



## **Weaponized: Virulence and Malware**

**infectionvectors.com**

**August 2006**

### **Overview**

Weapons are used to increase the harm someone can inflict, leveraging the strength of its wielder many times over to produce a grander result than could be expected without the tool. Applied to computer-based crimes, a piece of malware is able to target millions of subjects at any given time, can be controlled with exceptional precision, and is capable of any number of attacks. The use of the virus, worm, or Trojan horse in the cyber arena has been compared to warfare many times before, often with grandiose terminology – this report is not intended to repeat such efforts. The use of malicious code on the Internet, however, does lend itself to some of the same academic discussions of weapon improvement as are seen in military and defense communities. This article examines the use of computer-borne weapons and compares their development to physical weapons – such as those being studied by biological warfare experts.

### **Battle Plan**

Using a virus as a weapon, whether biological or cyber in origin is not a complete strategy in itself. A biological virus cannot win a battle; the virus cannot take over a country or execute any plan other than to make people and animals sick. A computer worm is incapable of destroying buildings, overthrowing governments, or making tactical decisions. In both cases, the virus may be perfectly capable of creating chaos, destabilizing an environment to allow for a complementary attack – making it a powerful component in attacks where the viral damage in itself is not the end goal. A biological pathogen may weaken a nation to the point where a physical invasion is made attractive (or winnable). A well-written, well-distributed Trojan may allow the author to invade a network and complete reconnaissance efforts – or it may absorb enough resources to allow for a completely separate cyber attack all together. But as a tactical utility, how the virus is made into weapons and into increasingly more useful (powerful) weapons is the subject of tremendous research.

In a series of articles in the Washington Post entitled “Five Years Later,” Joby Warrick documented the state of bioterrorism research and defense efforts.<sup>1</sup> Many of the issues raised are reminiscent of those faced by cyber-virus fighters. A biological virus is designed, quite obviously, as a weapon. Just as traditional weaponry (explosive-based) is optimized by increasing its destructive power or delivery precision, it is also made less effective by studying those areas for countermeasures. The biological threats are studied for the same reasons – to make stronger toxins and to understand the threat so that medicine and detection can be improved.

In the articles mentioned above, Warrick points out three attributes that are indicative of strengthening biological weapons:

- “Weaponized” Pathogens – taking a threat and turning into a form that is easily distributed
- “Novel Delivery Systems” – ability to disperse the weapon across a broad area
- “Genetically Engineered Threats” – making the weapon more deadly or more difficult to stop

One can already see correlations to improving Internet-based attacks and defenses in the characteristics above. Developers of both types of weapons certainly seem to have the same aims. As Warrick writes of the bioterrorist’s goal to produce, “...superbugs that ordinary drugs cannot stop...”<sup>1</sup> The desire of malware authors to produce undetectable, yet still powerful applications drives a number of innovations in viruses, worms, Trojans, and related software. Examples of each trait above are noted in the following sections.

## Dispenser

Having a lot of targets and having a strategy for distribution is not enough for a piece of malware that is difficult to move. An exceptionally large piece of code is not readily transferred, even with the high-bandwidth environments of many infection targets. The same is true for malware “collections” – attacks that require a number of utilities or configuration files to accomplish its mission. This is the case for many IRC-based bots, as they need to include configuration files to ensure the compromised machine correctly reports back to the author-controlled channel.<sup>2</sup> That handicap has been overcome by packaging the entire contents into a self-extracting archive, generally an SFX file (as the SFX format allows for scripting both the decompression routine and executing a file from the package).<sup>3</sup> The single SFX file provides a means of taking the concept of a mIRC-based Trojan to a “weaponized” form.

