Computing Under Occupation

Klaus Kursawe and Stefan Katzenbeisser
Philips Research
Information and System Security Group
Eindhoven, The Netherlands
{klaus.kursawe, stefan.katzenbeisser}@philips.com

ABSTRACT

Recent investigations have found a massively increasing professionalisation and organization of attacks executed on consumer computing systems. Simultaneously, the systems we are trying to defend are getting more and more complex and networked, while promising security technologies—such as trusted boot and strong process isolation—appear to have troubles finding their way into mainstream devices.

This leads us to the conclusion that we may be forced to accept that the security war is lost for now, and that a considerable portion of all consumer PCs is under control of some organized malicious entity. In this work, we investigate the options left to the defenders in this scenario: Assuming that PC World is under control of a hostile force, how can we (a) survive (i.e., work) in a meaningful way, and (b) destroy the economic value for the attacker without severely damaging our own resources.

1. INTRODUCTION

The year is 2015 A.D. PC-Land is entirely occupied by the legions of Spam. Well, not entirely. One small village of indomitable users still holds out against the invaders. And life is not easy for the spammers who garrison the fortified camps of Windum, Linos, Phishinum and Wormus . . .

Most work in the area of computer security addresses the problems of protecting a computer system against attacks and providing measures for recovery once a system has been compromised. Those include, for example, research on virus scanners, software security and intrusion detection, secure backups, audit logs and patch distribution. For small PC-based platforms, in spite of great advances in security technology, this approach has shown limited success. The rates of corrupted computers are rising dramatically, with numbers claiming up to a quarter of all PCs on the Internet being controlled by some malicious entity.

In contrast to the typical attacks seen in the 1990's, which were mainly performed by individuals out of curiosity, we are increasingly witnessing large-scale attacks, driven by commercial motives, like large-scale data or identity theft, distribution of SPAM or the creation of bot-nets for Distributed Denial-of-Service (DDoS) attacks. A recent Symantec report [21] concludes that "organizations will have to deal with a well entrenched, experienced, and dedicated group of bot network owners instead of a population of hobby hackers."

In a worrisome tendency, the underground part of the computing industry has followed the example of the major companies and turned itself from a product into a service oriented industry [9]. For a moderate fee (starting at about $1000) one can purchase hand tailored Trojans, and for a little more the producers offer help with the distribution, server space to collect gathered data, and even a support hotline [12]. This underground industry has a surprisingly refined job structure; it contains the actual programmers, gatherers, service personnel, quality testers, etc. In addition, we see some competition between bot-net operators. Already now, bot-nets are competing against each other, i.e., bots try to protect an infected computer from competing bots in order to maintain a steady network.

In addition to enhanced and increasingly organized attacker activities, attack tools are getting more and more sophisticated [16]. Consider the example of Trojans: users do no longer need to actively take part in the distribution; a simple visit to the wrong website can install a Trojan by a drive-by-infection. This also makes it harder to react quickly—modern Trojans do not spread aggressively like the early worms (which quickly attracted attention), but may spread slowly and undetected; a recently discovered Trojan has been in the wild for at least 50 days before being found [12]. Even sophisticated techniques designed to defend computer systems can be 'hijacked' by attackers in order to automate the generation of new malware [19]. As another example, bot-nets are increasingly structured in a peer-to-peer fashion, and do not require a central control server. This makes it more difficult to shut them down.

Future technologies suggest even more trouble. Polymorphic attack software (which currently makes up about 3% of the malicious code [21]) can completely change its shape to the point that there is no more commonality between two instances of the attack program than between two random bit strings. Attacks on underlying libraries have led to security exploits that are largely browser independent, as all popular browsers build on the same base libraries. Virtual machine based root kits can hide deeply under the operating system and be practically undetectable unless the machine boots from an alternative source.

On the defending side, advances are progressing slowly. Trusted computing offers a chance to detect manipulations, but is adopted at a glacial speed and loses much of its potential in the presence of insecure operating systems. Malware scanners are continuously improving, but still it requires a major effort to keep them up-to-date; furthermore, even current virus scanners offer little security against fast-spreading malware.

1.1 The New Paradigm

Given the radical advances on organization and technology on the attacker side, we may end up in a situation where it is virtually impossible to keep an attacker out of a significant number of PCs worldwide. In this case, the focus will shift from *protecting* a computer system from attacks towards technologies that allow us to *cope* with infections and still offer some limited form of security, both for the attacked systems and the Internet as a whole. We thus propose to investigate a new paradigm for information security:

> *Assume that we lose the security war, so that a certain percentage of all PCs on the Internet are compromised through malware. Can we, even on 'occupied' computers, still offer some limited form of security? And, given that most attacks are economically motivated, can we destroy the business model of the attackers instead of beefing up security measures, in order to decrease the economic value of 'occupied' computers?*

Thus, in this paper we make the challenging assumption that attack technology will advance more quickly than defense methods, resulting in a large portion of all computer systems being controlled by some organized group of bot-net owners. We furthermore assume that the main motivation of the attackers is economical, in that they offer services—like data and identity theft, SPAM distribution and DDoS attacks—to third parties and collect revenues.

