Chapter 3
Adversarial Dynamics: The Conficker Case Study

Daniel Bilar, George Cybenko and John Murphy

Abstract It is well known that computer and network security is an adversarial challenge. Attackers develop exploits and defenders respond to them through updates, service packs or other defensive measures. In non-adversarial situations, such as automobile safety, advances on one side are not countered by the other side and so progress can be demonstrated over time. In adversarial situations, advances by one side are countered by the other and so oscillatory performance typically emerges. This paper contains a detailed study of the coevolution of the Conficker Worm and associated defenses against it. It demonstrates, in concrete terms, that attackers and defenders each present *moving targets* to the other. After detailing specific adaptations of attackers and defenders in the context of Conficker and its variants, we briefly develop a quantitative model for explaining the coevolution based on what what we call *Quantitative Attack Graphs* (QAG) which involve attackers selecting shortest paths through an attack graph with defenders investing in hardening the shortest path edges appropriately.

Daniel Bilar
Process Query Systems LLC, 16 Cavendish Ct., Lebanon NH 03766
Siege Technologies, 33 S Commercial St., Manchester NH 03101 e-mail: dbilar@acm.org

John Murphy
Process Query Systems LLC, 16 Cavendish Ct., Lebanon NH 03766
e-mail: jmurphy@flowtraq.com

George Cybenko
Process Query Systems LLC, 16 Cavendish Ct., Lebanon NH 03766
Dartmouth College, Hanover NH 03750
e-mail: gvc@flowtraq.com, gvc@dartmouth.edu

3.1 Introduction

Progress in operational cyber security has been difficult to demonstrate. In spite of the research and development investments made over more than 30 years, many government, commercial and consumer information systems continue to be successfully attacked and exploited on a routine basis. By contrast, research and development investments in automobile, rail and aviation safety over the same time periods have led to significant, demonstrable improvements in the corresponding domains.

Advances in standard performance measures for automobile, train and airline transportation (namely fatalities per unit of travel) are depicted at the top of Fig. 3.1, while corresponding measures for cyber security are depicted at the bottom.

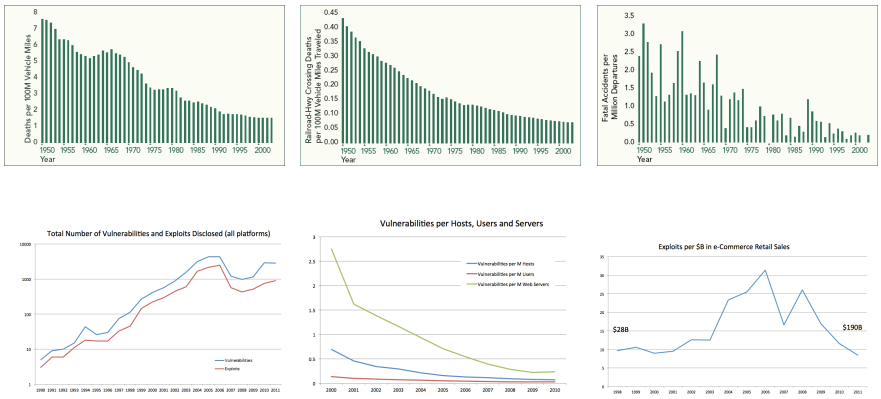

Fig. 3.1: Top Left - TRAFFIC FATALITY RATES: U.S. Motor Vehicle Fatalities per 100 Million Vehicle Miles, 1950-2003. (Source: National Highway Traffic Safety Administration). Top Middle - RAILROAD-HIGHWAY CROSSING FATALITY RATES: U.S. Railroad- Highway Crossing Fatalities per 100 Million Vehicle Miles, 1950-2003. (Source: Federal Railroad Administration, Bureau of Transportation Statistics). Top Right - AVIATION FATALITY RATES: Fatal accidents per million departures for U.S. scheduled service airlines, 1950-2003. Accidents due to sabotage or terrorism are not included. (Source: Air Transport Association). Bottom Left - VULNERABILITY AND EXPLOIT COUNTS: Total Number of Vulnerabilities and Exploits (Graph produced by P. Sweeney from data in OSVDB). Bottom Middle - VULNERABILITIES PER HOSTS/USERS/SERVERS: Total Number of Vulnerabilities normalized by internet hosts, users and servers (Graph produced by P. Sweeney from data in OSVDB, Internet Systems Consortium (number of hosts on the Internet), Netcraft webserver survey (number of webservers on the Internet), and Internet World Stats (number of Internet users).). Bottom Right - EXPLOITS NORMALIZED BY ECOMMERCE: Total number of exploits per billion dollars of e-commerce (Graph produced by P. Sweeney from data in OSVDB and www.census.gov/estats)

A major difference between automobile safety and information security is that in the former the adversaries are natural laws that don't change, while in the latter the adversaries are rational humans who adapt quickly and creatively. Consequently, we argue that we cannot understand or model the cyber security landscape in terms of steadily making progress towards an asymptotic "solution" as the transportation statistics suggest is happening in that domain.

In Fig. 3.1, the bottom row of plots show, from left to right, vulnerabilities and/or exploits in absolute numbers (bottom left), normalized by the estimated number of hosts, users and servers (bottom middle) and normalized by the estimated amount of e-commerce (bottom right). The total number of reported vulnerabilities and exploits is growing at first and leveling off (note that this is a logarithmic plot), with some noticeable oscillations especially in recent years; however, the trend is most meaningful if normalized by some measure of corresponding "activity." For example, traffic fatality statistics are routinely normalized by vehicle miles travelled or aircraft takeoffs.

We have not settled the matter about what the correct or analogous normalization for cyber vulnerabilities and exploits would be. If we were to normalize by the number of different operating system platforms for example, the plot would basically resemble the bottom left because there simply are not that many different platforms available in the market. If we normalize by the total number of users, hosts or servers, there is a precipitous drop in vulnerabilities as the bottom middle plot shows. However, if we normalize by e-commerce "transactions" as measured by estimated total e-commerce, the bottom right plot in Fig. 3.1 shows major oscillations without an obvious extrapolation into the future.

The point is that unlike domains in which we can measure progress against a stationary environment, cyber security must be viewed as an ongoing sequence of moves, countermoves, deceptions and strategic adaptations by the various actors involved - attackers, defenders, vendors and decision/policy makers. Accordingly, we believe that the appropriate science for understanding the evolving landscape of cyber security is not the logic of formal systems or new software engineering techniques. Instead, it is an emerging subarea of game theory that investigates dynamics in adversarial situations and the biases of competing human agents that drive those dynamics (see [9, 23, 8] for example).

3.1.1 Adversarial Behavior Analytics vs Classical Game Theory "Solutions"

The original goals of Game Theory were to model adversarial environments and to optimize strategies for operating in those environments. This would seem ideal for modeling cyber operations as well as other national security situations - indeed, there is a community of researchers currently investigating the application of classical Game Theory to information assurance and cyber operations.

However, the overwhelming focus of Game Theory research over the past 60 years has been on the problem of "solving" games that are defined *a priori*. That is, most Game Theory research to date begins by assuming a game is already defined (namely, the players, their possible moves and payoffs) and then explores properties of optimal strategies and how to compute them. Optimality is with respect to a solution criterion such as Nash Equilibrium or Pareto Optimality [5].

An obvious and growing criticism of the classical approach is that in most real world adversarial situations players do not know who the other players are, what their possible moves might be and, perhaps most importantly, what their preferred outcomes or objectives are. Put another way, none of the players actually know the complete details of the game that they are playing! A further complication is that few people outside of the Game Theory literati know what a Nash Equilibrium is, let alone how to compute one, so they typically cannot be expected to play the Nash solution.

As a result, while Game Theory can inform us about how to play chess, checkers, poker and simple illustrative examples found in most Game Theory texts, it has not been as useful in the majority of real-world adversarial situations as one might have hoped for (see [19, 20] for an interesting discussion). New directions and ideas are needed, especially in the area of cyber security.

3.1.2 Our Approach

Adversarial behavior analytics is the empirical study of players' actions in adversarial situations. The "game" in these adversarial situations is implicit and can only be understood in terms of the moves players make and how they evolve their play in response to each other's moves [12].

We have studied historical data from a variety of cyber and national security domains such as computer vulnerability databases, offensive and defensive coevolution of wormbots such as Conficker, and US border security [22, 24]. The data show that the "success rate" or other performance metrics in these different domains oscillate over time - they are not converging to any asymptote. In fact, when players are continually responding as best they can to their opponents' play, periodic and even chaotic behaviors can be exhibited [23].

Such oscillations are indicators of and intrinsic to adversarial dynamics in complex, competitive environments. In particular, each player is adapting incrementally to the observed play of his or her opponents. This can be modeled by systems of differential equations known as *replicator equations* [9, 22].