Numerous examples of this type of attack are readily available on the Internet. Many attackers utilize a dual-phased scheme in an effort to get their malware on target machines. This generally involves a mass email blatantly asking the recipient to download and execute the malicious code:

```
<strong>Hello </strong><br>
You have just received a postcard from <a href="http://www.yahoo.com">
www.yahoo.com</a> .<br>
<strong>If you'd like to see the rest of the message click <a
href="http://[removed]/~info/postcard.gif.exe">here</a> to
receive your animated postcard! </strong><br><br>
```

Although the requests do not mention that the application to be retrieved is actually an IRC-bot, there is a wide array of tactics used to coax the reader to click the provided link without looking behind the curtain, so to speak. Some implore the reader to download a desirable file, such as an urgent update:

**Click Here Then Open Or Run For Automatic Update**



Now you can get updates for Windows, Office and other Microsoft applications all in one place. Microsoft Update is a new service that brings you all the features and benefits of Windows Update plus downloads for other Microsoft applications including Office.

**Benefits**

- **Improve your computer's health and security.**  
Regularly update your computer with the latest software from Microsoft to boost the security and reliability of your Windows PC.
- **Easy and flexible.**  
Microsoft Update is easy to set up and use. [Click Here Then Open Or Run For Automatic Update](#)

The text of the image shown above came with an email indicating that a “new virus” was on the loose; implying that the “update” offered would protect the recipient from infection.

From: update@microsoft.com  
Subject: Warning! New Virus On The Internet! Update Now!

Of course, the download was neither a security update nor from Microsoft. The scam is not quite as compelling when viewing the HTML behind the message, as seen in this snippet from the attempt above:

```
<TD width="100%"><FONT size=4>&nbsp;<FONT color=#ff0000><A class=red href="http://update.microsoft.go.ro/update.exe" target=_blank><FONT color=#ff0000 size=6><STRONG>Click Here Then Open Or Run For Automatic Update</STRONG></FONT></A></FONT></FONT></TD></TR></TBODY></TABLE>
```

This “virus update” from the “go.ro” domain is, as mentioned above, an IRC-based bot collection, utilizing not only a customized copy of mIRC, but also complementary programs to hide the mIRC window and add autostart entries to the local Registry. It comes as a single EXE, an SFX file that was analyzed with WinRAR. The SFX includes a script that places the enclosed files in the correct location and kicks off the installation routine:

update.exe:SFX:700KB (717,558 bytes)

```
;The comment below contains SFX script commands

Path=c:\windows\system\
SavePath
Setup=svchost.exe
Silent=2
Overwrite=1
```

As seen above, the “setup” file is “svchost.exe,” which, to be sure is not the “Generic Host Process for Win32 Services” one is accustomed to seeing in the “system32” directory of Microsoft Windows builds. This application is actually a modified version of mIRC, the popular IRC client (popular with both legitimate IRC users and bot coders). This revision of mIRC, internally named “FldBot” by “Botz Team,” has a copyright 2004 tag indicating authorship by stefys and GHoSTKiLL. Even the most casual perusal of the debug output yields significant clues as to what this code was meant to do:

**Snippet #1:**

```
004C9344 |. 68 7C1E5800   PUSH explorer.00581E7C   ; /Arg4 = 00581E7C ASCII
"C:\WINDOWS\system32\mirc.ini"
004C9349 |. 68 E25C5800   PUSH explorer.00585CE2   ; |Arg1 = 00585CE2 ASCII "113"
004C934E |. E8 A739F6FF   CALL explorer.0042CCFA   ; \explorer.0042CCFA
004C9353 |. 50            PUSH EAX                 ; |Arg3
004C9354 |. 6A 00        PUSH 0                   ; /Arg2 = 00000000
004C9356 |. 83C3 08      ADD EBX,8                 ; |
```

**Snippet #2:**

```
004036D0 |. BB C0E75500   MOV EBX,explorer.0055E7C0 ; ASCII "agent"
004036D5 |. BE 7A9F5700   MOV ESI,explorer.00579F7A ; ASCII "* Connect retry #5
London.UK.Eu.UnderNet.org (6667)"
004036DA |. BD 7C1E5800   MOV EBP,explorer.00581E7C ; ASCII
"C:\WINDOWS\system32\mirc.ini"
```

