2013 5th International Conference on Cyber Conflict
K. Podins, J. Stinissen, M. Maybaum (Eds.)


# A Control Measure Framework to Limit Collateral Damage and Propagation of Cyber Weapons

**David Raymond**
United States Military Academy
West Point, New York, USA

**Gregory Conti**
United States Military Academy
West Point, New York, USA

**Tom Cross**
Lancope Inc.
Alpharetta, Georgia, USA

**Robert Fanelli**
United States Cyber Command
Fort Meade, Maryland, USA

**Abstract:** With the recognition of cyberspace as a warfighting domain by the U.S. Department of Defense, we anticipate increased use of malicious software as weapons during hostilities between nation-states. Such conflict could occur solely on computer networks, but increasingly will be used in conjunction with traditional kinetic attack, or even to eliminate the need for kinetic attack. In either context, precise targeting and effective limiting of collateral damage from cyber weaponry are desired goals of any nation seeking to comply with the law of war. Since at least the Morris Worm, malicious software found in the wild has frequently contained mechanisms to target effectively, limit propagation, allow self-destruction, and minimize consumption of host resources to prevent detection and damage. This paper surveys major variants of malicious software from 1982 to present and synthesizes the control measures they contain that might limit collateral damage in future cyber weapons. As part of this work, we provide a framework for critical analysis of such measures. Our results indicate that a compelling framework for critical analysis emerges by studying these measures allowing classification of new forms of malware and providing insight into future novel technical mechanisms for limiting collateral damage.

**Keywords:** *cyber operations, malware controls, collateral damage, law of armed conflict*

The views expressed in this article are those of the authors and do not reflect the official policy or position of West Point, the Department of the Army, US Cyber Command, or the United States Government.

# 1. INTRODUCTION

As the world becomes more reliant on computers and networks, it is only natural that they will become targets during geopolitical conflict. Computer networks underlie our public and private utility infrastructures, banking and financial systems, and military command and control systems, all potentially lucrative targets. Targeting these networks requires an offensive cybersecurity capability and the addition of cyberspace components to the U.S. and other countries' military organizations highlights the potential for future conflict in the cyber domain.

Centuries of armed conflict have informed an ethical and legal framework for warfare, which includes a responsibility to limit collateral damage. Current Law of Armed Conflict (LOAC) addresses traditional armed conflict, but is not well-defined in the cyber domain. The complex interaction and highly interconnected nature of systems in cyber and physical space make the application of these laws more challenging. However, this ambiguity cannot be an excuse to act without regard to ethical considerations in cyberspace. Offensive cyber weapons created for use in interstate conflict could cause serious collateral damage to physical and informational assets. For example, malware designed to shut down industrial control systems in an adversary's munitions manufacturing facility might accidentally shut down a hospital's power control system. We must carefully follow the LOAC in the development of our cyber weapons if we are to justify their use to ourselves and to the international community.

This research examines ways cyber weapons can be controlled to limit collateral damage. Even the earliest malicious software included controls to limit infection and restrict spread to certain systems. The first well-known computer worm, the Morris Worm, could only infect DEC VAX computers running specific operating systems and checked whether a machine was already infected to limit resource consumption [1]. Notably, these checks failed to function properly and the worm degraded much of the ARPANET.

This paper provides a framework of controls that cyber weapons developers can use to more carefully control their software and avoid unwanted collateral effects. The value of our framework is that it demonstrates how malware can be controlled to severely reduce the threat of collateral damage, and it provides a template against which malware can be evaluated to determine how well it conforms to the LOAC. The framework does not consider third-party control of malicious software released by other individuals or organizations, such as the FBI's response to the DNSChanger malware [2]. Furthermore, the framework is of little use to malicious actors who create malware without regard for ethical considerations.

Section 2 of this work provides background and discusses related malware research. Section 3 provides a representative sampling of malware that has employed control mechanisms to limit its spread and Section 4 proposes a framework for controlling malware to allow specific targeting while limiting collateral damage. Section 5 presents analysis and our conclusions.

# 2. BACKGROUND AND RELATED WORK

## A. BACKGROUND

In military terminology, *targeting* refers to the process of selecting appropriate capabilities to achieve a commander's desired effects. Capabilities can either be kinetic (bombs and bullets) or non-kinetic (leaflets and press releases). Non-kinetic capabilities are usually preferred because they minimize loss of life. For example, the Stuxnet virus seems to have been designed to sabotage centrifuges at Iran's Natanz nuclear enrichment facility [3]. A bombing campaign might have achieved the same purpose, but with potentially high casualties.