The replicator equations are typically third-degree nonlinear so that the resulting dynamics are difficult to predict analytically. However, the inverse problem of observing behaviors and estimating parameters of the replicator equations that result in those behaviors are tractable computational problems. In particular, it is possible to observe game play and strategy evolution and then make inferences about the players' motives, costs and move options.

This kind of modeling approach can explain the non-convergent dynamics we are seeing in cyber security and will help us forecast the various players' future strategies. Recognizing and harnessing the realities of such dynamic coevolution will be a key ingredient to dominating cyber operations.

3.1.3 Organization of the Paper

After this introduction, we examine in detail the documented structure of the Conficker Worm and defenses against it. This is, to our knowledge, the first quantitative attempt to extract and highlight specific adaptations in an adversarial setting. Assessment of these moves in the context of classical solution concepts such as Nash Equilibria is then discussed and is followed by an analysis of the estimated goals and motives of Conficker's developers. We then develop the notion of a Quantitative Attack Graph (QAG) and present some generic analyses of the adversarial nature of that model.

3.2 Conficker Analysis

Drawing on published sources [15][3][16][21][7], we model the interactions between Conficker (specifically its spread and update mechanisms) and its "Ecosystem" (i.e., the networked computing substrate it operates on: Microsoft, the Internet Infrastructure, the worm analysis community) as an adversarial game between two players.

3.2.1 Conficker Internal and External State Diagram

We first analyzed Conficker's internal state diagrams in terms of armoring, update and scan/infect mechanisms. The goal was to identify vulnerable points to disable Conficker (Fig. 3.2).

One area we identified was environmental mutations. Conficker A exited upon detection of a Ukrainian keyboard locale. Conficker C's well thought out innovation, its P2P module, kills Conficker C if a debugger is detected (Fig. 3.3). We also found that manipulating the Random Number Generator affects the scan/infected IP range and the Domain Generation Algorithms IP rendezvous points for potential updates (Fig. 3.4). Subversion of software/hardware encryption/hashing functionality by triggering on the public key decryption (the RSA public key is known) disables the install of new binaries (Fig. 3.5). Finally, manipulation of elapsed time/tick count (through memory writes and/or direct clock influence) affects state transitions in all mechanisms (Fig. 3.6).

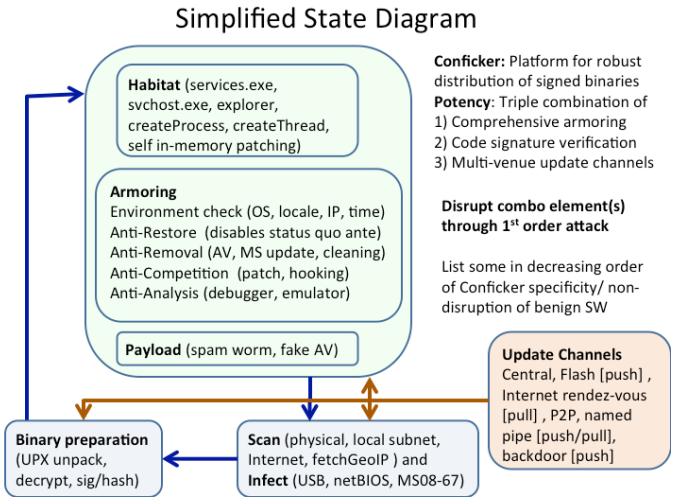

Fig. 3.2: Conficker's armoring (green), update (red) and scan/infect (blue) mechanisms

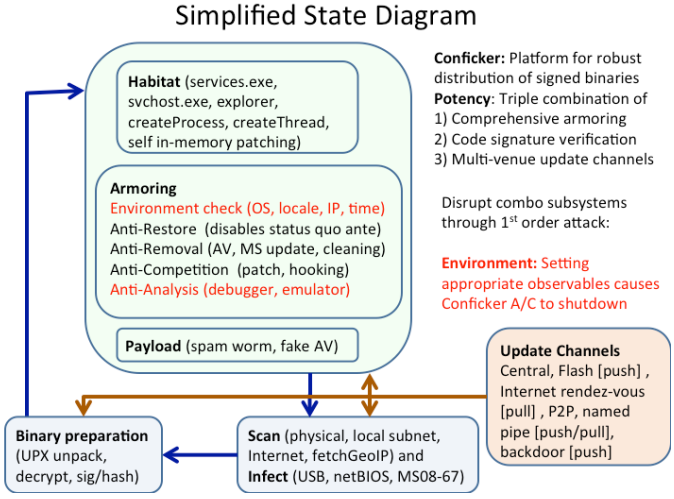

Fig. 3.3: Setting environmental observables (debugger present, VM environment, keyboard locale) causes shutdown

Also of interest is the susceptible host population view, representing the external state transitions. Fig. 3.7 shows the migration chart of the Conficker variants, while

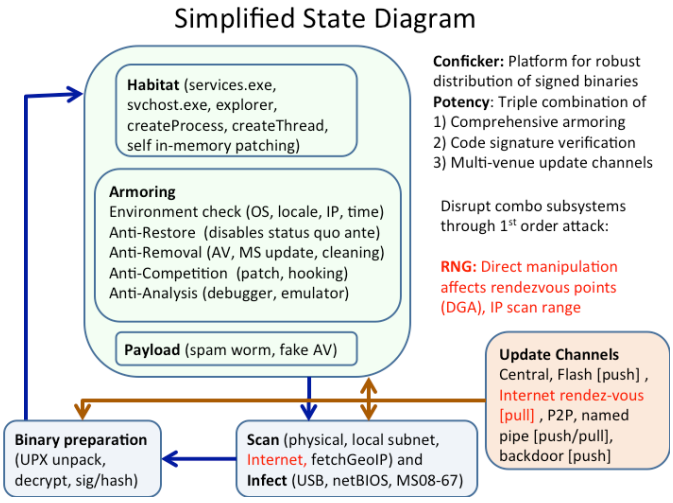

Fig. 3.4: Manipulating the Random Number Generator affects the scan/infected IP range and the Domain Generation Algorithms IP rendezvous points

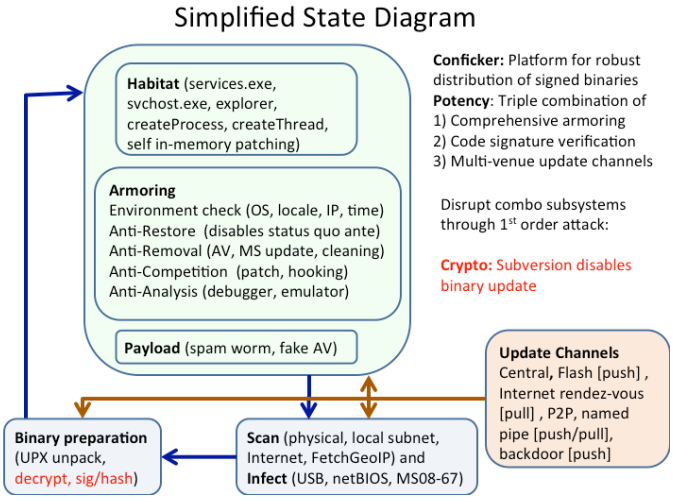

Fig. 3.5: Subversion of encryption functionality disables the install of new binaries

Fig. 3.8 shows the individual Conficker A/B/C/D/E host state changes. For a general discussion on subverting end systems through subsystems, see [1].

Lack of a common naming scheme for Conficker and disagreement among analysts which release constitute new versions complicate matters somewhat. For ex-

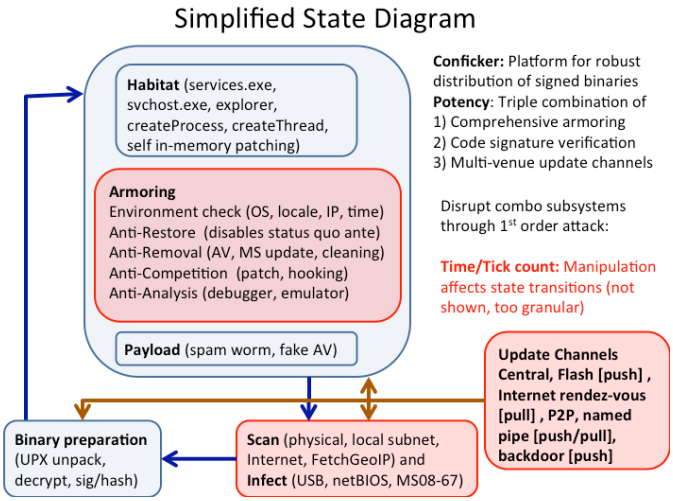

Fig. 3.6: Control of elapsed time/tick count affects state transitions in all mechanisms

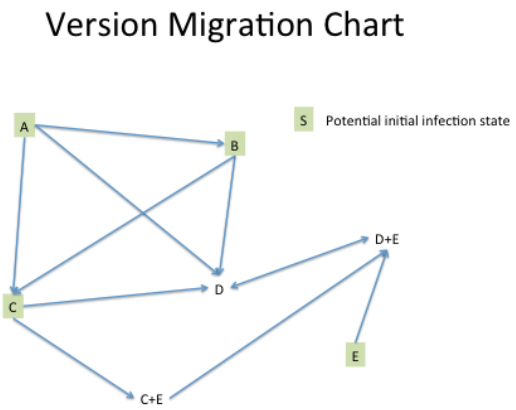

Fig. 3.7: Transitions and updates to Conficker variants

ample, the third release (Microsoft's Conficker.C) is recognized as only incremental by the SRI-based Conficker Working Group (CWG), and is not recognized at all by Symantec.