As with other mIRC-bots, this one comes complete with the requisite configuration files, ensuring that the client connects to the author-controlled channel. In this case, some of the text files are named with “EXE” and “COM” extensions. The use of the executable designation is likely an attempt to obfuscate the presence of rogue files; their location is “%Windows%\system” where one is used to seeing executable code but not a lot of text files. The contents of the respective files are familiar to those that have examined IRC bots:

mirc.ini:

```
[ident]
active=yes
userid=thief
system=UNIX
port=113

[dde]
ServerStatus=on
ServiceName=anti-virus
CheckName=off

[mirc]
user=$read(users.exe)
email=best
nick=hzhuha
anick=hzhuha2
```

remote.ini

```
[users]
```

```
n1=100:*!*@meanDevil.users.undernet.org
n2=100:*!*@nike13.users.undernet.org
n3=100:*!*@byz.users.undernet.org
n4=100:*!*@BAROSANU1.users.undernet.org
n5=100:*!*@fumegatorul.users.undernet.org
n6=100:*!*@cbo.users.undernet.org
```

```
[variables]
n0=%chan #re-ops warez12
```

**id.exe**

-a text file with 16,445 names for the identd server

**rundll.exe**

-text file with server addresses:

**vir.exe**

-collection of 2,269 NICKS

These files, plus others (noted in the Appendix) such as the mIRC command script file “win.ini” allow the compromised machine to make a connection to the controller’s IRC channel and await additional instructions. The routine is not complete, however, as win.ini calls two other applications. The first is a copy of HideWindow 1.43 by Adrian Lopez (circa 1996).<sup>4</sup> The program makes visible windows invisible (and vice versa) to someone viewing the Desktop. It should be clear to the reader that the malware coder was somewhat handcuffed by the choice of execution routine, as the mIRC windows is quite visible during startup and is hidden only after a few seconds (as HideWindow/Window Hider is called by an already running copy of mIRC). This is an interesting decision on the coder’s part, as the person that coded the package could very well have had the IRC client begin minimized, at the very least, or initiate the entire sequence with a separate file that ensured that Window Hider did its job as soon as mIRC began.

The second piece of code called by svchost.exe is a DLL crafted to ensure that the proper autostart entries are added to the Windows Registry and attempts to remove some of the default mIRC directories from a user’s purview. This file, “reg.dll,” is yet another file type added to the mix – a packed executable that is only accessible via the svchost.exe application.

```
UpackByDwing@ PE L à.Upack
```

The line above, snipped from a strings output, shows the packer of choice, Upack. Called through a debugger, the intentions of this code are also fairly clear:

```
0021FB7C 00CF73E1 ASCII ":\windows\system\svchost.exe"
...
0021FBD0 7C90F0AA RETURN to ntdll.7C90F0AA from
ntdll.RtlMultiByteToUnicodeN
```

```

0021FBD4  00635D40  UNICODE "SOFTWARE\Microsoft\Windows\CurrentVersion"
...
0021FBE4  7C90EE18  ntdll.7C90EE18
0021FBE8  7C91044C  RETURN to ntdll.7C91044C from ntdll.7C90EDC2
0021FBEC  7C910570  ntdll.7C910570
0021FBF0  /0021FC04
0021FBF4  |7C9109BC  RETURN to ntdll.7C9109BC from ntdll.RtlFreeHeap
...
0021FC00  |00635D40  UNICODE "SOFTWARE\Microsoft\Windows\CurrentVersion"
...
0021FCF0  100340D8  ASCII "SOFTWARE\Microsoft\Windows\CurrentVersion"
0021FCF4  00000000  ..
0021FCF8  00000000  ..
0021FCFC  100340CC  ASCII "Run"

```

The most commonly used Windows startup key (...CurrentVersion\Run) plus the default location of the bot file add up to a standard means of ensuring that the malware is executed with each restart. Using a compressed DLL to accomplish something that also could have been set via plain batch file is another interesting facet of this sample where numerous tactics are bundled together with a singular purpose.

The basis for the attack, however, is forged from readily available, seemingly innocuous tools. The applications are easy to obtain and deploy, as the window hiding, IRC, and executable packaging applications require little training to successfully configure. All of the tools are found with simple searches and would likely trip an alarm in only the most highly-guarded environments. This modest set of tools is capable of opening the door to any complementary attack<sup>5</sup> – allowing the controller to retrieve any asset from the compromised device, use the zombie in larger attacks, or sell the bot to be part of another plan altogether.