Military leaders have recently recognized the potential for cyber weapons to produce effects that meet the commander's intent, either in conjunction with or in lieu of kinetic operations. Furthermore, cyber weapons are difficult to attribute to a specific individual, organization, or nation-state, carry minimal risk to friendly and enemy forces, and limit collateral damage in the traditional sense.

The cyber attacks launched against Georgia during hostilities with Russia in August 2008 exemplify the potential of this new form of warfare. The Russian invasion of Georgia was preceded by cyber attacks consisting of website defacements and distributed denial-of-service (DDoS) attacks targeting government, news media, and financial websites [4]. These attacks limited the Georgian government's ability to coordinate a response to the Russians and prevented Georgia from getting their story to rest of the world. Whether these cyber attacks were coordinated by the Russian government or not, they were of benefit to Russia's subsequent invasion [5].

As cyber weapons become more attractive as a component of military action, legal and ethical questions arise with regard to the LOAC and its application to cyberspace. In this work, we are concerned primarily with *jus in bello*, or the ethics of conduct during warfare, and specifically the ethics concerning the use of cyber weapons and the potential for collateral damage [6]. The principle of *distinction* requires that non-combatants be avoided in attacks because they are not legitimate military targets. According to this tenet, military leaders should also avoid collateral effects on non-combatants. The principle of *proportionality*

dictates that the defense against an aggressor must be proportional to the attack. While completely avoiding collateral damage is not always possible, proportionality dictates that collateral effects be minimized [6].

Several countries have recently increased their capabilities to conduct cyber operations. U.S. Cyber Command was established in May 2010 and is "responsible for planning, coordinating, integrating, synchronizing, and directing activities to operate and defend the Department of Defense information networks and when directed, conducts full-spectrum military cyberspace operations." China began forming a cyber force as early as 1997, and in July 2010 announced the establishment of an 'Information Protection Base' within the People's Liberation Army (PLA) to defend their networks [7]. Russia and Iran have well-defined military objectives in cyberspace [8] [9]. These are just a few examples of world leaders formalizing their cyber security efforts and placing them at least partially under the control of their militaries. As cyber operations increasingly become the purview of military leaders and are used as a component of military operations, it is important that we define the boundaries of moral-ethical behavior for the deployment of cyber weapons. Some countries will choose not to employ control measures in their cyber weapons. Those countries that choose not to follow established standards of behavior with their cyber weapons should be treated by the international community like countries that ignore the LOAC in other areas.

## B. RELATED MALWARE RESEARCH

Cohen conducted extensive virus experiments starting in the 1980s, first coding them and then developing virus defenses. One of his earliest papers provided pseudocode for a generic virus that included one of the basic malware controls, checking to see whether a file was already infected before modifying it [10]. Cohen studied the potential for identifying malware on a system and proved that no single algorithm can positively detect all computer viruses. He also made a case for the benevolent use of computer viruses [10].

Some of the earliest published research on computer malware is in Ludwig's *Little Black Book of Computer Viruses* [11] and his follow-on, *Giant Black Book of Computer Viruses* [12]. These seminal books describe the development of self-replicating malware and discuss methods for hiding malicious code and avoiding antivirus software. Ludwig even envisioned the potential for military applications of malicious software back in 1990 [11], an idea that has only recently been acknowledged by government and military leaders.

Research by Fanelli explored a methodology for targeting and controlling collateral damage in cyber operations [13]. He argued that the LOAC mandates that countries

seek to avoid collateral damage in cyber operations and shows that, despite the complex nature of these operations, it is possible to affect specific targets while minimizing effects on non-target systems and organizations.

Importantly, much of the most significant related research comes from academic and industry analyses of each malware family and variant. We include key references later in the paper.

# 3. MALWARE CONTROL EXAMPLES

Some consider malware such as viruses and worms to be uncontrolled once released, however, from the earliest examples of malicious software, controls were used to limit propagation and restrict behavior. Recent malware examples use sophisticated controls that even seem to specifically target organizations or facilities. The most basic control measure, observed in 1987 with the discovery of the Stoned virus, is for malware to check to see whether the target system is already infected [14]. Stoned alters the master boot record (MBR) on floppy disks and moves the original MBR to another location on the disk. Once resident in memory, Stoned checks whether disks inserted into the computer are already infected and, if so, does not alter them. For Stoned, this pre-infection check prevents the original MBR from being overwritten. In other malware it might be done to prevent unnecessary resource consumption or for other reasons. Control measures have grown in sophistication and the evolution of controls is summarized in Table I.