Table 3.1 shows the names currently used by each group. While we have not seen evidence that this naming confusion had any measurable effect on the ability

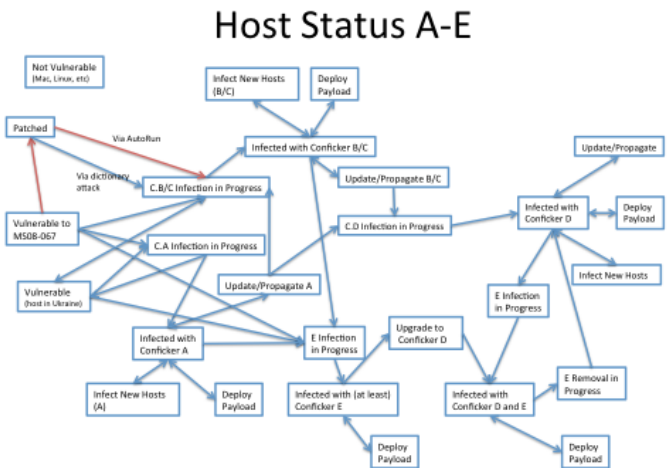

Fig. 3.8: Variant transitions and external state diagram

to defend against Conficker, it was a confounding factor in researching Conficker. We use the Microsoft nomenclature in this report, except where noted.

Table 3.1: Conficker naming conventions according to Microsoft, SRI and Symantec

Microsoft	SRI	Symantec
Conficker.A	Conficker.A	W32.Downadup
Conficker.B	Conficker.B	W32.Downadup.B
Conficker.C	Conficker.B++	
Conficker.D	Conficker.C	W32.Downadup.C
Conficker.E	Conficker.E	W32.Downadup.E

3.2.2 Time-evolution of Conficker

Two timelines are involved, Conficker's and the Ecosystem. Within those timelines, we distinguish among three epochs with corresponding time regimes: The era before the first appearance of Conficker A ("BCA"), followed by the emergence of the Conficker timeline ("ACA"). The final post-Conficker E ("PCE") epoch is not analyzed further in this report.

In BCA, the necessary Ecosystem pre-conditions for the viability/emergence of Conficker have to be met. These pre-conditions include "long reach propagation", a long range internet accessible vulnerability (in Conficker's case: MS08-067 RPC-DCOM), and "weaponization", an exploit for that vulnerability in a form that can be integrated into a worm (in Conficker's case: $37.80 Chinese-made exploit kit for RPC-DCOM) [17].

As a simplifying abstraction, we view the developments in the BCA era as Ecosystem configuration fluctuations, rather than moves in a game. Once a Conficker-amenable configuration arises (i.e a worm-integratable exploit for long range vulnerability), a race-to-market begins between attacker and defender to plug the security "hole". This typically takes the form of vendor software patches to fix competing with a worm exemplar to exploit the vulnerability. Events unfolding under this time regime can be modeled as a multi-objective optimization problem, balancing product performance (the "quality" of both the patch and the worm) and speed-to-market (who gets to the security hole first).

Conficker A appeared on Nov. 20, 2008. From then onwards (the ACA era), we start interpreting measures by the Ecosystem and Conficker A/B/C/D/E code evolution as moves and counter-moves in an adversarial game. We illustrate this with Fig. 3.9 below.

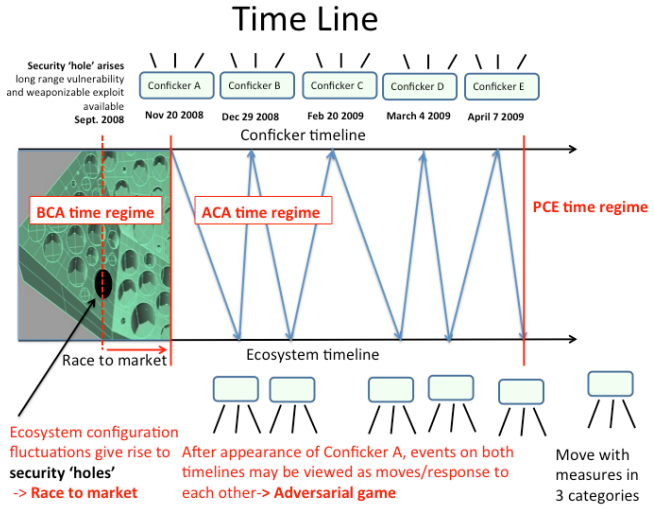

Fig. 3.9: Conficker and Ecosystem timelines. We distinguish among three epochs, but only the ACA time regime is analyzed in this paper. Race-to-market begins with an Ecosystem configuration that includes a long-range vulnerability and a concomitant worm-integratable exploit.

3.2.3 Game Moves

The game consists of sequential moves between Conficker and the Ecosystem. We fix an adversarial game with 5 rounds, consisting of 5 attacker moves and 5 responses. Moves consist of one or more measures, which are further grouped into four categories. Each Conficker move consists of the measures implemented by its respective Conficker variant (A/B/C/D/E). The measures that the Ecosystem implemented chronologically after Conficker's move are treated as the Ecosystem's response.

We group measures into Spread/Infect if they affect Conficker's spread and infection mechanisms in the wider sense. Similarly, measures that affect Conficker's update mechanism are grouped under Update, and anti-analysis/self-defense measures under Armor. Measures that do not fit into prior categories are grouped under Other.

A count of measures is given below in Table 3.2. In game-theoretic nomenclature, measures represent strategies. In each of the 5 rounds, players can select one or more strategies per category. For example, in the Spread/Infect category, Conficker E can maximally choose from the power set of 6 pure strategies (i.e 2^6=64 possibilities), while the Ecosystem can maximally choose from the power set of 8 pure strategies (i.e. 2^8=256 possibilities).

Table 3.2: Maximum pure strategy count in each category

Player	Spread/Infect	Update	Armor
Conficker	6	12	7
Ecosystem	8	2	2

However, chronology excludes some possibilities: For instance, the Ecosystem cannot use strategy `AVsigD` as a response to Conficker A, since that anti-viral signature for Conficker D could not yet have been developed. The actual strategies chosen in each round are given in Table 3.3 (new measures are **bolded**, dropped measures are ~~struck-through~~). For example, Conficker chose three out of six pure strategies `MS08-067` and `EnvCheck, FetchGeoIP` for Spread/Infect, and Ecosystem responded with three strategies `MSpatch, AVsigA, DenyGeoIP` out of 8 available pure strategies.

3.3 Nash Equilibrium or Myopic Best Response?

In Game Theory, a Nash Equilibrium arises when, in full knowledge of the other player's strategy, a player cannot benefit by changing his own strategy[1]. The notion of "benefit" is captured with payoffs for each strategy; these payoffs are in turn related to the goals of the players. To determine whether the actual strategies chosen in each game round between Conficker and the Ecosystem could be interpreted as achieving Nash Equilibria (given plausible payoff/utility functions), we examine the moves between the Ecosystem and Conficker A in the category Spread/Infect.

Conficker's pure strategies are `EnvCheck`, `MS08-067`, `USB`, `LocalShare`, `FetchGeoIP`, `InclGeoIP` and the Ecosystem's available pure strategies are `MSpatch`, `AVsigA`, `FakeGeoIP`, `DenyGeoIP`. Conficker implemented `MS08-067`, `EnvCheck`, `FetchGeoIP` and the Ecosystem responded with `MSpatch`, `AVsigA`, `DenyGeoIP`.