When considering what truly makes the attacks worthwhile, however, one will invariably return to the sheer volume of possible targets. Simple tools and schemes are viable in this model because the malware can be presented to so many targets in a short period of time. This means that a mere 1% success rate can yield thousands of new zombies. The spamming infrastructure that is available to anyone with a bot to push is well-established and above the suspicions of most security analysts. The cacophony of spam has made the Internet community numb to the random messages hitting Inboxes and filters. Spam can easily take up more time and money for a dedicated administrator than any other single security problem. The user base is accustomed to the annoyance as there is little shelter from the incessant advertisements flying across cyberspace. Acquiescence is a normal reaction to the unstoppable flurry of spam, especially when it appears to be only an irritant, not a threat to the integrity of our networks. Inside this cloud the attacker operates with its greatest ally – anonymity, a tool that makes distributing weapons exceptionally easy.

## Salient

Crafting a successful malicious code attack requires a number of good planning practices: notably, a sense of precisely what it is one calls “successful.” In addition, a solid delivery mechanism is important to any endeavor whose viability requires a high volume of targets. Having both of these traits would make a malicious coder and their code dangerous to the Internet.

Salinity, a virus that appeared in 2003, fits such a profile. Designed to add itself to executable files on Windows machines, Salinity installs a keylogger and initiates some destructive acts against anti-virus-related files. In themselves, these crimes open the door to many other attacks, including using stolen passwords and adding new pieces of malware to an already infected system without fear of detection.

Instead of simply dropping a copy of itself in the “startup” directory or adding a value to one of the autostart keys, Salinity’s file infection routine often aims for the files already set to start with the OS, avoiding possible detection by someone (or some tool) examining the Registry:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run  
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

Once running, the virus avoids a number of files in the system directories, as they are protected by Microsoft’s System File Checker (which would replace the files incessantly once the “modification” was detected). For other executables, Salinity overwrites the original entry point (OEP) with its own decryption routine (which points to the appended viral code) and moves the OEP so that the original program is executed after the malware. The component dropped by Salinity will connect to an IRC channel and await commands, reminiscent of the previous section’s threat.<sup>6</sup>

In early 2006, Salinity was found traveling with revisions of the Beagle worm. The revision of Beagle used for such dissemination was actually quite old (in relative terms to the life of Beagle) – possibly indicating that someone adapted the source dropped by Beagle back in 2004. Although it is presumed that the release was quite deliberate, as Salinity is a file infector, it is impossible to definitively determine whether this “piggybacking” was intentional or not. In either case, the slow-spreading PE-infector Salinity was given quite a distribution boost by attaching itself to a worm with the propagation speed of Beagle. The interest in Beagle does not stop with mobility; certain Salinity variants also looked for specific files left by Beagle infections. Notably, the virus searches for “vcremoval.dll,” the list of email addresses harvested by Beagle variants beginning in early 2006.

In the years since Beagle was released (January of 2004), the security world has witnessed a worm that is managed by a fairly clear business plan.<sup>7</sup> The process taken to ensure Beagle is successful, including the way it harvests new “products” from unsuspecting boxes has been a model for all malware authors. Indeed, the Beagle release patterns and testing methods have been called a “blueprint” for attackers.<sup>8</sup> If, in fact, the Salinity developers are taking that blueprint and running an illicit business in the spirit of

the Beagle author, then it cannot be a surprise to security analysts. Combining publicly available malware data could allow for virtually anyone with moderate technical knowledge to combine their predisposition for crime (in any form) with a successful volume-driven campaign across the electronic assets of the world. Sality may be the first of many high-profile examples to come.

Is this a case of “weaponizing” the payload of Sality? Consider the second characteristic of the biologically-based systems described above: “novel delivery.” The distribution mechanism in this case could be changed easily to avoid signature-based detection (as has been evidenced by the original releases of Beagle). In addition, the Beagle worm was already designed to terminate most well-known security applications – making it the perfect infantry for the Sality virus. Utilizing both PE infection and network worm propagation tactics certainly allows for a broad base of possible targets in a short period of time.

