Solution Architecture for Cyber Deterrence

Thomas Mowbray
Solution Architecture for Cyber Deterrence

GIAC (GPEN) Gold Certification

Author: Thomas J. Mowbray, Thomas.Mowbray@TASC.com
Advisor: Dr. Kees Leune

Accepted: April 12, 2010

Abstract

For a government cyber deterrence strategy to be effective, it must have network penetration tools, as well as, tools for distributed denial of service (DDOS), parallel scanning, reconnaissance, surveillance, and other capabilities. Most importantly, it must be able to assess cyber-attack attribution rapidly, and with certainty.¹ This paper furthers the definition of cyber-deterrence architectures and evaluates of elements of future architectures in a penetration testing environment. Leveraging available policy research, a line-of-sight analysis is conducted from strategic goals to pen testing source code, filling in important architectural gaps. Policy implications of the proposed technical solutions are discussed. Cyber-deterrence capabilities are assessed at strategic and technical levels, technologies are envisioned that provide components of the solution, and the results are documented as conceptual architecture with research prototypes.

¹ The opinions expressed in this paper are those of the author.
1. Introduction

The mission of cyber deterrence is to prevent an enemy from conducting future attacks by changing their minds, by attacking their technology, or by more palpable means. This definition is derived from influential policy papers including Libicki (2009), Beidleman (2009), Alexander (2007), and Kugler (2009). The goal of cyber deterrence is to deny enemies “freedom of action in cyberspace” (Alexander, 2007). In response to a cyber attack, retaliation is possible, but is not limited to the cyber domain. For example, in the late 90’s the Russian government declared that it could respond to a cyber attack with any of its strategic weapons, including nuclear (Libicki, 2009). McAfee estimates that about 120 countries are using the Internet for state-sponsored information operations, primarily espionage (McAfee, 2009).

Perhaps the most famous international cyber war started April 27, 2007, when Estonia was hit by a 3-week, nationwide cyber attack (Beidleman, 2009). In the previous years, Estonia had sponsored national initiatives to become the world’s most high tech society, and had become deeply dependent on the Internet. Almost all government services were delivered electronically, and cash purchases at retail points-of-sale had been replaced with Internet transactions. When the country was then bombarded with distributed denial of service (DDOS) attacks like ping floods, the economy was virtually shut down for weeks. These attacks are believed to be related to a political incident involving the relocation of a statue of Lenin, but several years later only one man has been convicted, and there is no credible evidence of Russian government complicity.

Similarly, a cyber war waged against Georgia during a real-world military conflict with Russia in August, 2008, shut down most of the country’s communications systems. (Beidleman, 2009) This example is more typical of strategic national uses of cyber war. An in-depth study of the People’s Republic of China’s cyber doctrine found that primary use of cyber warfare is the temporary disruption of systems and communications to delay military retaliation (Krekel, Bakos, & Barnett, 2009). For example, were China to launch

---

2 The definition of deterrence has been unified to encompass the views of primary policy sources. For example, “palpable means” may include confiscation, termination, incarceration, casualty, death, or destruction.

Author Name, email@address
its long-anticipated invasion of Taiwan, focused cyber attacks on key United States (US) military nodes would likely be used to delay US military response (according to Krekel et. al., 2009), until after the conquest becomes a fait accompli.

But is cyber war hype? Many pundits would agree. (Walt, 2010) What is happening “in the wild” on the Internet today is largely uncontrolled and deeply intrusive, but does it rise to the level of war? Cold war, certainly, but hot-war, quite rarely, except in cases such as Estonia and Georgia. This paper clearly distinguishes between cyber-crime and cyber-war. It is the difference between peace-time rules of engagement and war-time rules of engagement. For our purposes, cyber war engages the active-duty military of a nation state in the aggressive defense of its territory, citizens, and resources.

The United States has recently released a summary of its national cyber strategy, the Comprehensive National Cybersecurity Initiative (CNCI) (Executive Office of the President, 2010). The CNCI includes 14 diverse initiatives such as defense of government networks, R&D coordination, and situational awareness through connections between cyber operations centers. One of the initiatives addresses cyber deterrence directly:

*Initiative #10. Define and develop enduring deterrence strategies and programs. Our Nation's senior policymakers must think through the long-range strategic options available to the United States in a world that depends on assuring the use of cyberspace. To date, the U.S. Government has been implementing traditional approaches to the cybersecurity problem—and these measures have not achieved the level of security needed. This Initiative is aimed at building an approach to cyber defense strategy that deters interference and attack in cyberspace by improving warning capabilities, articulating roles for private sector and international partners, and developing appropriate responses by both state and non-state actors.* (Executive Office of the President, 2010)

News coverage of the release of this CNCI summary document revealed that cyber deterrence is still in the very early planning stages. A former US official stated that work to date on cyber deterrence and computer network attack is incomplete because
decision-making criteria have not been developed. A cyber deterrence decision calculus that might fill this role is discussed in Section 5.

Acquisition of military cyber warfare capabilities is escalating and becoming increasingly public. For example, as openly declared in their press, the People’s Republic of China organizes forces of cyber warriors that “have an all-conquering offensive technology” (Alexander, 2007).

The United States\textsuperscript{3} (US) military is actively establishing cyber commands with defensive and offensive cyber missions in anticipation of a joint-services cyber force, the US Cyber Command. Recently the US government declared operational three component cyber forces: the Marine Corps Force Cyber Command, the 24\textsuperscript{th} Air Force, and the Navy’s U.S. Fleet Cyber Command/U.S. 10\textsuperscript{th} Fleet. According to the public FY2010 budget justification, the Department of Defense (DoD) is spending about US$30 million per year on cyber warfare research and development (R&D). The immense strategic importance of cyberspace to social stability and the world economy should merit a budget allocation orders-of-magnitude greater.

Military doctrine defines three major areas of Computer Network Operations (CNO): computer network defense (CND), computer network attack (CNA), and computer network exploitation (CNE) (DoD, 2006b). It is expected that the majority of IT security investment goes to improving and maintaining network defenses. However, as the CNCI admits, CND efforts to date are grossly inadequate. Even the most advanced high-tech organizations are frequently penetrated and exploited by Advanced Persistent Threat (APT) (Rafferty, 2010). Strong CNA and CNE capabilities must be developed as key elements of a comprehensive defense posture that includes cyber deterrence.