For Spread/Infect, a reasonable goal/motive for Conficker is to increase the number of infected hosts. A reasonable goal for the Ecosystem player is to reduce the vulnerable host population. A Nash Equilibrium would imply that exploiting the MS08-067 vulnerability in conjunction with the Ukraine keyboard locale check and checking IP location through the GeoIP mapping at maxmind.net gives the best payoff for Conficker in terms of increasing number of infected hosts. Conversely, issuing the MS08-067 patch, moving the web address of the GeoIP database and adding Conficker A anti-virus signature constitutes the best response to reduce the vulnerable host population. However, when Conficker A appeared in Nov/Dec 2008, a much more effective Ecosystem move to stem initial spread and contain the number of infected hosts would have been to replace the maxmind.net GeoIP lookup with a fake database (which was done months later, in June 2009). In addition, a non-realized measure such as ingress filtering of RPC TCP port 445 communications would have constituted an effective mitigation strategy.

A retrospective analysis of actual moves by Conficker and the Ecosystem suggests that they do not compute Nash Equilibria over strategy sets. We surmise this is due to the size of the strategy power sets and the incomplete information nature of the game. Instead, the players respond myopically with perceived best responses to the situation at each time step. Furthermore, that analytical framework assumes both players have the same model of the game, which may not be true.

3.3.1 Example of a Myopic Attacker Move

Conficker B introduced two new methods of self-propagation, apparently aimed at accessing networks or portions of networks not available to the randomized long

[1] This implies a Nash Equilibrium test: If, after revealing the player's strategies to one another, no player changes his strategy, despite knowing the actions of his opponents, a Nash Equilibrium has been reached.

range IP address vulnerability used by Conficker A. The first new method used the local area network's LocalShare, so that an infected host could in turn infect those of its local peers with insecure passwords. The second new method was to have infected hosts attempt to infect removable media such as USB keys with an attack vector via the Windows AutoRun feature.

In retrospect, the addition of the AutoRun propagation produced mixed results. As an attack vector for otherwise inaccessible systems, it proved effective: in 2009 there was not as strong a sense of security with respect to these devices, and USB drives were frequently used to bridge air gaps. However, it is not clear that bridging those air gaps was a useful thing in building a platform for releasing signed binaries, since the hosts on the other side of that gap could be infected but could not necessary access the Internet to be updated or download payloads. In Microsoft's survey of Conficker propagation methods as a percent of attempted attacks [13], AutoRun was implicated in only 6% of attempts, suggesting that the addition of this vector was ultimately of limited utility.

Moreover, the USB drive vector was implicated in early 2009 in a string of high-profile infections such as the city of Manchester, UK and the French military. These attacks spurred greatly increased media scrutiny of Conficker, and in turn appear to have lead to accelerated adoption of anti-Conficker measures on the local level. We observed a similar increase in media interest and subsequent awareness when in 2009/2010 the Stuxnet worm spread to targets outside of Iran [11]. However, whereas Conficker's USB drive vector was in all likelihood intentionally added, it cannot be conclusively determined whether the Stuxnet leak was intentional or inadvertent.

3.3.2 Example of a Myopic Defense Move

One of the earliest moves made by the defenders in response to Conficker A was to interrupt the ability of infected hosts to retrieve the GeoIP database file from the domain maxmind.net. As part of its scanning and propagation routine, Conficker A generated a randomized list of IP addresses and checked each one against the GeoIP database file to see whether it was in Ukraine or not.

The GeoIP database file was moved by the administrators of maxmind.net to another URL shortly after the release of Conficker A. The effect of this move is difficult to gauge, as there was not an explicit study done at the time. However, experts believed that it could have slowed the propagation of the worm, and the next version of Conficker (B) included the GeoIP database along with the worm itself, even though the new location was well-known.

One of the hallmarks of the Conficker attack has been a heterogeneous botnet in which multiple older versions coexist alongside the most recent. This setup facilitated the use of a later move by Felix Leder and Tillmann Werner at the Honeynet Project to quantify the effect of this specific counterattack. Hosts infected with Conficker A that had not been upgraded to later versions were still, as of June 2009,

contacting maxmind.net requesting the GeoIP file that had been taken down. Leder and Werner approached maxmind.net and had them substitute a specially prepared GeoIP file that listed the entire Internet as being in Ukraine, and then tracked the number of unique IP addresses requesting that file as an indirect way of measuring new infections.

Leder and Werner found that after the substitution (at a date in early July that they did not specify), the number of unique IP addresses contacting maxmind.net dropped precipitously, as shown in Fig. 3.10. While this move was not made during the original back-and-forth, it is suggestive that the original move may have been effective in reducing the number of new infections.

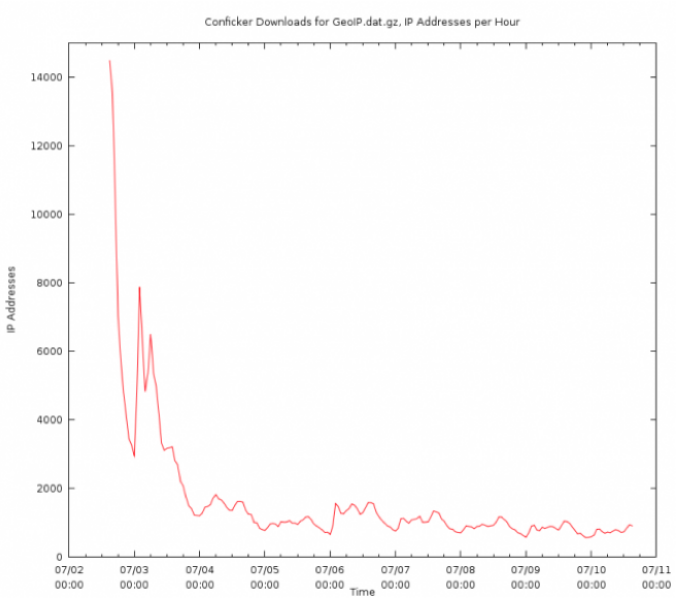

Fig. 3.10: Unique IP addresses requesting GeoIP database as a proxy for new infections after poisoning DB. (Source: Honeynet Project, `https://honeynet.org/node/462`)

It also serves to illustrate why the original GeoIP use constituted a myopic move on the part of the Conficker creators. By using an external resource not under their control, Conficker's authors empowered the defenders with the following means:

1. An attack vector against data used for propagation;
2. A means of directly tracking new infections and thus evaluating the efficacy of successive moves.

On this occasion, this opportunity was partly missed by the defenders, who quickly found other means of attacking Conficker and who had other means of indirectly

tracking its spread. However, this example is also illustrative of the kind of move that defenders need to be making, in that it not only affected the adversary, but also included a mechanism for tracking its effectiveness.

Table 3.3 outlines a shorthand list of Conficker and Ecosystem measures and a description of shorthand measures is given in Table 3.4.

We illustrate in Figs. 3.11-3.15. by means of internal state diagrams the measures that Conficker A/B/C/D/E adopted.

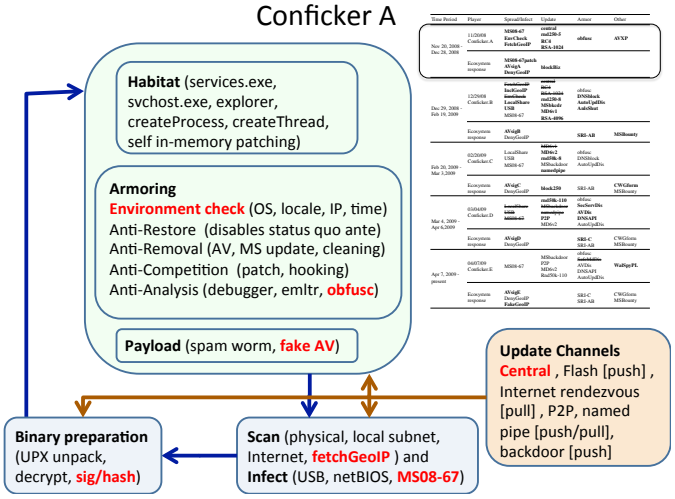

Fig. 3.11: Conficker A. Biggest Achilles heel is centralization.

3.4 Analysis of Conficker Goals/Motives

As of February 2012, we think that the most likely goal of Conficker's creator(s) is the creation and maintenance of a large-scale reliable platform for crimeware distribution. This is corroborated by the payloads that Conficker A and E installed, as well as developments that transpired in June 2011: A group in the Ukraine was arrested for using Conficker to distribute phishing payloads to banks. The most sophisticated botnet in the world today, the 4th generation TDSS/TDL4 [18][6], seems to be geared towards the installation of adware/spyware and spam - in other words, crimeware.

