061322503/