The elements of cyber deterrence should be explored, and important strategic considerations taken into account, including legal frameworks authorizing CNA and CNE for defensive operations. If defensive cyber attacks are conducted, how could they be legalized? How can it be conducted in an effective and efficient manner?

\textsuperscript{3} The United States is used here as a representative of nations defending against cyber attack for many reasons: the author’s perspective, abundant open source information, and US national security goals which represent the world’s interests (see Section 2, Figure 2-1).

Author Name, email@address
2. **Methodology and Research Assumptions**

Network penetration technologies and military botnets are tools for cyber deterrence, but technical research cannot have much impact without connecting them back to legal and international realities. How could we operate a military botnet or use our pen testing skills to fight cyber enemies lawfully? Certain elements are assumed immutable, including National Security Goals (NSG), Strategic Functions, and Technical Functions (see Figure 2-1). The NSGs are taken directly from a standard military textbook (Kugler, 2006); the Strategic Functions are based on a paper by a RAND policy expert (Libicki, 2009); and the technical functions are based on course material from the SANS Institute, real-world pen testing experience, and hands-on work with other toolkits such as BackTrack⁴.

The variable elements in Figure 2-1 are the deliverables of this research. The Solution Architecture for Cyber Deterrence is scoped in Section 6. Elements of it are detailed in Sections 7 and 11. In order to legally do what needs to be done technically, we make some suggestions for enabling treaties and legislation in Section 4.

---


Author Name, email@address
Solution Architecture for Cyber Deterrence

Cyber deterrence and related hard problems are solvable with current technologies, assuming:

Figure 2-1. Framework for this Research – A Strategic Line-of-Sight
• Enabling treaties, laws, and policies are enacted to make cyber deterrence legal and diplomatically workable\(^5\) (see Section 4); and

• Sufficient a priori knowledge of cyber attackers is collected to rapidly assign attribution and plan counter attacks in most instances (see Section 7).

There is no point in re-doing the outstanding work of professional policy researchers in cyber deterrence. To avoid getting bogged down in policy debates, this research assumes the results of their work and builds on their achievements. Many relevant points of their research are summarized in Sections 1, 3, and 5.

This study followed a systematic approach beginning with a broad survey to define requirements and selecting a handful of focus areas for hands-on experimentation. The major steps included:

1) Literature Search – review cyber deterrence papers & compile bibliography

2) Characterize “the deterrence problem” and this paper’s focus/scope

3) Define solution reference model – block diagram that reveals the major Deterrence capabilities and scope of the solution

5) Select key focus areas and conduct technical analysis – Cyber deterrence is an expansive domain that could require many volumes to cover completely. In this gold paper, we focus on one of the most-critical requirements for cyber deterrence: determining attack attribution within milliseconds. We also deep dive into a technology of particular interest for penetration testers, parallel and distributed scanning, with source code examples and performance benchmarks.

Over the past two years, many discussions, and hands-on pen testing experiences have contributed to this research. A cyber warfare reading group was established at Northrop Grumman (now TASC) and helped explore botnets in depth. This group hosted many deep-dive discussions of major botnet architectures, including Ghostnet, Torpig, Storm, fast flux botnets, and ICQ botnets, as well as technical methods of the APT.

\(^5\) One of the major NSG goals is maintenance of US diplomatic alliances. Actions such as cyber retaliation directed at foreign assets could easily lead to diplomatic fallout. Future treaties should address these inevitabilities.

Author Name, email@address
3. **Cyber Deterrence Challenges**

Defining a solution architecture for cyber deterrence is difficult due to technical, legal, and strategic factors, including:

- the inherent difficulty of assigning attribution on the Internet, (and the ease of hiding attribution or misdirecting attribution it to other parties);
- the unpredictability the impacts of cyber attacks;
- the potential for damage due to counter-retaliation; and
- the fact that states, non-state actors, and individuals are at a peer level, capable of waging cyber attacks with readily available attack tools – For example, third parties joining the fray is a common occurrence during publicized Internet attacks (Libicki, 2009).

There is no directly applicable legal framework for a basis for cyber war. The nearest analogy is the real world Law of War, which comprises international treaties going back hundreds of years. One of the most applicable treaties is the Charter of the United Nations (UN), signed in 1945, which differentiates between legal and illegal acts of war. The charter sanctions self defense against use of force by states that threaten territorial integrity or political independence. The UN charter defines the scope of armed attacks as “invasion or attack, bombardment, blockade of ports or coasts, and attacks on land, sea, or air forces of another state” (Beidleman, 2009). If the impacts of a cyber attack could be shown as equivalent to one of these real-world acts of war, then a CNA response could be legally justified.

However, physical damage is a possible but unlikely cyber attack outcome. A cyber attack that scrambles the records of major financial institutions in New York City would be devastating to the US and world economy, but would impart no physical damage, thus not constituting an act-of-war. More likely are cyber attacks such as Estonia and Georgia suffered, deeply disrupting communications and systems through denial of service.

No fatalities have been directly attributable to cyber attacks. However, fatalities are possible, as demonstrated by a simulated cyber attack on the Washington DC subway
system, which led to dozens of simulated fatalities (Beidleman, 2009). In that case, Schmitt Analysis technique was applied, leading to the conclusion that it was indeed an “armed attack.” However, this attack was judged much less significant than an attack by humans onsite, since in this case the attackers were remote. Schmitt Analysis is a method for deciding if a cyber attack meets the criterion of the UN Charter for acts of war (Beidleman, 2009). A key weakness of Schmitt Analysis is its heavy reliance on physical damage, injuries, and fatalities as key criteria.

The Council of Europe’s Convention on Cybercrime addresses cyber criminality but does not address cyber war by states and non-state actors. The convention is signed or ratified by about 40 developed nations including the United States, Canada, South Africa, Japan, and various countries of Eurasia. Many countries with non-coincidental association with malware and cyber attacks (such as Russia, China, and Brazil) are not signatories. The treaty includes standard definitions of cybercrime and leaves it up to each individual country to define laws and penalties, which vary widely.