We stress that although the binary payloads so far have been crimeware, the design of Conficker enables deployment and execution of arbitrary signed binary pay-

Table 3.3: Measures implemented by Conficker and Ecosystem between November 2008 and April/May 2009. **Bolded** indicates newly introduced measures. ~~Strike-through~~ indicates dropped measures

Time Period	Player	Spread/Infect	Update	Armor	Other
Nov 20, 2008 - Dec 28, 2008	11/20/08 Conficker.A	**MS08-067** **EnvCheck** **FetchGeoIP**	**central** **rnd250-5** **RC4** **RSA-1024**	**obfusc**	**AVXP**
	Ecosystem response	**AVsigA** **DenyGeoIP** **MSpatch**	**blockBiz**		
Dec 29, 2008 - Feb 19, 2009	12/29/08 Conficker.B	~~FetchGeoIP~~ **InclGeoIP** ~~EnvCheck~~ **LocalShare** **USB** MS08-067	~~central~~ ~~RC4~~ ~~RSA-1024~~ **rnd250-8** **MSbkcdr** **MD6v1** **RSA-4096**	obfusc **DNSblock** **AutoUpdDis** **AnlsShut**	
	Ecosystem response	**AVsigB** DenyGeoIP		**SRI-AB**	**MSBounty**
Feb 20, 2009 - Mar 3, 2009	02/20/09 Conficker.C	LocalShare USB MS08-067	~~MD6v1~~ **MD6v2** **rnd50k-8** MSbackdoor **namedpipe**	obfusc DNSblock AutoUpdDis	
	Ecosystem response	**AVsigC** DenyGeoIP	**block250**	SRI-AB	**CWGform** MSBounty
Mar 4, 2009 - Apr 6, 2009	03/04/09 Conficker.D	~~LocalShare~~ ~~USB~~ ~~MS08-067~~	**rnd50k-110** ~~MSbackdoor~~ ~~namedpipe~~ **P2P** MD6v2	obfusc **SecServDis** **AVDis** **DNSAPI** AutoUpdDis	
	Ecosystem response	**AVsigD** DenyGeoIP		**SRI-C** SRI-AB	CWGform MSBounty
Apr 7, 2009 - present	04/07/09 Conficker.E	MS08-067	MSbackdoor P2P MD6v2 Rnd50k-110	obfusc ~~SafeMdDis~~ AVDis DNSAPI AutoUpdDis	**WalSpyPL**
	Ecosystem response	**AVsigE** DenyGeoIP **FakeGeoIP**		SRI-C SRI-AB	CWGform MSBounty

Table 3.4: Description of Conficker and Ecosystem measures between Nov 20, 2008 and April/May 2009

Category	Player	Shorthand	Description
Spread/Infect	Conficker	MS08-067	Internet-accessible RPC vulnerability
		LocalShare	Local subnet Windows share drives
		USB	Local physical USB drives
		FetchGeoIP	Fetch GeoIP of IP to physical locations
		InclGeoIP	Embed GeoIP gzip file in Conficker
	Ecosystem	DenyGeoIP	Move GeoIP file to different location
		FakeGeoIP	Map all IP to Ukraine locations
		MSpatch	Software patch for MS08-067
Update	Conficker	central	Pull from trafficconverter.biz
		rnd250-5	Pull from 250 rnd domains in 5 TLDs
		rnd250-8	Pull from 250 rnd domains in 8 TLDs
		MSbckdr	Backdoor patch for MS08-067
		Rnd50k-8	Pull from 50k rnd domains in 8 TLDs
		namedpipe	Download URL transmitted to pipe
		Rnd50k-110	Pull from 50k rnd domains in 110 TLDs
		RC4	Conficker RC4 encryption
		RSA-1024	Conficker 1024 public RSA key
		RSA-4096	Conficker 4096 public RSA key
		MD6v1	first version MD6 hash implementation
		MD6v2	patched MD6 hash implementation
	Ecosystem	blockBiz	Take down trafficconverter.biz
		block250	Register all 250-5 and 250-8 domains
Armor	Conficker	DNSBlock	Block DNS lookups
		AutoUpdDis	Disable MS AutoUpdate
		SafeMdDis	Disable Windows Security Services
		AVDis	Disable AV processes
		EnvCheck	Check environmental parameters
		AnlsShut	Anti-Analysis mechanisms
		obfusc	Code obfuscation
	Ecosystem	AVsigA	Anti-virus signature for Conficker A
		AVsigB	Anti-virus signature for Conficker B
		AVsigC	Anti-virus signature for Conficker C
		AVsigD	Anti-virus signature for Conficker D
		AVsigE	Anti-virus signature for Conficker E
Other	Conficker	AVXPPL	Fake anti-virus XP payload
		CWGform	Conficker Working Group forms
		WalSpyPL	Waledac/SpyProtect payload
	Ecosystem	SRI-AB	SRI report on Conficker A/B
		SRI-C	ASRI report on Conficker A/B
		MSBounty	Microsoft $250,000 reward

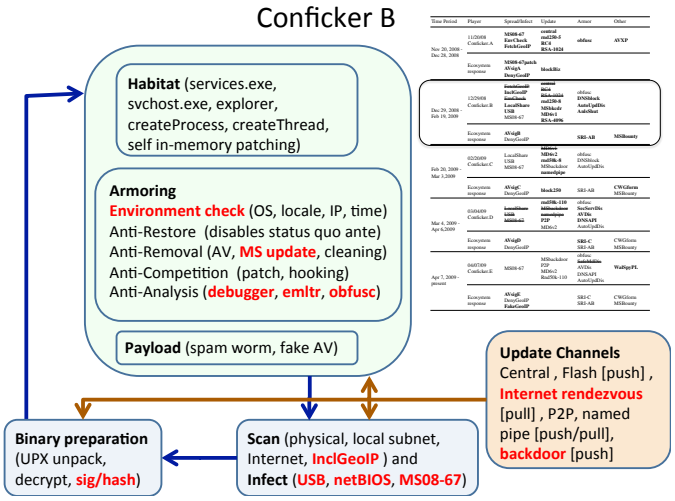

Fig. 3.12: Conficker B: Diversification of update channels and infection vectors

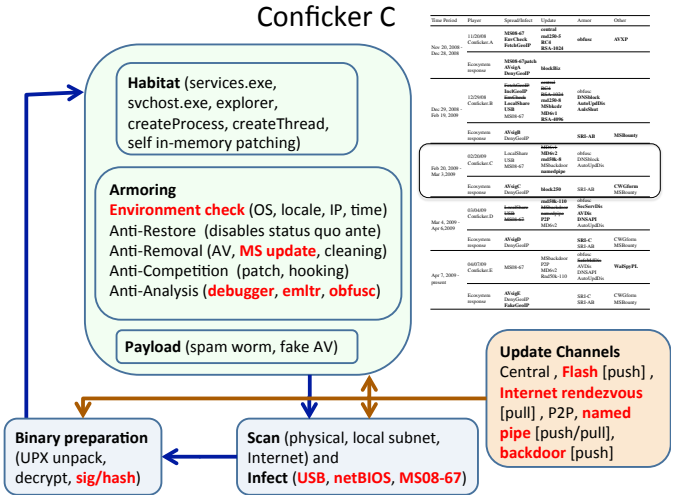

Fig. 3.13: Conficker C: More update channels, MD6 fix and extensive anti-analysis measures

loads which could be used for sabotage, DDoS attacks, data destruction, intellectual property theft - virtually any payload, provided it is signed.

As a secondary plausible goal, it seems the creator(s) of Conficker wanted to preserve anonymity. Denial of attribution is the normal state of affairs for worms

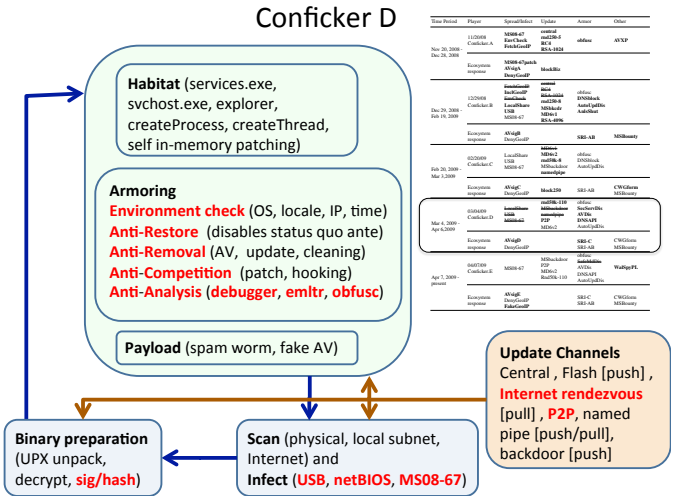

Fig. 3.14: Conficker D: Hardening with P2P, enlarged rendezvous and extensive armoring

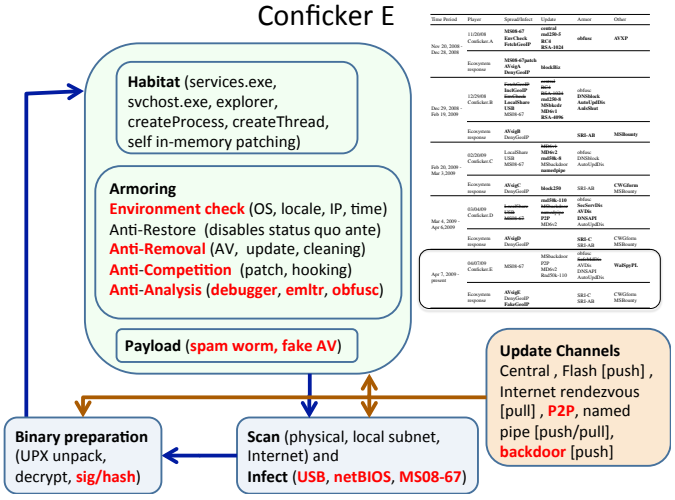

Fig. 3.15: Conficker E: Bootstrapping C to D and prototype crimeware payload

and malware, but it is not an absolute - some actors may not prioritize anonymity. So far, no group has credibly claimed responsibility for this worm.