One of the first large-scale cases of malware infection was the Morris Worm, released in November 1988 by Robert Tappan Morris [1]. Morris took advantage of vulnerabilities in the *fingerd* and *sendmail* daemons in some versions of Berkeley Software Distribution (BSD) UNIX. The worm was written to affect Sun Microsystems Sun-3 systems and VAX systems running 4 BSD, however, it did not affect systems running the Sun-4 operating system (OS) even though Morris pointed out the flaw in its *fingerd* daemon to staff at Carnegie-Mellon University a year before the worm was released [1]. This oversight may have been designed to draw attention away from the worm's author, but shows that malware can be written to exploit not only a specific OS, but particular versions of that OS.

The Morris Worm checked to see whether a target was already infected and if so, would not re-infect it, thus limiting propagation and reducing resource consumption on affected systems. The worm was programmed to probabilistically skip this check one in seven infections to make it harder to eradicate [1]. Morris' lack of understanding of the potential propagation rate and incomplete testing caused the worm to replicate much faster than anticipated.

Table I. Mapping of Control Mechanisms to Representative Malware since 1982.

| | Check previous infection | Check OS and version | Check application software | Self propagation | Check date/time | Propagation counters | Checks language settings | Check for reverse engineering | Contact C2 | Other controls |
|---|---|---|---|---|---|---|---|---|---|---|
| **Elk Cloner (1982) [29]** | | Apple II | | Infects boot sector | | | | | | Payload executes every 50th boot |
| **Stoned (1987) [14]** | Check master boot record | | | Infects floppy | | | | | | |
| **Morris Worm (1988) [1]** | Attempt port connect | Sun-3 & Vax BSD | | using sendmail & fingerd | | | | | | |
| **Jerusalem (1989) [15]** | Checks files for infection | | | Infects boot sector | Payload executes on Friday 13th | | | | | |
| **Frodo (1989) [28]** | | | | Infects boot sector | Propagates on Sept 22 | | | | | |
| **Michelangelo (1991) [16]** | | | | Infects boot sector | March 6th | | | | | |
| **Concept (1995) [27]** | Checks for macro | | MS Word 6 | Infects Word docs | | Only sends to 50 contacts | | | | |
| **Melissa (1999) [26]** | Registry key | | MS Word 97/2000 | Sends itself to addr book entries | | | | | | Run only once per session; check day/time |
| **Code Red (2001) [25]** | | Windows NT/2000 | | IIS exploit in HTTP GET | Behavior based on day of month | | English Windows | | | Will not infect if file c:\ notworm is present |
| **Code Red II (2001) [24]** | Atom | Windows NT/2000 | | IIS exploit in HTTP GET | Termination on Oct. 1, 2001 | | Chinese Windows | | | Aware of IP address to avoid reinfection |
| **Blaster (2002) [23]** | Mutex | Windows2000/XP | | DCOM RPC(135) and tftp(4444) | Executes DoS on certain dates | | | | | |
| **Welchia (2003) [22]** | Mutex | Windows 2000/XP | IIS and msblast.exe | DCOM RPC(135) and http (80) | Terminates on Jan. 1, 2004 | | | | | Attempts to disinfect Blaster and patch host |
| **MyDoom (2004) [21]** | Mutex | | | Email or file sharing (Kazaa) | DDos on 2/1/04; stop spread 2/12 | | | | Listener | Avoids propagation to certain domains (.gov/.mil) |
| **Storm Worm (2007) [20]** | Registry key | | Shuts down A/V | | | | | Checks for VM or dbg | Overnet P2P | Geolocation via network address |
| **Conficker (2008) [19]** | Mutex | Windows variants | | Network and USB | Contact C2 starting on 26 Nov 2008 | | Ukranian settings | Check for VM | HTTP and P2P | |
| **IXESHE (2009) [18]** | Registry key | | | | | | | | HTTP | Primary vector is targeted email |
| **Stuxnet (2010) [3]** | Mutex Registry key | Windows variants | | Network and USB | Several date checks | Delete USB at 3 infections | | | P2P/RPC | Checks for specific attached hardware |
| **Flame (2012) [17]** | Mutex | Windows variants | | USB & Win update | | | | | HTTPS | |

Another control is to deliver a payload on a specific date. The Jerusalem virus, discovered in 1989, triggered on any Friday the 13th, and the Michelangelo virus (1991), deleted important data on March 6th [15] [16]. Targeting an organization on a specific date might help to coordinate a large scale cyber attack or to coordinate cyber-based effects with a kinetic operation.