The lack of policy and legal frameworks was dramatically played out in a simulated cyber war attack on the US national infrastructure, in a televised exercise called Cyber Shockwave (Bipartisan Policy Center, 2010). The players in the simulation were experienced former Federal executives, who had played similar roles in prior US administrations. News coverage of the event was widespread, and sources are listed on the Bipartisan Policy Center web site. Here a few choice examples with important observations of the event:

- National Public Radio reported: "The general consensus of the panel today was that we are not prepared to deal with these kinds of attacks," said Eileen McMenamin, vice president of communications at the Bipartisan Policy Center. "Whether these threats come from individual hackers, state organizations or terrorist groups, they are very real and something we really need to be prepared for."

- Techeye.net reported: Senior officials said that legislation was needed to provide for defence against a cyber attack, while private organisations were totally unprepared for such a scene to be played out in reality.

Author Name, email@address
• AOLnews.com reported: The intelligence community might be able to track down the attacks to a specific country or even pinpoint computer servers, but figuring out precisely what person or group is behind the attack may be impossible. "Without attribution, we can't go to the issue of retaliation" … a participant stated.

4. Legal and Treaty Assumptions

Due to the magnitude of cyber threats, there is an urgent need for countries to make headway on laws and treaties addressing cyber warfare. There are more than 2,000 active botnets on the Chinese Internet alone, and possibly an order of magnitude more botnets globally (Zhuge, Holz, Han, Song, & Zou, 2007). Even a relatively modest botnet of 1,000 hosts could take down almost any business website or government Internet portal with a denial of service attack. Each botnet owner is capable of significant acts of cyber crime and cyber war. Governments must act to manage these threats to protect their own societies and the global economy, which is increasingly dependent upon the Internet.

From Cyber Shockwave and other evidence, we can conclude that an effective cyber deterrence strategy and architecture are not feasible given current US and international legal and policy frameworks. In order for this research to proceed, we must make some enabling assumptions about future legal and treaty frameworks.

The following list of assumptions is driven by our technical research as shown in Figure 2-1. Our research premise is that we can solve cyber deterrence challenges technically (see Sections 6, 7, and 11), but a key caveat is that we need to have a legal justification where none exists today. As a result, this section answers the question: If we must do what we propose technically, what legal justification do we need in terms of treaties and laws?

This section is not a formal legal analysis. However, discussion is possible using the reasonable-person standard of common sense. Eventually, legal formalities can be resolved by diplomats, legislators, and advisors. This discussion also assumes that our cyber operation is explicitly military in nature.
• We suppose that there will be future national laws and international treaties governing the conduct of cyber war and cyber operations. We will call the member nations in future treaties “signatories.”

• Future national laws will allow legal conduct of cyber operations against perceived enemy states and non-state actors.

• Future international treaties will allow legal conduct of cyber operations traversing signatories’ networks. The treaties should not mandate the disclosure of the cyber operations (an important provision, addressed as sub rosa retaliation and implicit deterrence in the next section).

• Servers involved in cyber attacks might be owned and operated by unaware third parties. Future treaties and laws should allow that servers used in cyber attacks or illegal activities can be exploited for monitoring purposes by signatory states, a critical provision for resolving attribution. If these servers are determined to be non-life critical (i.e. not critical to healthcare delivery or power infrastructure), defensive CNA can be legally conducted to these machines and networks in response to an attack (including the use of the machines as a counter-attacking force).

• Treaties should define the criteria for state-attribution of an attack, and how state-enablement of attack (such as the location of attack servers within national boundaries) can be addressed diplomatically, militarily, and legally.

• Certain cyber operations should probably be restricted by treaty. For example, the use of third party hosts (machines not involved in cyber crimes or cyber war attacks) to conduct defensive CNA should probably be prohibited.

• The differentiation between military and non-military is a key tenant of the legal (defensive) conduct of warfare. A legitimate military force always wears uniforms with insignia; otherwise, a non-uniformed fighting force could be considered unlawful combatants and potential war criminals. This concept could be emulated in cyberspace by a treaty provision such as: signatories shall use readily

6 This statement does not exclude the creative adoption selective items of local apparel to further peaceful civilian-military relations.

Author Name, email@address
attributable network assets to conduct military cyber operations, i.e. the IPs and domains shall be readily traceable to military government sources though common Internet services such as whois.\(^7\)

- For purposes of law enforcement and cyber defense, there should be government authorities that would allow official investigations of illegal and potentially hostile (i.e. cyber war and cyber terror) activities on 3rd party servers.

- For major cyber-attacks, attribution can very likely be determined or strategically inferred. (Kugler, 2009). International cooperation, diplomacy and law enforcement should eliminate many of the criminal and independent threats. As Kugler (2009) states, that leaves the remaining threats to be peer-level state actors, rogue states, and terrorists. Deterrence of each potential adversary should be handled in a tailored response. (Kugler, 2009).

With these assumed treaties and laws in place, cyber retaliation (defensive CNA) would be considered fair play under certain treaty-proscribed conditions. We have also allowed the use of both implicit and explicit cyber deterrence policies, and substantial leeway in terms of the need for public disclosure of cyber operations. In the next section, we apply these legal assumptions to the strategy of cyber deterrence.

5. **Cyber Deterrence Strategy**

While it is not necessary to be a policy expert, it is useful to understand how cyber deterrence strategies and policies would operate in practice. Fortunately, RAND Corporation researcher Martin Libicki (2009) has conducted the needed policy and strategy analysis, and RAND has granted permission to use some of his key figures in this paper to help explain the concepts. Libicki developed his RAND theories under the sponsorship of the US Air Force (USAF) in late 2009, at the same time the USAF was standing up their own cyber force, the 24\(^{th}\) Air Force (as well as Navy and Marine Corps). Given the timing, it is a very influential paper, and representative of the state of the art of cyber deterrence strategy.

---

\(^7\) Non-military uses of cyber warfare is not addressed here; it is beyond the scope of this gold paper.

**Author Name, email@address**
Figure 5-1 is Libicki’s concept for how a response to a cyber war attack would be conducted. Note that his RAND manuscript, Cyberdeterrence and Cyberwar, includes hundreds of pages of possible actions and potential reactions, detailing strategic tradeoffs in cyber war. In this model, Libicki uses the term “sub rosa” to indicate that the retaliatory cyber attack is intentionally not publicized. To be a deterrent, the original attacker must draw the conclusion that the counter attack is connected to the original attack. A key goal of cyber deterrence is changing the potential attackers’ mindset, forcing them to reconsider the benefits and consequences of conducting an attack. Diplomatic, kinetic military, and public relations approaches are not addressed in this paper, only cyber-related aspects of deterrence.