## On Virulence

Main Entry: **vir·u·lent**

Pronunciation: 'vir-&l&nt, 'vir-y&-

Function: *adjective*

Etymology: Middle English, from Latin *virulentus*, from *virus* poison

**l a** : marked by a rapid, severe, and destructive course <a *virulent* infection> **b** : able to overcome bodily defensive mechanisms : markedly pathogenic <*virulent* bacteria> <sup>9</sup>

Warrick noted that one goal of the weapon designer is to make the pathogen more deadly when it is released. In terms of computer attacks, often “more deadly” is associated with destruction of data. Many of the viruses and virus hoaxes that have traversed the Internet have been concerned with erasing hard disks or groups of files. Security professionals are probably more concerned with surreptitious remote control of their servers than a virus that reformats a data drive – there are always backups, but there is little comfort in for privacy violations, bad press coverage, lost trade secrets, etc. And although each of us will evaluate the impact to confidentiality, integrity, and availability of our respective systems differently, providing a control channel for a distant, unknown attacker to manipulate at will is disheartening. Of course, although the traditional notion of computer viruses conjures images of wrecked desktops, the modern piece of malcode is more likely to house a remote control vehicle for the author. The popularity of bots and the associated bot-making kits have cemented the ubiquity of such applications.

The once annoying IRC bot (as it was originally incarnated for use in IRC chat forums) has become one of the biggest threats to Internet-connected systems, as it has the capability to infect unprotected machines and launch large-scale attacks against those that are otherwise secure. The bot has accompanied the open-ended payloads of many modern worms – leaving the end result to the controller. The building blocks for any given attack, however, remain quite familiar. Even among the various bot families, there is a consistent feel to their features:

- Incorporation of rootkit-like functions (hide processes, files, etc.)

- Ability to control propagation (“stealth” modes, idling, etc.)
- Retrieval of arbitrary files (and execution...)

The modularity of the bots and their associated construction kits<sup>10</sup> has allowed malware authors to be quite nimble with new exploits. Once a vulnerability and suitable attack have been published, it is often a new iteration of SDbot, Agobot, or the like that takes advantage of them first. This was true in 2004 with the LSASS (MS04-011) overflow<sup>11</sup> and remains true through 2006. In August of 2006, MS06-040 was released by Microsoft. It noted a flaw in the Server Service in Windows 2000, 2003, and XP systems. The first piece of malware exploiting this hole in the wild was a revision of Randex/SDbot<sup>12</sup> which was released less than two weeks after the security bulletin. This makes both the flaw more dangerous to have resident on a system (as it can be exploited via automated attack) as well as the aging bot infrastructure (which is strengthened by an increased ability to inflict damage on a wider array of devices).

Quite often, the focus for a security analyst is the payload carried by a worm or virus. That is a natural reaction; everyone wonders exactly what a piece of malware will do once it infects a system. But to effectively fight malware (not a single strain, but the class of software as a whole), it is imperative that the security community identify the strengths and weaknesses of the Internet (and connected systems) with regard to the traits of “weaponization” taken from Joby Warrick’s article.

### **Solvency**

With the focus on taking the common, even with regard to malware, and making it more dangerous, it is incumbent on security analysts to establish procedures for safeguarding network systems. To counter bioterrorism threats, the US established the Bioshield Project,<sup>13</sup> in part, an effort to ensure that vaccines/medicines would be available in a timely fashion. There are numerous lessons to be learned from this endeavor. First, the use of an incentive-based approach should not be overlooked when considering how to bolster security supply lines. In the case of anti-virus research, there are a vast number of creative analysts that may be able to construct possible attack vectors and payload descriptors of future attacks. In addition to the traditional anti-virus companies, these outlets can be plugged into the malware defense infrastructure.

Second, the “market a cure” approach not only pushes incentives for research and production, but also ensures that a distribution channel exists for the vaccines. Getting the information, signatures, and cleaning software into the hands of security professionals, whether just inside a large organization or across the globe cannot be discounted.