The Concept virus (1995) was the first known to take advantage of macros in the Microsoft Office suite [27], infecting computers with Microsoft Word installed. In 1999, the much more sophisticated Melissa virus took advantage of the Microsoft Office macro framework, using Word macros to replicate via emails using Microsoft Outlook. Melissa had other controls, running only once per session to limit propagation and displaying a special message if the minute of the hour matched the day of the month at the time of infection [26]. Melissa limited propagation by sending itself to only 50 entries on the victims' Microsoft Outlook address book. Other email worms, such as MyDoom (2004) limit email propagation by avoiding certain domains [21]. One could imagine extending this functionality to limit propagation to address book entries with specific email domains, telephone prefixes, surnames, or mailing addresses.

A recent trend in malware is to terminate to prevent analysis when running in a virtual machine or debugger. An example is the Storm Worm mentioned above [20]. Another is one of the most ubiquitous worms ever deployed, Conficker, identified in November 2008 [19]. Conficker has used a variety of control mechanisms over several revisions. It self-replicates, trying to connect to other computers on a local network by exploiting a Windows service vulnerability [19]. Later variants replicated via removable media and using a peer-to-peer mechanism. Conficker checks the OS version to determine which exploit to trigger, checks network connectivity, and attempts to subvert firewalls. Version A would not infect systems whose keyboard language layout was set to Ukrainian or that had a Ukrainian IP address. Starting with version B, Conficker attempted to shut down antivirus products on the target. After infection, Conficker checks the date and beginning on 26 Nov 2008, attempted connections to command and control (C2) servers to download more code. It also encrypted its payload and employed anti-debugging logic to self-destruct if it sensed an attempt at forensic analysis [19].

From our analysis, the rise of the Advanced Persistent Threat (APT) in the late 2000s has seen more carefully targeted infection attempts, often in the form of direct emails that contain links to web sites or files that take advantage of application vulnerabilities to plant malicious code on the recipient's computer. After initial infection, many of these tools contact a C2 server for additional instructions or to download new modules. Removing self-propagation allows attackers to target an individual or organization with more precision. It is less reliable, however, because targeted emails can draw suspicion and they typically require user action, such

as opening a file or clicking on a link. An example of this type of threat is the IXESHE (2009) APT campaign [18].

One of the most tightly-controlled pieces of malware ever discovered is Stuxnet [3]. The consensus among several security firms, including Symantec, Kaspersky Labs, and others, is that Stuxnet was designed to cause subtle failures in industrial equipment. Before installing itself, Stuxnet ensures a certain system configuration is present. It first checks the operating system and version, choosing to only target specific Windows systems. Stuxnet then checks a registry key to determine whether the host is already infected and checks the system date, exiting if the date is after 24 June 2012.

Once installed, Stuxnet only affects specific types of Programmable Logic Controllers (PLCs) supervised by the Siemens Company's Step 7 software and connected to frequency converter devices manufactured by the Fararo Paya company in Iran or the Vacon company in Finland. Specifically, Stuxnet would only infect S7-315 PLCs attached to arrays of 33 or more frequency converter devices or S7-417 PLCs attached to 6 groups of 164 frequency converter drives.

Many suspect that Stuxnet was designed to target Iranian centrifuges at their Natanz uranium enrichment plant [3]. It did, however, propagate beyond Natanz, both through infected machines that left the network and joined another, and via USB flash drives. Analysis of the code indicates that Stuxnet should delete itself from infected USB drives after three infections and that it should have deleted itself after 21 days, however these controls were non-functional [30]. Stuxnet was discovered in 2010 after this error allowed it to infect systems in several countries [31]. While additional machines were infected and there was a cost associated with eradication, Stuxnet was inert on devices that did not meet the above configurations. Collateral damage therefore was minimized by the specific controls included in Stuxnet.

In May 2012, researchers at Kaspersky Lab identified a new piece of malware, dubbed Flame, whose primary propagation mechanism is infected USB drives. Flame is also the first known instance of malware to subvert Windows Update [17]. Infected machines can masquerade as Windows Proxy Auto-Discovery (WPAD) servers and hijack requests for Windows Updates within their local network to provide malicious patches. The authors of Flame used forged certificates that allowed them to make their illegitimate Windows updates appear to be signed by Microsoft. Infected hosts contact a C2 server for modules and instructions. C2 servers can send a kill module that causes the malware to be wiped from the system.