In an interesting case study, the DoD conducted a cyber attack on a Saudi website suspected of recruiting and planning insurgent warfare in Iraq (Nakashima, 2010). The recruiting website was actually sponsored by the Saudi government and another US agency to collect information about extremists. DoD took action after a task force of major US agencies concurred, but not unanimously. The resulting attack “inadvertently disrupted more than 300 servers in Saudi Arabia, Germany, and Texas.” (Nakashima, 2010) Diplomatic consequences included expressions of frustration by Saudi and German governments.
In Figure 5-1, there is a situational awareness (or surveillance) activity that detects candidate cyber war incidents. Second, the defenders must decide if the event is an actual attack, versus some other type of incident such as an unrelated hardware failure. Third, a causal analysis determines if the motivations for attack are consistent with a state actor. Fourth, the defenders must determine the level of public awareness of the attack. For example, a greater public awareness, such as a widespread communications or power failure (as simulated in Cyber Shockwave), could make the necessity of an explicit response more urgent to policymakers. Fifth, state or non-state attribution is assessed.

Author Name, email@address
Proving attack attribution is one of the most technically and strategically difficult steps in the process, and should figure prominently in our solution architecture. Sixth, the strength of the case for public attribution is assessed. Seventh, methods of retaliation are considered. Note that cyber retaliation is only one of many possible methods. For example, diplomatic gestures, law enforcement, economic sanctions, or kinetic attacks are other possibilities. It is interesting to contrast Libicki’s prescriptive approach to cyber deterrence with the policy-driven approach offered by (Kugler, 2009). Kugler places emphasis on thresholds and control of escalation. Cyber attacks can be characterized by severity, and a strategic calculus defined based upon a ladder of escalation, with escalating severity of responses at each level. Kugler (2009) also emphasizes the need for proactive analysis of each potential cyber adversary, and appropriate public and private deterrence messaging. The need for an explicit US cyber deterrence policy that communicates resolve, willpower, and credible capability for overwhelming response is critical. (Kugler, 2009).

In Figure 5-2, Libicki quantifies cyber deterrence decision-making with sample numbers. The decision is primarily driven by the strength of attribution, shown in the left two columns. In this sample, impact values are assigned the various outcomes. Libicki assumes an “ouch” factor, assigning the impact of the initial attack on the defender’s interests. The “oops” factor assigns impact of counter-attack with wrong attribution. Explicit deterrence means that the counterattack policy is disclosed to the attacker, possibly by a public announcement. Implicit deterrence involves no public or direct disclosure to the attacker. The “risky” factor has implicit and explicit values signifying the risk of counterattack, as does the “wimpy” factor signifying the public and attacker perception that the defender will not retaliate. The above factors would probably be pre-established for a variety of situations, and the odds (including of attribution) are adjusted for a given incident. Libicki (2009) totals the relative cost, which he calls “pain,” for the major policy options, assigning 122.25 points for an implicit deterrence policy and 154.63 points for explicit policies. In this case, Libicki (2009) suggests that implicit deterrence is the best option.
The implications of a cyber deterrence strategy must be considered in terms of an architectural reference model.
6. Reference Model

This paper presents an architectural design perspective for cyber deterrence, implying that the scope of the solution will be outlined as well as some of the key requirements and design concepts without over-constraining the engineering of a future implementation.

To define scope, we can envision a cyber deterrence architecture with the required capabilities as in the figure below. A key requirement is that this architecture must provide a wide range of response options. (Kugler, 2009).

![Cyber Deterrence Reference Architecture](image)

*Figure 6-1. Cyber Deterrence Reference Architecture*

Figure 6-1 identifies four capabilities groupings, including surveillance, penetration, integration, and advanced capabilities. The surveillance capabilities ensure that the defenders are aware of potential cyber attackers, their capabilities and their actions. Penetration capabilities could be used for computer network exploitation (CNE), in order to understand potential or actual attackers, and investigate attribution and certain forms of CNA. Integration capabilities allow human and automated knowledge sources to collaborate to build an understanding of the computer network environment, such as populating a knowledge base about potential attackers, to accelerate the assignment of

Author Name, email@address
7. Solution Architecture

Certain aspects of this architecture are more interesting and challenging than others. To be true to architectural principles, we should consider a few of the more challenging aspects, and lay others aside for engineering analysis or future research.

One of the capabilities that could be engineered is military botnets. A vision for a military botnet was proposed in Armed Forces Journal that would reside on existing hosts (e.g. administrative workstations) distributed worldwide on military installations (Williamson, 2008). It is unnecessary to use rootkits for this botnet, because the software is installed overtly. The botnet would need a resilient command and control infrastructure but does not need to be stealthy as seen in botnets in the wild. A peer-to-peer communication mechanism as used by the majority of the Storm Worm bots is particularly resilient and difficult to hijack (Holz, Steiner, Dahl, Biersack, & Freiling, 2008). The tradeoff is that peer-to-peer command and control (C2) is indirect, and might not be as timely as required. A direct C2 botnet communication structure (possibly using broadcast packets) for speed, paired with a redundant peer-to-peer mechanism, could provide the advantages of both approaches.

A military botnet could be used for distributed scanning. This might have a deterrence effect, knowing as potential attacker that an easily attributable government is actively scanning your Internet assets. Alternatively, active scanning might force even more stealth or have diplomatic consequences. Scripting automated scans on particular hosts and ports using widely available tools is not difficult. The author has created over a
dozen of these simple scripts for a custom pen test toolkit, using Linux/Unix bash shell and common pen testing tools including amap, nmap, dig, nslookup, and netcat. Unfortunately, even simple scans on a local subnet, such as ping sweeps, can take nearly a whole second per host, including operating system overhead (see Section 11.3 Performance Benchmarks). Obviously that would violate our timeliness requirement.