Of course, a call for requiring email authentication mechanisms is requisite when discussing mail-borne schemes. However, this call has been made before in numerous forums and is far from a complete solution to the malware threat. As has been seen here, Internet attacks are heavily reliant upon their “delivery systems” to be successful. But, that system is predicated upon the product that the author (or releaser, as the two have been seen to be different<sup>x</sup>) wishes to harvest. A bot herder looking to produce an army for

hire has a different motivation than the leaser, possibly wanting to distribute a new piece of code from an anonymous network. There has been an antivirus industry for years, one that will continue to make strides defeating malware as it is used to create the bots. What has begun to emerge, however, is an effort to discover the bot nets before they are used for (additional) evil ends. Currently, bot net controllers take the very commonplace: email, IRC, file transfers, IM, etc., and turn them into very powerful weapons. Detecting the malicious communications within the ever-present background noise of the Internet will prove to be both difficult and exceptionally rewarding.

## References

1. Warrick, Joby. The Washington Post. "The Secretive Fight Against Bioterror." (Part of "Five Years Later" series of articles): [http://www.washingtonpost.com/wp-dyn/content/article/2006/07/29/AR2006072900592\\_pf.html](http://www.washingtonpost.com/wp-dyn/content/article/2006/07/29/AR2006072900592_pf.html)

2. For more on bots and bot nets, please see: "The Mytob Infantry" [http://www.infectionvectors.com/library/mytob\\_infantry\\_iv.pdf](http://www.infectionvectors.com/library/mytob_infantry_iv.pdf), "One's Complement" [http://www.infectionvectors.com/vectors/ones\\_complement.htm](http://www.infectionvectors.com/vectors/ones_complement.htm), and, "Agobot and the 'Kit'chen Sink." [http://www.infectionvectors.com/vectors/Agobot\\_&\\_the\\_Kitchen\\_Sink.pdf](http://www.infectionvectors.com/vectors/Agobot_&_the_Kitchen_Sink.pdf).

3. SFX, or "Self eXtracting" archives are capable of executing a program, copying files to a predetermined directory, and other useful functions. Win-RAR's site offers some additional information: [http://www.win-rar.com/index.php?id=24&kb\\_article\\_id=42](http://www.win-rar.com/index.php?id=24&kb_article_id=42).

4. Window Hider turns up as a possible undesirable program with many antivirus products. One example is the McAfee product: [http://vil.nai.com/vil/content/v\\_99939.htm/](http://vil.nai.com/vil/content/v_99939.htm/)

5. An excellent summary of bot family commands: "Know your Enemy: Tracking Botnets - Bot-Commands" Honeynet Project 17 February 2005. <http://www.honeynet.org/papers/bots/botnet-commands.html>

6. For a report on Sality's technical workings, please see the following Symantec document: [http://www.symantec.com/security\\_response/writeup.jsp?docid=2006-011714-3948-99](http://www.symantec.com/security_response/writeup.jsp?docid=2006-011714-3948-99).

Information on one recent variant, Sality.Q, can be found at: F-Secure [http://www.f-secure.com/v-descs/sality\\_q.shtml](http://www.f-secure.com/v-descs/sality_q.shtml).

The Beagle (Bagle) and Sality connection can be explored via: McAfee Bagle.gen!Sality [http://vil.nai.com/vil/Content/v\\_138598.htm](http://vil.nai.com/vil/Content/v_138598.htm).  
[http://us.mcafee.com/virusInfo/default.asp?id=description&virus\\_k=138530](http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=138530)

Beagle.BW [http://www.symantec.com/security\\_response/writeup.jsp?docid=2006-030114-0908-99&tabid=2](http://www.symantec.com/security_response/writeup.jsp?docid=2006-030114-0908-99&tabid=2)

Sality's Fake Error Message & Search for Beagle email list: Win32/Sality Family <http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?ID=52797>

7. "Years of the Beagle" Parts 1-5 of the Beagle History. [http://www.infectionvectors.com/library/years\\_of\\_the\\_beagle.pdf](http://www.infectionvectors.com/library/years_of_the_beagle.pdf).