3.4.1 Lessons Learned

The Lessons Learned document produced by the Conficker Working Group (CWG) essentially described the CWG's view of the final equilibrium: the botnet spread was limited, and the authors of the botnet were restricted from actually using it. We discuss three perspectives: measurement, media and goals.

3.4.1.1 Measurement

Adversarial games are played poorly in an absence of information. The ability to accurately assess the state of the game is vital to making intelligent and informed strategic decisions. This played out in Conficker domain in two ways.

The written and openly published analysis of the Conficker worm was instrumental to the defense against it. This was done mostly on an ad-hoc basis and then disseminated to blogs and security sites. This distributed effort then informed the organized effort. In particular, the identification of the randomized site generator was a crucial step in slowing the spread of Conficker.

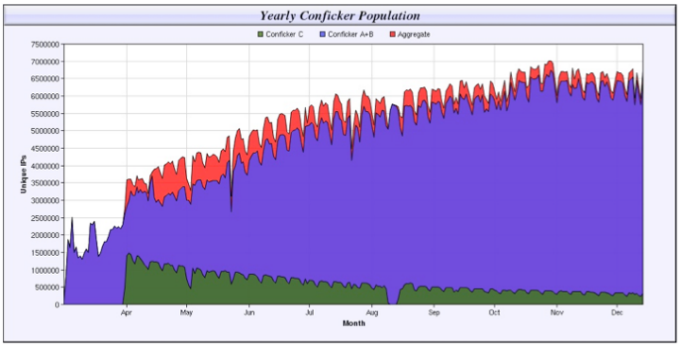

Fig. 3.16: Estimated Conficker population, Mar 2009-Dec 2009. Source: `http://www.shadowserver.org/wiki/uploads/Stats/conficker-population-year.png` Copyright ©2012 The Shadowserver Foundation. Reproduced with permission.

The second type of measurement involved the ability to assess effectiveness on a large scale. The CWG concluded in their Lessons Learned report that the Conficker authors were effectively thwarted by the CWG's efforts, particularly the moves to block the update mechanism. This finding is based on the lack of a large attack and on their estimates of the size of the heterogeneous Conficker botnet over time. This estimate is made by tracking connection attempts to sinkhole IP ranges under the control of defenders.

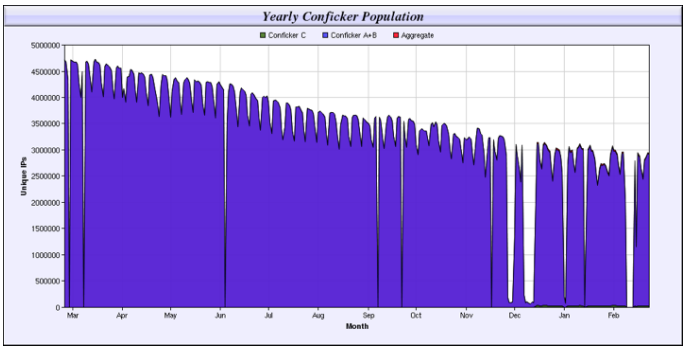

Fig. 3.17: Estimated Conficker population Mar 2011-Feb 2012. (Figure from `http://www.shadowserver.org/wiki/uploads/Stats/conficker-population-year.png` Copyright ©2012 The Shadowserver Foundation. Reproduced with permission.
)

According to this estimate, Conficker D (shown in green in Fig. 3.16) did not appear to have anywhere near the ongoing prevalence of A, B, and C combined, with a high point of over a million unique infected IP addresses identified in April 2009. Fig. 3.16 shows the changes in estimates for March through December 2009, the period following the estimated release of Conficker C and the Feb 12 official launch of the CWG. Fig. 3.17 shows the one year period ending February 2012 for comparison.

The CWG reports that their success on a technical level rested upon two factors: their ability to suppress the spread of Conficker D, and their ability to prevent the Conficker authors from gaining control of the A/B/C botnet. However, there is little analysis available of effective means of measuring the spread of Conficker, and there is evidence that Conficker D and E together included a masking strategy.

On a broader level, the CWG identifies their abilities to organize the defense community, get the word out about effective remediation, and to work with domain registrars on an amicable and organized basis - all crucial to their success in limiting the spread of Conficker. Although a number of the identified moves on the defense's part were unilateral (e.g. issuing the MS08-067 patch, removing the GeoIP file from maxmind.net), these organization efforts were critical to their ability to block the update domains. While in principle this could have eventually been done on an ad-hoc basis by the registrars themselves, in the event it took a persistent and organized team to push these changes through in a short time.

The question naturally arises as to why remediation efforts have largely ceased. Assessments vary, but the consensus seems to be that because the botnet is not perceived to be doing anything, defenders have individually

> come to the conclusion that it is not worth the expenditure of resources to remove the worm from their networks (even when they know they are infected): Given any large number of infected systems, remediation becomes a very difficult task, and even harder to justify when the infection does nothing. We have no doubt that many folks infected with Conficker may not even be aware that they have been compromised. We see this issue frequently, not only with Conficker, but also with other infections that clearly do demonstrate malicious activity. Any remediation effort from the provider's perspective will be painful and lengthy. There are no easy answers here.
> *ShadowServer.net, "Conficker"*
> `http://www.shadowserver.org/wiki/pmwiki.php/Stats/Conficker`

This is the expected outcome, from an adversarial perspective - expending additional resources to further suppress an adversary believed to be effectively beaten makes little sense.

3.4.1.2 Media

The remediation effort against Conficker was pursued primarily by individuals and smaller organizations. Only the actual owners and operators of the hosts in question could scan their systems for Conficker and either patch uninfected hosts or download and use removal tools. However, media attention can be an effective driver of public policy decisions and in publicizing rewards. As such, the publicity surrounding Conficker is an important part of the adversarial game.

There is evidence that both sides used or manipulated media scrutiny. Microsoft's February 12, 2009 press release announcing the creation of the Conficker Working Group served to:

1. Publicize efforts, recognizing the contributions of individuals and organizations and in turn providing an incentive to continue cooperating;
2. Publicize the $250,000 reward offered by Microsoft for information resulting in the arrest and conviction of the Conficker authors;
3. Propagate links to their web site containing Conficker removal tools;
4. Attempt to spur neutral actors to work against the Conficker authors, by framing the worm as a threat to the Internet community worldwide;
5. Attempt to persuade individuals to be more vigilant in removing the worm.

In terms of moves in an adversarial game, this was an attempt to:

1. Increase defender morale;
2. Make an attack against the persons operating the botnet;
3. Work around Conficker's DNSblock move;
4. Enlist additional resources;

5. Increase the rate at which independent actors apply remediation methods.

The effectiveness of this press release can be partly evaluated by examining the prevalence of “Conficker” as a Google search term and in news articles at the time. Labeled “B” in the Google Trends graph shown in Fig. 3.18, it resulted in a modest increase in activity. News articles on Conficker, particularly relating to a string of high-profile infections, kept the worm in the news. It is difficult to determine to what extent this affected ground-level remediation, but clearly Microsoft and other actors considered it possible.