To accelerate scanning, scans could be distributed across a botnet, by employing parallel processing techniques, such as map-reduce, the parallel search architecture used by Google (Dean, & Ghemawat, 2004). For example, we could use a custom host list to parcel out distributed scans (map phase), and aggregate results back to a command server (reduce phase). For this paper, the author programmed a simple botnet as an architectural prototype in Python. The botnet performs threaded parallel and distributed scanning. Annotated source code and performance measures appear in the Appendix. Other than useful architectural insights for military botnets, the key result of the prototype work was the discovery that ordinary scan scripts (e.g. ping sweeps) can be accelerated about 40X faster using multi-threaded parallel Python (see the Appendix for source code).

If it is not used for other purposes, a military botnet hosted on government computers could be concealed until it is activated for an attack. Attribution for this type of botnet would be relatively easy to determine since it would emanate from IP addresses of government installations, all owned by the same government. In this case, clear attribution has deterrence advantages for both offensive (inhibiting state actors from attacking) and defensive (clear association with counter-retaliation). Constructing the database of domains and IP addresses for rapid attribution could be readily gleaned from public domain reconnaissance such as google hacking, whois lookups, and nslookup records.

For more typical botnets, the requirement to assign attribution within milliseconds is particularly challenging (McConnell, 2010). Attribution on the Internet, in general, is a hard problem, irrespective of the timeliness requirement. The Attribution problem, per se, is addressed in a separate research paper in preparation (Mowbray, 2010). In this gold paper, we envision mechanisms for timely attribution as a top architectural priority.
One solution approach is to reinvent the problem. Reengineering the Internet to support attribution, geo-location, intelligence, and impact analysis was proposed by McConnell, but is probably a solution that is decades away (McConnell, 2010). Also there is significant pushback to this idea based upon privacy concerns. This paper assumes that the cyber deterrence architecture will be deployed in an environment comprising current Internet technologies.

First, let us define a typical attack architecture before we propose our deterrence of it. Let’s assume our attackers control one of the thousands of botnets present on the Internet today (Zhuge et. al., 2007). We could assume that the attackers are a handful of people controlling a botnet for an organization, which may have a government affiliation. Alternatively, the attacker could be an individual. The botnet is three tiers, a few attack servers, several control servers, and bots, many of which are illegally installed on PCs unbeknownst to the PC owners (Figure 7-1). The control servers may resist detection with flux mechanisms or hiding behind a chat communication service. These are typical characteristics of today’s botnets (Bacher, Holz, Kotter & Wicherski, 2008; Stone-Gross et. al., 2009; Nazario, & Holz, 2008).
Assigning and proving attribution of a cyber incident after-the-fact without prior knowledge of the attacker is probably infeasible, given the timeliness requirement. A typical solution approach, borrowed from artificial intelligence research, is to increase the level of a priori knowledge (what our system knows before problem solving starts) until the problem is solvable by automation. Suppose that we have collected knowledge about the most likely potential attackers. There are numerous attributes that would be useful. Categories of these attributes include: personal attributes, organizational attributes, and systems attributes. To assign attribution we would need specific knowledge of the
attackers’ botnet, such as attacker IP addresses, control server domain names, server IPs, large sample of bot IP addresses, protocols used, encoding schemes, and other indicators.

If one of the nodes we are defending comes under a DDOS attack, we could correlate bot IP addresses with known attackers, and under ideal conditions, our surveillance network could monitor (or sniff) control messages and perhaps even capture some control commands from attacker systems. In that case, attribution would be assured. Note that the surveillance network would have to be somewhere on the network routes of the attacker and control servers. Ideally, the attack servers and/or the botnet control servers are monitored directly, perhaps through exploitation and installation of a rootkit (which would be allowed under our assumed legal framework outlined in Section 4). The attackers’ hosts are also ideal places to mount a counter-attack. A database of bot addresses affiliated with each botnet could be used to quickly determine the likely attack organization. Monitors on the attack or control servers could then confirm the attribution, and later be used as part of a retaliatory response.

What if the attack and control servers are hardened and cannot be exploited externally? Socially engineered emails with malicious attachments could be employed. This method of exploitation has been proven effective in all manner of high technology organizations by the Advanced Persistent Threat (Rafferty, 2010; Information Warfare Monitor, 2009). A number of rootkits are publically available from security researchers such as the Hacker Defender, Immunity’s DR Rootkit, and classics such as the LRK, Universal Root Kit and AFX Windows Rootkit (Skoudis, 2004).

Figure 7-2 is a conceptual solution architecture for rapid attribution. At the core is a blackboard knowledge base which is populated by various knowledge sources. For example, a parallel scanning tool could analyze and collect network and server information automatically. Various sensors, such as honeynets, bots, and human-supplied analyses or even partner-supplied information could be contributed to the knowledge base.
Figure 7-2. Conceptual Architecture for Rapid Attribution

An attribution engine could reason using the knowledge base, given information about an incident, including matching addresses from a black list database, confirming information using data from other knowledge sources, and returning results with an evidence-chain to human analysts.

Using address spoofing, the attacker can obscure or even falsely assign attribution. Kugler (2009) claims that for major cyber attacks, the attacker may claim attribution or attribution can be strategically inferred. However, that was the opposite of the scenario presented during the Cyber Shockwave simulation (See Section 3). Technical spoof detection approaches exist to protect DNS servers and wireless networks, for example comparing sequence numbers with previously sent packet contents, or embedding cookies in the request data. (Guo, Chen, & Chiueh, 2005) For our solution, we might consider the use and patterns of spoofing to be part of the overall signature which helps identify attackers. In addition, well distributed network sensors, might be able to correlate between spoofed packets near their origin and those involved in the attack.
The author prototyped some of the knowledge base data structures in Python. Figure 7-3 is an example of what the knowledge base might contain. Caution: These are actual examples from network scans of server IPs from the Aurora intrusions. The prototype code uses the Python dictionary type which is an association list of keys and values. An embedded association list defines the attributes of each node. With appropriate locks to manage concurrency, this structure could be used as a concurrently accessed blackboard to represent arbitrary information about networks, servers, and attackers. The knowledge base could be analyzed automatically and utilized in a variety of ways, such as an attribution determination.