# 4. FRAMEWORK FOR MALWARE CONTROLS

Our malware control framework builds upon the cyberspace planes suggested by Fanelli (see Figure 1) [13]. The geographic plane includes the physical location of the target and includes the implications imposed by geographic boundaries, as well as physical aspects of the location of a specific target system such as power infrastructure and building location. The physical plane includes a device's physical hardware and protocols that allow for communication. This plane encompasses the physical layer (layer 1) of the Open Systems Interconnection (OSI) model, in addition to other features of a device's hardware such as serial numbers and types of attached peripheral devices. We subdivide the logical plane into the top six layers (layers 2 - 7) of the OSI model, which provides logical abstraction layers for communications systems (detailed in Table II). The cyber persona plane resides above the logical plane and includes individual virtual identities in the cyber domain. Finally, the supervisory plane includes persons and systems that provide the command and control necessary to start, stop, or redirect cyber weapons. Our framework for malware controls, discussed below, maps the cyberspace planes to cyber weapon control measures.

The following paragraphs differentiate between active and passive cyber weapon control measures, then map different types of control measures to the cyberspace planes in Figure 1.

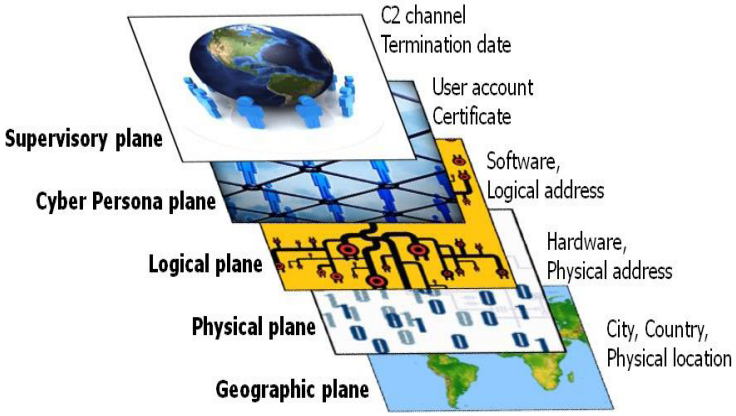


Figure 1. Cyberspace planes. The four planes described in [13] are expanded to include the Geographic plane and the OSI model

## A. CONTROL MEASURE CLASSIFICATIONS

We classify malware control mechanisms as active, passive, or hybrid. Passive (or autonomous) control has a cyber weapon observing its environment and acting on those observations, based solely on internal logic resident in its code. Observations can include a variety of system characteristics like those mentioned in Section 3, such as inspecting the current date, checking for existing copies of itself, examining registry keys, reviewing installed operating system and application software, or checking for attached hardware.

Active control measures allow an external decision-maker to decide what actions to take either by directly issuing commands or via code updates. Examples include malware that contacts a C2 server and receives instructions or a virus that opens a port and sends notification that the machine is ready to accept network connections.

Hybrid control is a combination of active and passive control measures. An example is malware that checks to see if a system meets certain configuration specifications and based on those observations, decides whether to contact a C2 node.

## B. CONTROL MEASURES BY CYBER PLANE

To maximize the reliability of malware targeting, the possible control measures at each cyber plane and OSI layer must be considered. These controls are summarized in Table II and are examined in the following paragraphs.

*1) Geographic Plane.* Here we consider control measures that require an agent deploying malware to be in the same geographic area as the targeted system. Someone might drop a USB "thumb" drive in a parking lot, hoping an employee will connect it to a computer that has network access to a target system. Someone might even be able to gain physical access to a target device to connect removable media or bypass login procedures to install software. These are all considered active control measures.

*2) Physical Plane/Physical Layer.* The physical plane of cyberspace maps almost directly to the physical layer of the OSI model. Different from the geographic plane, the physical plane includes components of a computer system and other hardware attached to it. Control measures at this layer are primarily passive ones, like checking serial numbers of peripheral devices or determining which physical layer protocols are being used. If reconnaissance of a target is sufficiently detailed, several such controls could be incorporated at the physical layer to provide very specific targeting, such as Stuxnet's checking for specific types of attached PLCs [30].