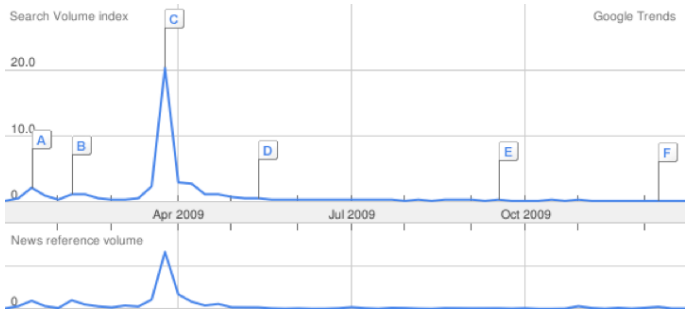

Fig. 3.18: Trend graph for search volume and news volume for ”Conficker” through 2009. Data from `trends.google.com`

However, the CWG did not achieve a cohesive public relations campaign. By releasing multiple press reports from individual members rather than from the working group as a whole, they largely denied themselves the ability to strategize and make deliberate moves with a minimum of leaks - a state of affairs that would have been necessary to implement, say, a comprehensive set of deception-based strategies (such as the GeoIP lookup by the HoneyNet project) [2].

There is evidence that the Conficker authors attempted to manipulate media attention. Conficker E was analyzed and determined to have a built-in event scheduled for April 1, 2009. That date is labeled “C” in Figure 16, and represents a peak. After that event passed without obvious incident, there was a flurry of activity (primarily pointing out that nothing happened), and then the level of scrutiny and public awareness dropped. It is not possible to determine whether this was intentional manipulation on the part of the Conficker authors, but it is plausible as a deliberate move to reduce attention, and consistent with changes they made to the worm to give it a lower profile.

3.4.1.3 Goals

The CWG's Lessons Learned document is itself a valuable outcome - self-assessment is a vital part of determining just what game was played and what the outcome was. There is a tendency to assign goals in hindsight according to what actually was achieved; in an effort to avoid that, their Lessons Learned document contains a lengthy section giving candid feedback by members of the working group on what the goals of the group were and assessing how well they were reached.

One of the outcomes of this feedback was a frank assessment of a lack of cohesion on secondary goals such as informing the public, spearheading remediation efforts, or forming an ongoing anti-malware concern. Indeed, the very phrasing of the CWG question to its members: *"In your opinion, what were the goals of the Conficker Working Group?"* implies loose coordination. While there was consensus that a key goal of the group was "to prevent the author from updating infected computers, control of the botnet and use of it to launch a significant cyber attack", there appears to be little consensus beyond that. And while most of those interviewed agreed that their efforts had been successful in achieving this agreed-upon key goal, that assessment was not universal.

In addition, because outreach was not identified as a primary goal, the CWG was somewhat circumscribed. Local ISPs were not deliberately or formally included, and government also had almost no participation in this effort except as a recipient of information. Particularly in the case of government, there was significant internal disagreement about the proper scope of the CGW and about the role of these external actors. This disagreement served to limit the potential moves that could be made by the defense, both in terms of concerted information-gathering at the ISP level and in terms of abilities reserved to government such as intelligence-gathering and subpoena power.

3.4.1.4 Analysis of CWG's Efforts

The ability to assess the success or failure of individual actions is vital to understanding the effects of those actions and subsequently, determining an effective course of action. The CWG team's ability to observe certain IP ranges under their control, as well as their ability to coordinate certain measures amongst themselves, certainly helped.

The most successful CWG strategies were those that were clearly agreed upon: The effort to block the set of randomized domains through the individual registrars (though not perfect) worked extremely well. Conversely, goals that were harder to articulate or were not shared among all participants (such as maintaining a cohesive PR voice in order to sustain a media strategy) were not as successful.

Incorporation of corroborating data (e.g. Google Trends, web logs of infected servers) would have given a more complete picture, allowed defenders to fine-tune their actions, and may have helped avoid myopic moves. More generally, the abil-

ity to incorporate observational data from pre-established processes to link these macro-observations to adversarial moves, seems critical in hindsight.

3.5 Analytic Model of Adversarial Quantitative Attack Graphs

In this final section, we develop some analytic methods describing how attackers and defenders might reason about attack graphs. Loosely speaking, an attack graph encodes sequences of steps an attacker would need to take to achieve a desired goal against a target system. The start state of an attack graph is where an attacker begins and the end state is "goal achieved".

To illustrate the concept, consider the following notional attack graph that is actually a high level depiction of a remote attack against a computer system with the goal of exfiltrating sensitive data. Attack graphs are common constructs in computer and network security analysis [14] [10].

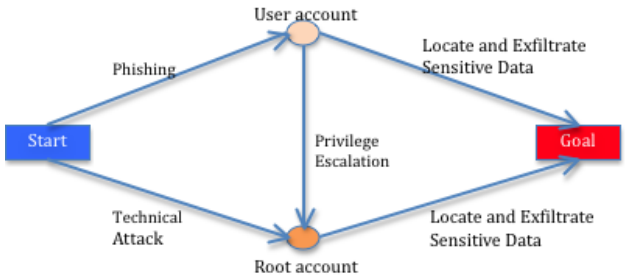

Fig. 3.19: Notional attack graph to illustrate the basic concept

Attack graph analysis has traditionally only addressed reachability, not adversarial dynamics aspects involving costs and strategies by the various agents [14][10]. There have been recent attempts to quantify attack graph analysis to include costs and transition probabilities to make the technology more appropriate to risk assessment and management [4]. Our intention here is to outline an approach that begins to get at actual adversarial dynamics through attack graph analytics. To that end, consider Fig. 3.20 below, which abstracts Fig. 3.19 into states, directed transition edges and consequently paths. This attack graph has three paths whose relationships with the labeled edges are depicted Table 3.5.

The relationships between path costs, edge costs and the path-edge relationships can be expressed quantitatively using the path-edge adjacency matrix M defined as

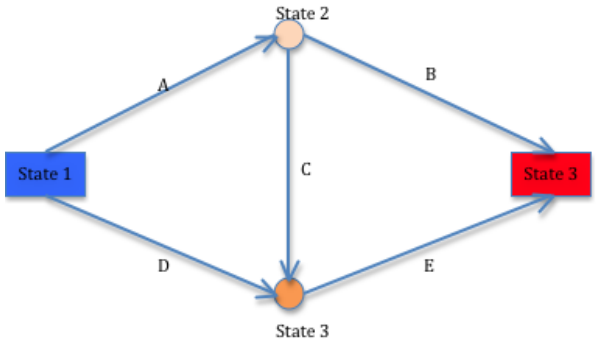

Fig. 3.20: Abstracted attack graph to illustrate the basic concepts

Table 3.5: Path-edge relationships together with path and edge costs

	A	B	C	D	E	Path Cost
Path 1	1	1	0	0	0	$Y_A + Y_B$
Path 2	1	0	1	0	1	$Y_A + Y_C + Y_E$
Path 3	0	0	0	1	1	$Y_D + Y_E$
Edge Cost	Y_A	Y_B	Y_C	Y_D	Y_E	

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

and the edge cost matrix $Y = [Y_A\ Y_B\ Y_C\ Y_D\ Y_E]'$ where $'$ denotes transpose. The relationship is that the path costs are $M * Y$. An informed and rational attacker would choose to exploit the minimal cost path whose cost is determined by the following optimization problem and inequalities.

$$\max \alpha$$

subject to

$$M * Y \geq \alpha \mathbf{1} \tag{3.1}$$

where $\mathbf{1}$ is the column vector of all 1's. All costs are by definition non-negative.

Assume that Y is the vector of current costs the attacker has to address as given by the attack graph and the defender has a total investment of D dollars to make in protecting the system. The defender has to allocate the D units across the various edges to make the attacker's goal more costly to reach.

If we let $T = [T_A\ T_B\ T_C\ T_D\ T_E]'$ denote the allocation of resources to attack path edges and assume, for simplicity at the moment, that there is a direct linear relation-

ship between investment and increased cost, then the defender's investment problem becomes a minimal cost attack-optimal defensive investment problem (MCA-ODI). The MCA-ODI problem is expressed as a linear program and can be solved by standard linear programming solvers including `linprog` in the MATLAB Optimization Toolbox.

$$\max \alpha$$

subject to

$$M * (Y + T) \geq \alpha \mathbf{1} \tag{3.2}$$
$$T * \mathbf{1} = D \tag{3.3}$$

such that

$$T \geq 0$$

The MCA-ODI formulation holds for every attack graph as quantified and interpreted above. Extensions to non-proportional and nonlinear relationships between T and the minimal cost attack path can be expressed as with the generalized formulation in Eqn. 3.4:

$$\max \alpha$$

subject to

$$M * (Y + f(T)) \geq \alpha \mathbf{1} \tag{3.4}$$
$$T * \mathbf{1} = D \tag{3.5}$$

such that

$$T \geq 0$$

where $f(T)$ is a general nonlinear function can be solved numerically but through more complex algorithms and with fewer analytic properties. Below, we demonstrate some simulations for the above problem with

$$Y = [Y_A \; Y_B \; Y_C \; Y_D \; Y_E]' = [5 \; 10 \; 15 \; 20 \; 25]'$$

and D varying between 0 and 100 units.