8. Keizer, Gregg. "Bagle Worm Seen As 'Blueprint' for Web Criminals." TechWeb, April 28, 2005. <http://www.techweb.com/wire/security/161601776>.

9. The Merriam-Webster Online Dictionary. <http://www.m-w.com/dictionary/virulent>

10. A review of one of the best-known bots and its construction kit can be found at:  
“Agobot and the ‘Kit’chen Sink.”

[http://www.infectionvectors.com/vectors/Agobot\\_&\\_the\\_Kit-chen\\_Sink.pdf](http://www.infectionvectors.com/vectors/Agobot_&_the_Kit-chen_Sink.pdf).

11. Ullrich, Johannes. Internet Storm Center Handler’s Diary. May 5, 2004.

<http://www.incidents.org/diary.php?date=2004-05-05>.

12. Randex.GEL is described on Symantec’s site:

[http://www.symantec.com/security\\_response/writeup.jsp?docid=2006-081910-4849-99](http://www.symantec.com/security_response/writeup.jsp?docid=2006-081910-4849-99).

13. “Project Bioshield: Progress in the War on Terror.”

<http://www.whitehouse.gov/infocus/bioshield/>

**Appendix: Additional Information for the Curious**“Virus Update” mIRC bot:

reg.dll’s Registry value insertion:

```

004AFB5A  803B 00          CMP BYTE PTR DS:[EBX],0
004AFB5D  ^75 95          JNZ SHORT svchost.004AFAF4
004AFB5F  3B37          CMP ESI,DWORD PTR DS:[EDI]
004AFB61  7D 0D          JGE SHORT svchost.004AFB70
004AFB63  8B45 90          MOV EAX,DWORD PTR SS:[EBP-70]
004AFB66  33D2          XOR EDX,EDX
004AFB68  8914B0        MOV DWORD PTR DS:[EAX+ESI*4],EDX

004AFB01  83C4 08          ADD ESP,8
004AFB04  8985 78FFFFFF    MOV DWORD PTR SS:[EBP-88],EAX
004AFB0A  83BD 78FFFFFF 00 CMP DWORD PTR SS:[EBP-88],0
004AFB11  75 09          JNZ SHORT svchost.004AFB1C
004AFB13  8B45 90          MOV EAX,DWORD PTR SS:[EBP-70]
004AFB16  891CB0        MOV DWORD PTR DS:[EAX+ESI*4],EBX
004AFB19  46            INC ESI
004AFB1A  EB 43          JMP SHORT svchost.004AFB5F

004AFB04: EAX=02AA4071, (ASCII "c:\winodws\system\svchost.exe")

```

win.com’s notification list:

```

darki
darkman
adi
adrian
ady
bebelusa
zmeu
bj
corekt
ro
ok
bnc
hack
hacker
ebay

```

Also included with the “update” mIRC bot, but not discussed in the main body of the paper is “Script1.ini,” another configuration file included for the IRC client:

```

[script]
n0=
n1=
n2=
n3=

```

```
n4=}
n5=alias ignore //ignore $1- | .ignore -wd * | .silence *
n6=on *:CONNECT://.timer 1 3 mode $me -i | .timer 0 1000 ignore -r |
.timer 0 1000 big | ignore -r | timer 1 120 big
n7=
```

rundll.exe (server names)

```
diemen.nl.eu.undernet.org:6667
London.UK.Eu.Undernet.Org:6667
oslo.no.eu.undernet.org:6667
Montreal.QC.CA.Undernet.org:6668
elsene.be.eu.undernet.org
Zagreb.HR.Eu.UnderNet.org:9999
helsinki.fi.eu.undernet.org:6667
miami.fl.us.undernet.org:6667
mesa.az.us.undernet.org:6667
eu.undernet.org:6667
us.undernet.org:6667
```

### Salinity Information

There are numerous iterations of the Salinity virus, which has been around over 3 years as of this writing. Many drop a DLL onto the system (the keylogger) and inject this routine into all running processes.

The list of files the later iterations attempt to delete will likely look familiar:

aler	kav
anda	nod
anti	outp
avp	scan
bidef	tren
clean	troj
guar	zone