Table II. Malware C2 Methods Mapped to the Five Cyberspace Planes from Figure 1.

| Cyber Plane/OSI Layer | | Command and Control Mechanisms |
|---|---|---|
| **Supervisory plane** | | (Active) Control malware via active C2 architecture<br>(Active) Human decision maker<br>(Hybrid) Develop targeting code update and push to malware on system |
| **Cyber persona plane** | | (Passive) Check for specific identity – user ID, email address, social network identity, etc.<br>(Active) Collect and report identify information to a controller |
| **Logical Plane** | **7. Application layer** | (Passive) Check OS or application software and versions<br>(Passive) Check hostname, domain of target systems<br>(Passive) Check for presence (or absence) of VM host<br>(Passive) Check for evidence of debugger<br>(Passive) Check local date and/or time<br>(Active) Propagation counter - limit automatic propagation to fixed number |
| | **6. Presentation layer** | (Passive) Check for specific encryption or encoding techniques used to translate data between network and application formats<br>(Passive) Check language/character set translations |
| | **5. Session layer** | (Passive) Check application layer protocol fields (e.g. fields in ICCP or ELCOM protocol messages can identify specific SCADA systems) |
| | **4. Transport layer** | (Passive) Check for specific transport-layer protocols used by target (e.g. COTP or TPTK protocols can indicate SCADA systems) |
| | **3. Network layer** | (Passive) Check for network address of target area or organization |
| | **2. Data link layer** | (Passive) Check link-layer protocol used in network<br>(Passive) Check medium access control (MAC) address or organizationally unique identifier (OUI) |
| **Phy Plane** | **1. Physical layer** | (Passive) Check physically connected devices or device serial numbers<br>(Passive) Check for RS-485 physical layer, used by many SCADA control systems<br>(Active) Restrict propagation to specific removable media |
| **Geographic Plane** | | (Active) Insider/physical access<br>(Active) Drop thumb drive in parking lot |

*3) Logical Plane*. This plane consists of the operating system, application software and software settings on a device. We further subdivide this layer into the upper six layers of the OSI model.

*a) Data Link Layer.* Here we are concerned with the data link layer protocols and addresses. Reconnaissance may tell us that an organization uses network adapters from a certain manufacturer. Malware could be programmed take action only on network interfaces with certain Organizationally Unique Identifiers (OUI). Data link layer protocols that might be examined include Ethernet, WiFi, Bluetooth, ZigBee, or others.

*b) Network Layer.* At the network layer, network addresses could be compared to a target entity's assigned network address space. While Network Address Translation (NAT) limits the reliability of inspecting network addresses, a thorough examination of a device's network environment, including routers and gateway addresses, might allow a better picture of the physical location of a device to be developed.

c) *Transport Layer.* While the majority of Internet connected systems use Transport Control Protocol (TCP) or User Datagram Protocol (UDP), many supervisory control and data acquisition (SCADA) systems and industrial control systems (ICS) use transport protocols tailored to those systems. Examples include the Connection Oriented Transport Protocol (COTP) and TPTK, which are sometimes used in place of, or in conjunction with, TCP [32]. Differentiating between SCADA or ICS systems and other networked devices might be important in a cyber campaign when a country's electrical infrastructure or power generation capability are to be targeted.

d) *Session Layer.* This layer, and the following presentation layer, are treated as part of the application layer in some network models. Here we list these layers separately as they have specific purposes with which cyber weapon control measures might be associated. The session layer provides the ability to establish a semi-permanent connection between two end points. At this layer, the Inter-Control Center Protocol (ICCP) and IEC 60870-6 (ELCOM) protocols are used for communication between utility control centers in SCADA and emergency management systems (EMS). Again, the ability to positively identify specific SCADA systems might be advantageous during cyber weapon employment.

e) *Presentation Layer.* This layer is used to convert data encoding or encryption formats used for network transfer to and from formats that can be used by the application layer. Passive control measures at this layer might include checking for specific data encoding or encryption techniques known to be used by the target entity.

f) *Application Layer.* There are a variety of passive checks that can be made at the application layer. Cyber weapons might be programmed to check for specific operating system software or versions, certain application software or versions, hostname, username, domain name, environment data such as date, time, or location settings, or the presence or absence of a virtual machine environment.

4*) Cyber Persona Plane.* As defined by Fanelli [13], this plane identifies identities in the cyber domain, which might have many-to-one or one-to-many, or many-to-many relationships with individuals in the physical world. The presence or absence of specific identities may be used to validate targets or limit application of effects.

Controls on the cyber persona plane consider indications of use or ownership by a specific person, group, corporation or government. Examples include account credentials, certificates, cookies, licensed software, biometric data, and observations of network activity such as logging into accounts correlated with a persona

5) *Supervisory Plane*. At the top of the hierarchy is the supervisory plane, which provides oversight and the authority to start, stop, modify, or redirect a cyber weapon or cyber campaign, within the limits of the weapon's capabilities and C2 infrastructure. At this level, operational decisions are made about the prosecution of a cyber campaign.