RECORD: 1

{'IPv4 Address': '173.201.21.161', 'FTP Open on Port': '21', 'RDP Open on Port': '3389', 'Ping Response': 'Alive', 'Attack Organization': 'Aurora', 'Attack Role': 'Control Server'}

RECORD: 2

{'IPv4 Address': '69.164.192.46', 'Ping Response': 'Alive', 'Attack Organization': 'Aurora', 'Attack Role': 'Control Server'}

RECORD: 3

{'IPv4 Address': '168.95.1.1', 'Ping Response': 'Alive', 'Attack Organization': 'Aurora', 'Attack Role': 'Control Server'}

RECORD: 4

{'IPv4 Address': '203.69.66.1', 'Ping Response': 'Alive', 'Attack Organization': 'Aurora', 'Attack Role': 'Control Server'}

*Figure 7-3. Sample Output from Prototype Knowledge Base*

---

Some of the whois records for these IP addresses appear to be legitimate businesses who may be unaware of their alleged involvement in Aurora. These Internet addresses (and their Aurora affiliation) are in the public domain, and were downloaded from the US-CERT web site on March 13, 2010 at: [http://www.us-cert.gov/cas/techalerts/TA10-055A.html](http://www.us-cert.gov/cas/techalerts/TA10-055A.html)

Author Name, email@address
8. Conclusions and Recommendations

Cyber deterrence is a challenging problem at many levels including strategic, operational, and technical. It is a relatively unexplored domain with huge uncertainties such as the as yet non-existent legal basis for cyber war.

There is an urgent need for new treaties addressing cyber war issues. A useful precedent is the Council of Europe’s Convention on Cybercrime. Section 4 covers some of the recommended treaty provisions which would enable effective cyber operations.

The scope of a cyber deterrence architecture is large and the requirements are numerous, as indicated by the reference model detailed in Section 6. Detailing each of the capability areas is beyond the scope of one gold paper.

One of most important challenges is determining attack attribution within milliseconds. A solution architecture that leverages a priori knowledge requires details about potential attackers be gathered beforehand. This should include perhaps about the top 1000 botnets, although the potential threat is likely an order of magnitude larger. Network sensors should be placed where attribution can be rapidly confirmed. A flexible architecture involving a blackboard and diverse knowledge sources could aggregate the required information and rapidly analyze it for cyber deterrence decision support.

Multi-threading is a programming technique with useful applications in penetration testing. More than 40X speedup was measured in the prototype, comparing multi-threaded Python scans to serial-loop scans programmed in shell scripts such as ping sweeps.

Hands-on programming and benchmarking are shown in the Appendix. This research grounded the concepts in reality, and resolved major architectural unknowns. The botnet prototype in the Appendix reveals strengths and weaknesses of multi-threading and distributed processing, with important architectural insights for future military botnets.

Author Name, email@address
9. Acknowledgements

Thanks to the SANS Institute for overseeing this exciting research in the gold paper program, and to the TASC Corporation for sponsoring SANS certification training, the TASC Institute, and the NGC/TASC Cyber Warfare Community of Interest (COI). Special thanks to my gold paper advisor, Kees Leune, who contributed many ideas and abundant encouragement. I am very fortunate to have someone with his credentials, including a recent PhD in Information Security, involved in this project.

Many thanks are also due to Wellhouse Consultants for their free online Python tutorial on multi-threading (http://www.wellho.net/). The threaded code in this paper is original, but is inspired by their tutorial example, in which they also claim dramatic speedup over serial processing. The Python code for the botnet merges threading concepts with distributed processing, including public domain code adapted from Python.org online documentation (http://docs.python.org). Python has around 1500 functions in the base language and built-in libraries, and numerous add-on code libraries. Efficiently searching for these functions is a crucial skill for pen testers coding in Python. For programming in Python 2.6, this google.com hack was very helpful because it weeds out the conflicting documentation for Python 3.1 and Python 3.2:

site:docs.python.org -3.1 -3.2 <WhatImSearchingFor>

10. References


Author Name, email@address
Bipartisan Policy Center, Cyber Shockwave, a cyber ware exercise televised by CNN.com on February 20 and 21, 2010. Information retrieved on March 6, 2010 from web site: http://www.bipartisanpolicy.org/events/cyber2010


Author Name, email@address
Economic and Security Review Commission on March 13, 2009 at:


McConnell, M. (February 28, 2010). To win the cyber-war, look to the Cold War.


Author Name, email@address


Walt, S. (March 30, 2010). Is the cyber threat overblown? Foreign Policy Magazine, Washington DC: Downloaded from the Foreign Policy web site on April 10, 2010 from: http://walt.foreignpolicy.com/posts/2010/03/30/is_the_cyber_threat_overblown


11. **Appendix – Architectural Prototypes**

The following sections present some architectural prototypes for multi-threaded and botnet-like distributed scanning. A handful of performance benchmarks were measured and used to calibrate deterministic projections of botnet performance. These experiments provided some interesting insights, including the limitations of the technologies, and improved architectures for production implementations of military botnets. Python version 2.6 was used for the examples.

11.1. **Baseline Code: Threaded Scanning**

The prototypes were developed incrementally, starting with serial scanning in Linux/Unix bash shell scripting and Python. These scans were then implemented using multi-threading in Python (Figure 11.1-1). A dramatic performance increase was observed using the multi-threaded code which shows that this form of scan acceleration can have practical applications in penetration testing.

**Thread.py**

```python
#!/usr/bin/python
import os
from threading import Thread
import time
start=time.time() 
print start 

scan="ping -n 1 -w 1000 " #windows
#scan="ping -c1 -w1 " #linux/unix
#scan="nmap -A -F" 
#scan="nmap -p 22,23,35,80,445 " 
max=65

class threadclass(Thread):
    def __init__(self,ip):
        Thread.__init__(self)
        self.ip = ip
        self.status = -1
    def run(self):
        result = os.popen(scan+self.ip,"r")
        self.status=result.read()

threadlist = []
```

Author Name, email@address
for host in range(1, max):
    ip = "192.168.85." + str(host)
    current = threadclass(ip)
    threadlist.append(current)
    current.start()

for t in threadlist:
    t.join()
    print "Status from ", t.ip, ", is ", repr(t.status)

print start
print time.ctime()

Figure 11.1. Python Source Code for Fast Multi-Threaded Scanning

In the Python code for Thread.py, each parallel thread is an instance of the class threadclass. In the first for loop, the threads were initialized and given an IP address to scan, then the thread was started. The threads performed the scans in parallel on a single host machine. In the second for loop, the threads rejoined the main process, and the results were printed.