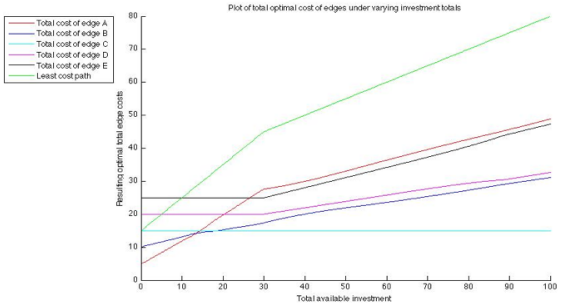

Fig. 3.21: Plot of total optimal edge costs starting with initial values specified by Y as above so the vertical axis depicts the optimal Y + T values as computed by solving the linear program. Note the changing allocations.

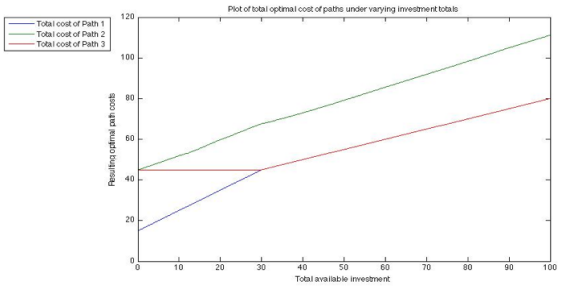

Fig. 3.22: Plot of total optimal path costs starting with initial values specified by Y as above so the vertical axis depicts the paths costs under optimal allocation as computed by solving the linear program. The minimal cost path is determined by the lowest curve.

3.6 Future Work

Future work extending the research presented in this paper includes:

1. Apply this methodology specifically to Conficker and/or other malware;
2. Use information markets or other mechanisms to quantify edge and path costs;
3. Compare and correlate computed investments with observed actions as extracted and analyzed in the Conficker analysis above;
4. Explore the actual functional relations between investments and attack edge cost increase beyond the linear relations we have used in this report;
5. Relate this optimal attack graph defense investment model and solutions with previously developed network interdiction problems, max flow-min cost formu-

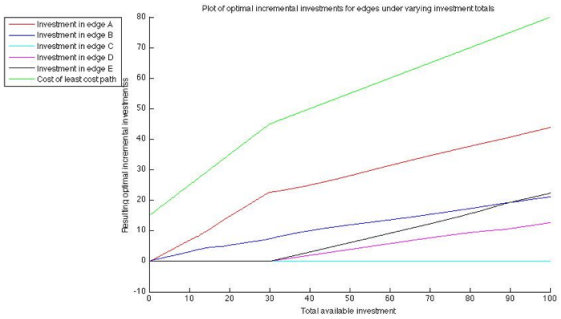

Fig. 3.23: Plot of total incremental investments per edge without the initial values as specified by Y but computed by the linear program including Y. The vertical axis depicts the edge investment allocation under optimal allocation as computed by solving the linear program. Note the different slopes and crossover point at 90 units.

lations (for scalability) and primal-dual interpretations as they might relate to Nash Equilibria or other game theoretical solution concepts;

6. The role of deception and counter-deception in repeated games which might communicate agents' perceived notions of costs and defensive postures to mislead the adversary.

Acknowledgements We thank Vincent Berk, Patrick Sweeney, David Sicilia, Gabriel Stocco, James Thomas House and other colleagues at Process Query Systems, Dartmouth College, Siege Technologies and elsewhere for discussions and contributions that have led to these findings. This work was partially supported by DARPA Contract FA8750-11-1-0253 at Dartmouth College and US DoD contracts to Process Query Systems. All opinions and results expressed in this article are those of the authors and do not represent the positions or opinions of the US Government or sponsoring agencies.

References

1. Bilar, D.: Degradation and subversion through subsystem attacks. IEEE Security and Privacy **8**, 70–73 (2010). DOI 10.1109/MSP.2010.122. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5523869`
2. Bilar, D., Saltaformaggio, B.: Using a novel behavioral stimuli-response framework to defend against adversarial cyberspace participants. In: Cyber Conflict (ICCC), 2011 3rd International Conference on, pp. 169–185. CCD COE, IEEE, Tallinn, Estonia (2011). URL `http://www.ccdcoe.org/publications/2011proceedings/UsingANovelBehavioralStimuli-ResponseFramework...-Bilar-Saltaformaggio.pdf`
3. Bowden, M.: Worm : The First Digital World War. Grove Press (2011)

4. Carin, L., Cybenko, G., Hughes, J.: Cybersecurity strategies: The queries methodology. Computer **41**(8), 20–26 (2008). DOI 10.1109/MC.2008.295. URL http://dx.doi.org/10.1109/MC.2008.295
5. Fudenberg, D., Tirole, J.: Game Theory. The MIT Press, Cambridge MA (1991)
6. Greengard, S.: The war against botnets. Commun. ACM **55**(2), 16–18 (2012). DOI 10.1145/2076450.2076456. URL http://doi.acm.org/10.1145/2076450.2076456
7. Group, C.W.: Lessons learned. http://www.confickerworkinggroup.org/wiki/uploads/Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf (2010)
8. Hart, S., Mas-Colell, A.: Uncoupled dynamics do not lead to Nash equilibrium. American Economic Review **93**, 1830–1836 (2003)
9. Hofbauer, J., Sigmund, K.: Evolutionary game dynamics. Bulletin (New Series) of the Amer. Math. Soc. **40**(4), 479–519 (2003)
10. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: Proceedings of the 22nd Annual Computer Security Applications Conference, ACSAC '06, pp. 121–130. IEEE Computer Society, Washington, DC, USA (2006). DOI 10.1109/ACSAC.2006.39. URL http://dx.doi.org/10.1109/ACSAC.2006.39
11. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. IEEE Security and Privacy **9**(3), 49–51 (2011). DOI 10.1109/MSP.2011.67. URL http://dx.doi.org/10.1109/MSP.2011.67
12. Levine, D.F.D.K.: The Theory of Learning in Games (Economic Learning and Social Evolution). The MIT Press, Cambridge MA (1998)
13. Microsoft: Microsoft Security Intelligence Report, Volume 12, July through December 2011. http://download.microsoft.com/download/C/9/A/C9A544AD-4150-43D3-80F7-4F1641EF910A/Microsoft_Security_Intelligence_Report_Volume_12_English.pdf (2012)
14. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: Proceedings of the 1998 workshop on New security paradigms, NSPW '98, pp. 71–79. ACM, New York, NY, USA (1998). DOI 10.1145/310889.310919. URL http://doi.acm.org/10.1145/310889.310919
15. Porras, P., Saidi, H., Yegneswaran, V.: Conficker c p2p protocol and implementation. SRI International, Menlo Park, CA, Tech. Rep (2009)
16. Porras, P., Saïdi, H., Yegneswaran, V.: A foray into conficker's logic and rendezvous points. In: Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more, LEET'09, pp. 7–7. USENIX Association, Berkeley, CA, USA (2009). URL http://dl.acm.org/citation.cfm?id=1855676.1855683
17. Ren, H., Ong, G.: Exploit ms-08-067 bundled in commercial malware kit. http://www.avertlabs.com/research/blog/index.php/2008/11/14/exploit-ms08-067-bundled-in-com mercial-malware-kit/ (14 Nov 2008)
18. Rodionov, E., Matrosov, A.: The evolution of TDL: Conquering x64. http://go.eset.com/us/resources/white-papers/The_Evolution_of_TDL.pdf (2011)
19. Rubinstein, A.: http://arielrubinstein.tau.ac.il/papers/afterwards.pdf (2007)
20. Rubinstein, A.: Theory of Games and Economic Behavior (Commemorative Edition). John von Neumann and Oskar Morgenstern (with an introduction by Harold Kuhn and an afterword by Ariel Rubinstein). Princeton University Press, Princeotn NJ (2007)
21. Shin, S., Gu, G.: Conficker and beyond: a large-scale empirical study. In: Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC '10, pp. 151–160. ACM, New York, NY, USA (2010). DOI 10.1145/1920261.1920285. URL http://doi.acm.org/10.1145/1920261.1920285
22. Sicilia, D., Cybenko, G.: Application of the replicator equation to decision-making processes in border security. Proceedings of SPIE Defense and Security, 2012, Baltimore MD (2012)
23. Sparrow, C., van Strien, S., Harris, C.: Fictitious play in 3 x 3 games: the transition between periodic and chaotic bahavior. Games and Economic Behavior **63**, 259–291 (2008)
24. Sweeney, P., Cybenko, G.: An analytic approach to cyber adversarial dynamics. Proceedings of SPIE Defense and Security, 2012, Baltimore MD (2012)