# 5. ANALYSIS AND CONCLUSIONS

One approach to analyzing malware controls is to specify undesired effects that cyber weapons might create, actions that can be taken to mitigate those effects, and corresponding controls that can provide mitigation. Table III provides one such analysis.

Based on our framework and the large variety of controls that can be used at varying levels of specificity and effectiveness, we believe that cyber weapons can be very carefully crafted and targeted to affect only specific systems and organizations, greatly reducing undesired collateral effects. As with kinetic attacks, more detailed intelligence allows for better targeting and weapon development. Decision-makers must weigh the value of a target against the potential for collateral damage and may have to assume risk. The difficulty in attributing a cyber attack to a specific entity might reduce the risk of being held accountable for collateral damage, but it does not alleviate the moral responsibility to limit it. Furthermore, very fine-grained controls used to ensure that cyber weapons will only affect specific targets might provide clues to the origin of those weapons.

Another risk that cyber weapon authors must consider is the potential for controls included in their software to identify their intentions. Had Stuxnet been analyzed before centrifuges were damaged, Iran might have suspected that those centrifuges were the target, causing them to tighten defenses. One novel approach to prevent such analysis is to encrypt the malware payload and use data gathered from the infected system, such as registry entries, portions of the physical or network address, or device serial numbers, to generate a decryption key. This technique is used in the Gauss malware (2012), which gathers information about the system path and installed software, then calculates an MD5 hash and attempts to use it as a key to decrypt the payload [33]. As of this writing, security researchers have not been able to decrypt and analyze the Gauss payload.

Table III. Malware Collateral Effects, Mitigating Actions, and Representative Controls

| Undesired collateral effects | Mitigating actions | Representative Controls |
|---|---|---|
| Unintended infection | Limit propagation to specific targets | Disallow self-replication<br>Infect systems only via spear-phishing with malicious attachment or link to download or through previously infected systems |
| Unintended payload execution causing loss of:<br>Confidentiality (data exposure)<br>Availability (loss of data, denial of service, consumption of network resources)<br>Integrity (data modification) | Prevent payload execution on non-target systems | Use only active control measures to activate payload<br>Use detailed reconnaissance to determine triggers for passive or hybrid control<br>Trigger malware based on known target configuration |
| Vulnerability disclosure to unintended individuals or general public | Prevent reverse engineering and subvert forensic investigation | Encryption<br>Tamper protection<br>Temporary payloads that delete themselves from memory |
| Attribution of attack or source of the malware | Eliminate evidence of authors | Encryption<br>Tamper protection<br>Use widely used languages, libraries, and coding techniques<br>Temporary payloads |

Despite the care with which cyber weapon controls may be developed, there is always the possibility of undesired effects such as affecting the wrong target. The ability to control malware is only as good as the intelligence informing its development. Just as kinetic weapons should not be used without sufficient intelligence regarding the target, cyber weapons should not be used unless intelligence is available to adequately limit potential damage to non-target systems.

As nations increasingly recognize the potential for cyber weapons as tools of warfare, it is important to find ways to ensure that they are used responsibly in a way that conforms with the LOAC and minimizes unwanted collateral effects. Since the introduction of malicious software, techniques have been used to control it, either actively or passively, to target specific systems or otherwise shape its effects. In this work we have established the potential to better control the behavior of cyber weapons and summarized previously used techniques. We go on to develop a framework for malware controls, mapping them using our cyber planes model and categories of propagation techniques. This framework can be used to incorporate effective controls during the development of cyber weapons. Of particular value is the ability to analyze malware in the context of this framework to determine whether it conforms to internationally recognized standards of ethical behavior during design and planning, while in use, and during post-use analysis by the aggressor, the target entity, or third-parties seeking to verify appropriate behavior.

# REFERENCES


[1] E. Spafford, «The Internet Worm Program: An Analysis,» Purdue University Technical Report CSD-TR-823, 1988.

[2] Federal Bureau of Investigations, «DNSChanger Malware,» September 2012. [Online]. Available: http://www.fbi.gov. [Accessed 4 September 2012].

[3] Falliere, L. Murchu and E. Chien, «W32.Stuxnet Dossier, v1.4,» Feb 2011. [Online]. Available: http://www.symantec.com. [Accessed September 2012].

[4] E. Tikk, K. Kaska, K. Runnimeri, M. Kert, A. Taliharm and L. Vihul, «Cyber Attacks Against Georgia: Legal Lessons Identified,» Tallin, Estonia, November 2008.

[5] M. Johnson, «Georgian Websites Under Attack - Don't Believe the Hype,» August 2008. [Online]. Available: www.shadowserver.org. [Accessed 4 September 2012].