11.2. Botnet for Distributed Scanning

The following Python architectural prototype implements a simple distributed botnet, which can perform several types of parallel scans such as ping sweeps and nmap scans. The architecture is a subset of the generic botnet shown earlier in Figure 7-1, including a control server and multiple bots. There are two source code listings for this prototype, the bot code (Server.py) and the command server code (Master.py).

First, the Server.py code was run on each bot machine (Figure 11.2-2). Server.py imported a platform-specific file (one of platform_windows.py, platform_linux.py, or platform_solaris.py) which contained the Windows or Linux syntax for the scan commands (5 operating systems were tested). The Server.py showed the Python code, registered the request handler and command function, then waited for remote procedure calls (using XMLRPC) from the control server, Master.py.
**Server.py**

```python
#!/usr/bin/python
import os
from SimpleXMLRPCServer import SimpleXMLRPCServer
from SimpleXMLRPCServer import SimpleXMLRPCRequestHandler
from platform_linux import * # import botport, command syntax

class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)

server = SimpleXMLRPCServer((myhost, botport),
    requestHandler=RequestHandler)
server.register_introspection_functions()

def command(key, botop):
    output = os.popen(cmd_prefix[botop]+key+cmd_suffix[botop],"r")
    result = output.read()
    return result
server.register_function(command)

server.serve_forever()
```

*Figure 11.2-1. Python Source Code for Bots*

The Master.py Python code implemented a simple botnet control server (see figure 11.2-3). First it obtained the server proxy for each bot. Commands were sent to the bots from parallel threads running in Master.py. The spawnbot class initialized a thread with a target IP key as it was instantiated. When the thread was started, the run() function allocated a bot to the thread, then sent a remote procedure call or command()to the bot, then stored the result. The first for loop initialized and started the threads. The second for loop re-joined the threads with the main process, and the scan results were printed. Notice how two locks were used to protect the variables whichbot and results from race conditions by competing parallel threads.

**Master.py**

```python
#!/usr/bin/python
import xmlrpclib
from threading import Thread
from botnet import * # imports botop, botips, targips, locks
```
import time
start = time.ctime()
print start

bot = []
for b in botips:
    bot.append(xmlrpcclient.ServerProxy(b))

whichbot = 0
results = []

class spawnbot(Thread):
    def __init__(self, key):
        Thread.__init__(self)
        self.key = key
        self.result = False
    def run(self):
        global whichbot
        wlock.acquire()
        self.mybot = bot[whichbot]  # map task to a bot
        whichbot = (whichbot + 1) % numbots
        wlock.release()
        self.result = self.mybot.command(self.key, botop)  # do task
        rlock.acquire()
        results.append(self.result)
        rlock.release()

botthreads = []
for key in targs:
    thisthread = spawnbot(key)
    botthreads.append(thisthread)
    thisthread.start()

for b in botthreads:
    b.join()

for i in range(len(results)):
    print "\nRESULT " +str(i+1)+"\n"+repr(results[i])

print "START "+start
print "END "+time.ctime()

---

Figure 11.2-3. Python Source Code for Botnet Command Server

---

Author Name, email@address
In order to make this code run cross-platform, all platform dependencies were migrated to the Server.py-side bot code, including the scan command code. Any platform-specific post-processing of the results could be performed there as well. Note that this architecture choice is in contrast to the approach taken by the APT, whose bot code is often only stubs, devoid of command code (Rafferty, 2010). The APT bots load command code only when needed and then delete it, making it more difficult for defenders to analyze bot capabilities.

The code as shown is fairly unstable, but sufficient for our architectural benchmarking purposes. To make this code more production-ready, a throttling mechanism is needed, as well as explicit handling of failures. In experiments when each bot was issued more than \( N \) commands in rapid succession, Master.py began throwing exceptions, indicating the connection was being refused. The fast threaded code in Master.py was overrunning Python remote procedure call infrastructure with too many queued commands. The value of \( N \) varied significantly between systems, from 8 to 50 queued XMLRPC requests. Five operating system variants were tested; the \( N \) threshold appeared consistent among operating systems of the same type. When fewer commands were queued, the invocations were relatively stable. Ideally, the botnet architecture should avoid this effect entirely.

In general, any distributed processing program can experience communication errors and remote system failures. As in Google’s map reduce framework, the master should keep track of unresponsive tasks and re-issue those tasks to alternative systems as needed (Dean, & Ghemawat, 2004). The strengths and weaknesses of this distributed scanning prototype gives us the architectural insight to build a reliable implementation of a military botnet. However, developing and releasing that code into the public domain would be problematic for obvious reasons.

## 11.3. Performance Benchmarks

The Python code was tested on a prototype botnet. These measures are reused in the following sections to calibrate performance projections for military botnets.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Ping --c1 --w1</th>
<th>Exceptions</th>
<th>Nmap --p 22,23,35,80,445</th>
<th>Exceptions</th>
</tr>
</thead>
</table>

Author Name, email@address
<table>
<thead>
<tr>
<th></th>
<th>(Min:Sec)</th>
<th>(Min:Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td># Ports Scanned</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>#IPs Scanned</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>#Target Machines in Range</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Serial Python Scan</td>
<td>1:00</td>
<td>0:18</td>
</tr>
<tr>
<td></td>
<td><strong>(Bash times similar)</strong></td>
<td></td>
</tr>
<tr>
<td>Threaded Python</td>
<td>0:01</td>
<td>0:01</td>
</tr>
<tr>
<td>Botnet with 2 Bots</td>
<td>0:34</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>0:25</td>
<td>0%</td>
</tr>
<tr>
<td>Botnet with 3 Bots</td>
<td>0:26</td>
<td>6%</td>
</tr>
<tr>
<td></td>
<td>0:13</td>
<td>0%</td>
</tr>
<tr>
<td>Botnet with 4 Bots</td>
<td>0:16</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>0:07</td>
<td>0%</td>
</tr>
</tbody>
</table>

*Figure 11-4. Performance Benchmarks*

The botnet and the scan range is large enough to show advantages over a serial scan, but not large enough to beat the threaded scans, which ran surprisingly fast (Figure 11-4). A key performance factor is the contrast between heavyweight OS process creation in the serialized scan versus lightweight, highly-parallel thread creation in the threaded scan. There is also a major theoretical performance difference, which is explored in the next section.