[6] M. Walzer, *Just and Unjust Wars*, New York: Basic Books, 1977.

[7] D. Ball, «China's Cyber Warfare Capabilities,» *Security Challenges,* vol. 7, no. 2, pp. 81 - 103, Winter 2011.

[8] D. J. Smith, «How Russia Harnesses Cyber War,» *Defense Dossier,* vol. 4, pp. 7 - 11, August 2012.

[9] I. Berman, «Cyberwar and Iranian Strategy,» *Defense Dossier,* vol. 4, pp. 12 - 15, August 2012.

[10] F. Cohen, «Computer Viruses, Theory and Experiments,» *Computers and Security,* vol. 6, no. 1, 1987.

[11] M. Ludwig, *The Little Black Book of Computer Viruses*, Show Low, AZ: American Eagle Publishing, 1990.

[12] M. Ludwig, *The Big Black Book of Computer Viruses*, Show Low, AZ: American Eagle Publishing, 1995.

[13] R. Fanelli and G. Conti, «A Methodology for Cyber Operations Targeting and Control of Collateral Damage in the Context of Lawful Armed Conflict,» in *International Conference on Cyber Conflict (CyCon)*, Tallinn, Estonia, June 2012.

[14] «Virus:Boot/Stoned,» F-Secure, [Online]. Available: http://www.f-secure.com. [Accessed 1 October 2012].

[15] «Jerusalem,» F-Secure, [Online]. Available: http://www.f-secure.com. [Accessed 30 September 2012].

[16] «Michelangelo (computer virus),» [Online]. Available: http://en.wikipedia.org. [Accessed 12 October 2012].

[17] M. Lee, «Flame Used Windows Update to Spread,» 5 June 2012. [Online]. Available: http://www.zdnet.com. [Accessed 4 August 2012].

[18] D. Sancho, J. D. Torre, M. Bakuei, N. Villeneuve and R. McArdle, «IXESHE: An APT Campaign,» Trend Micro Research Paper, September 2012.

[19] N. Fitzgibbon and M. Wood, «Conficker.C: A Technical Analysis,» 2009.

[20] X. Chen, J. Andersen, Z. M. Mao, M. Bailey and J. Nazario, «Towards an Understanding of Anti-virtualization and Anti-debugging Behavior in Modern Malware,» in *IEEE International Conference on Dependable Systems and Networks (DNS)*, Ann Arbor, MI, June 2008.

[21] «Worm:W32/Mydoom,» [Online]. Available: http://www.f-secure.com. [Accessed 18 December 2012].

[22] «Worm:W32/Welchi,» F-Secure, [Online]. Available: http://f-secure.com/. [Accessed 30 December 2012].

[23] «Virus alert about the Blaster worm and its variants,» Microsoft, [Online]. Available: http://support.microsoft.com. [Accessed 18 December 2012].

[24] D. Moore, C. Shannon and J. Brown, «Code-Red: a case study on the spread and victims of an Internet Worm,» in *2nd ACM SIGCOMM Workshop on Internet Measurement*, Marseille, France, November 2002.

[25] J. C. Dolak, «The Code Red Worm,» SANS Institute Reading Room, 2001.

[26] Carnegie-Mellon University Software Engineering Institute, «CERT Advisory CA-1999-04 Melissa Macro Virus,» March 1999. [Online]. Available: http://www.cert.org. [Accessed 12 August 2012].

[27] «Virus:W32/Concept,» F-Secure, [Online]. Available: http://f-secure.com. [Accessed 12 October 2012].

[28] «Frodo,» [Online]. Available: http://virus.wikia.com. [Accessed 14 December 2012].

[29] «Elk Cloner,» [Online]. Available: http://virus.wikia.com. [Accessed 12 December 2012].

[30] B. Schneier, «Stuxnet,» 7 Oct 2007. [Online]. Available: http://www.schneier.com. [Accessed 19 December 2012].

[31] D. Sanger, «Obama Order Sped Up Wave of Cyberattacks Against Iran,» 1 July 2012. [Online]. Available: http://www.nytimes.com. [Accessed 25 July 2012].

[32] IETF, «RFC 1006 - ISO Transport Service on top of the TCP,» May 1987. [Online]. Available: http://tools.ietf.org/. [Accessed 1 December 2012].

[33] Kaspersky Labs, «The Mystery of the Encrypted Gauss Payload,» 14 August 2012. [Online]. Available: http://www.securelist.com. [Accessed 20 Dec. 2012].