An attempt was made to benchmark the effects of thrashing caused by a large number of threads running in parallel. The following curve resulted showing the number of threads (0..300) horizontally, and the number of seconds (0..12) shown vertically for a ping –c1 –w1 sweep (Figure 11-4). There is a distinct knee in this curve around 150 threads on a Windows Vista box. In the projections, maximum threads are limited to around 25 to throttle this effect.
11.4. Deterministic Models of Performance

Scanning performance needs to be modeled with a set of deterministic equations for time estimation.

Suppose there are N scans to be performed, each scan takes \( t(i) \) time, and there are associated overheads \( h(j) \). In these models \( h(j) \) are assumed constant, but in reality they vary with factors such as network utilization, memory utilization, process thrashing, and other factors. These other factors need to be managed and throttled in production implementations.

*Serial Scan* – In the prototype, the performance of Unix/Linux Bash shell scripts and Python serial scans is virtually identical.

\[
T(s) = \sum_{i=1}^{N} t(i) + N \cdot h(s) + h(r)
\]

where \( h(s) \) is per serial scan overhead and \( h(r) \) is final reporting overhead.

*Parallel (Threaded) Scan* – This models the threaded scan code shown in Thread.py.

\[
T(p) = \max_{i=1..N} t(i) + N \cdot h(p) + h(r)
\]
where \( h(p) \) is per threaded scan overhead. Note how the serial equation is Order(N), scaling with the number of scans, and the parallel equation is Order (1), or constant time depending only upon the longest task.\(^9\)

**Distributed Serial Scan** – This models the botnet implemented with Server.py and Master.py.

\[
T(ds) = \max(\sum(t(i)) \text{ from } i=k*m+1..(k+1)*m + m*h(s)) \text{ from } i=1..N + (N/m)*h(d) + h(r)
\]

where \( m \) is the number of scans per batch per bot, \( k \) is the scan batch size, and \( h(d) \) is per batch overhead, e.g. distributed messaging, application code, IO, and operating system. The equation is proportional to Order (N/m).

**Distributed Parallel (Threaded) Scan** – This is a theoretical botnet we have not implemented, but may represent some of best architectural choices by moving the threading to the bots and using master code that manages throttles and bot failures.

\[
T(dp) = \max(\max(t(i)+m*h(p))) + (N/m)*h(d) + h(r)
\]

where \( m \) is the number of scans per batch per bot, \( k \) is the scan batch, and \( h(d) \) is per distributed command overhead. The searching tasks in this equation are Order(1) in this equation, but the overhead scales by Order(N/m). This predicts that the overhead becomes the dominant performance factor when we perform a large number of scans in parallel.

There are a number of other performance factors which are not modeled here, but must be accounted for in the implementation, such as limitations on threading, botnet size, network congestion, operating system performance, memory utilization and so forth. These equations must be realized in a spreadsheet, then the variables must be adjusted to fit the benchmarks developed in section 11.3.

### 11.5. Projections for Military Botnets

The following table shows some performance projections for N scan tasks using the different scan architectures modeled in Section 11.4. The constants were adjusted

---

\(^9\) In theory, assuming unlimited threads. We discussed the thrashing effects of threading in the previous subsection.

Author Name, email@address
until the numbers matched the empirical benchmarks in Section 11.3 for ping sweep. This data is highly extrapolated for large N and should be verified empirically in future research. Acquiring the required capabilities is beyond the scope of this gold paper.

<table>
<thead>
<tr>
<th>Constants</th>
<th>Fitted</th>
<th>N</th>
<th>Serial</th>
<th>Parallel</th>
<th>Dist Serial</th>
<th>Bots</th>
<th>Dist Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>t(i)</td>
<td>0.4</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h(s)</td>
<td>0.52</td>
<td>10</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h(r)</td>
<td>0.1</td>
<td>20</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h(p)</td>
<td>0.01</td>
<td>30</td>
<td>28</td>
<td>1</td>
<td>25</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>h(d)</td>
<td>1</td>
<td>40</td>
<td>37</td>
<td>1</td>
<td>25</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>m</td>
<td>25</td>
<td>50</td>
<td>46</td>
<td>1</td>
<td>25</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
<td>60</td>
<td>2</td>
<td>26</td>
<td>3</td>
<td>26</td>
</tr>
<tr>
<td>DS Bots</td>
<td></td>
<td>100</td>
<td>92</td>
<td>3</td>
<td>27</td>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>33.3</td>
<td>200</td>
<td>184</td>
<td>5</td>
<td>31</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>26</td>
<td>500</td>
<td>460</td>
<td>thrashing</td>
<td>42</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>16.6</td>
<td>1,000</td>
<td>920</td>
<td></td>
<td>60</td>
<td>37</td>
</tr>
<tr>
<td>accel factor</td>
<td>85%</td>
<td>2,000</td>
<td>1,840</td>
<td></td>
<td>97</td>
<td>74</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5,000</td>
<td>4,600</td>
<td></td>
<td>208</td>
<td>185</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10,000</td>
<td>9,200</td>
<td></td>
<td>392</td>
<td>369</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100,000</td>
<td>92,000</td>
<td></td>
<td>3,704</td>
<td>3,681</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500,000</td>
<td>460,000</td>
<td></td>
<td>18,424</td>
<td>18,401</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,000,000</td>
<td>920,000</td>
<td></td>
<td>36,824</td>
<td>36,801</td>
<td>394</td>
</tr>
</tbody>
</table>

*Figure 11-5. Performance Projections for Military Botnet*

The number of bots in the main table was calculated as ceiling (N/m), equivalent to the number of scan task batches of size m searches. There are known botnets (such as Storm Worm) in the wild that exceed the proposed scales (Holz, 2008). This column is shared by both the distributed serial and distributed parallel botnet architectures. In the prototype botnet, the distributed serial (DS) bots varied in number from 2 to 4. To fit the prototype measurements an acceleration factor was added that more correctly matches the estimated times (distributed speedup) with the actual measurements.

Adjusting the scan time (t(i)) we see how the projections behave for different types of scans. For example, in experiments nmap –A (all) port scans on an active host can take as long as 15 or more seconds. The table projects that single-host, multi-threaded scans are a fast option for scans up to a hundred or so addresses. Depending on how

Author Name, email@address
many bots are fielded, and other factors such as network contention, the distributed parallel botnet may still run fast even for very large scans.