We propose to investigate the following two related problems in more detail:

- Assuming that a PC is controlled by a malicious entity, whose aim is to maximize profit out of its 'possession'. Can we—even in the presence of a malicious attacker—offer some limited form of security for the most valuable transactions (such as e-banking) or assets (such as private files)?
- Can we make the 'business' of the attackers less attractive by applying security technologies that are particularly tailored towards destroying the business model of the attackers? Rather than prohibiting an infection of a computer in the first place (which may be too challenging and costly), can we use technologies that make it economically unattractive to attack a system?

We believe that a systematic study of those two questions will become a central issue for next-generation Internet security mechanisms. In this paper, we first briefly sketch in Section 2 the business model of the attackers applied so far. Subsequently, in Section 3 we propose a number of tools that can be used on 'occupied' computers to provide security for the most precious tasks and documents and decrease the overall value of an 'occupied' computer for an attacker. The central goal of the techniques presented in the paper is to make it *cost-inefficient* to use the computer, rather than increasing the cost of gaining access in the first place. Here, cost-inefficiency can be achieved by causing cost (mostly due to enforcing human effort), but also by reducing the potential for the attacker to earn money by abusing the victim computer, or by increasing the risk for the attacker. Given the hardness of the problem, none of our approaches can provide a complete solution, and substantial work has to be done to make them work in practice; however, we do think that our ideas point in the right direction, and can inspire further work to undermine the economics of cybercrime.

1.2 Related Problems

'Computing under occupation' can be stated as the problem of securely performing some pre-determined task on an untrusted platform. There has been an increased interest—in various fields of computing—for solutions that enable the secure execution of code on a potentially untrusted platform; even though partial solutions have been obtained, many security problems are still open.

The most prominent field is Digital Rights Management (DRM). Here, some data (usually entertainment content) has to be rendered and displayed on the users' PC, while the system tries to prevent the user from obtaining a digital copy of the data. Even though the DRM player may be considered trustworthy, the underlying operating system may be not. This has led to a number of developments, including device-renderer authentication, code obfuscation, whitebox cryptography and means for revocation of compromised players. Similarly, copy protection of software attempts to assure that only legitimate software runs on a host, even if the attacker has full access over the system. No security solution exists so far that enables perfectly secure DRM or copy protection schemes; we rather witness an 'arms race', where system designers try to raise the level of effort required for an attacker to compromise a system.

Related questions arise in the field of grid computing, where a computation task is distributed onto a network of heterogeneous computers. If parts of the network cannot be trusted to perform the scheduled actions correctly and honestly, technological measures must be available to detect the dishonest network nodes and deprive them of critical information. Similarly, mobile agents, which travel between different hosts, face the problem of protecting their critical data from potentially compromised hosts.

Comparable problems arise in the field of online games as well. Here, an attacker—usually a dissatisfied player—tries to alter the game program to his advantage, for example by getting additional information or by automating some actions. Again, a presumably 'trustworthy' software (the game client) has to operate in a hostile environment. For example, bots can enhance the gaming capability of a user beyond a level that can be expected from a human; even implementation details, like the imperfections of the underlying data transmission network, repeatedly have been used to gain in-game advantages. In-game cheating threatens the economic value of game assets and hence has real-world economic implications.

2. ECONOMICS OF THE ATTACKER

While the early bot-nets appear to have been primarily run by amateurs because 'they could', modern bot-nets are professional enterprises, with the eventual goal of making money. Thus, great security benefit can be obtained from decreasing the economic value of the system for an attacker. If no money can be made out of a bot-net, attackers will eventually shift to more lucrative targets, as for example discussed in [8] for the case of Adware.

There are several possibilities for an attacker to convert resources of 'occupied' computers into financial gain (see also [2]):

- **System resources.** The computer can be used to perform computation, or more likely, the network interface can be used to extend the size of the bot-net, execute a denial of service attack, or send SPAM/Phishing mails. The users' computer can also serve as a distribution point for illegal data. As a relatively new form of fraud, the system resources can be used to simulate a user watching online advertisements, thus generating corresponding revenues.

  Economically, the rental fees for a *bot-week* (i.e., one corrupted computer for one week time) are between 2.5 and 6 cents for a spambot, or about $1 a week for the 'raw' bot [20, 13].

  A slight tendency appears to be that attackers are moving from the low margin commercial SPAM to the more profitable Phishing and stock manipulation SPAM; this might be due to increased competition and user awareness, and due to the fact that spamming is increasingly becoming illegal in the first place, making it easier for the attacker to go 'all the way' into illegality rather than maintaining some traces of honest business.[1]

- **Data theft.** Data on 'occupied' systems can be turned into money. Valuable data may include the identity of the users (including, but not limited to, credit card information), the users data on other people (e.g., email addresses of friends), or commercial data (e.g., corporate email stored on a personal computer). Credit card data can be sold for as much as $25 (with all credentials included) down to $6 (for the number alone).

- **Identity theft.** Another use for a stolen ID is to abuse the users reputation; a hacked eBay account, for example, can be used to make positive comments about a fraudulent dealer. Alternatively, the account can be used to perform virtual slander on a competitor.

- **User Transactions.** In a more advanced e-banking fraud, the attacker taps directly into the users' transactions with a bank and redirects authorized transactions; as most e-banking architectures do not allow a user to reliably verify the transaction she authorizes, there is little security against this attack. The authors are not aware of the monetary dimension of this problem. However, given the opportunities, this appears to be one of the more rewarding attacks.

- **User blackmailing.** Finally, the user can be held hostage and directly be forced to pay; the attacker may encrypt the hard-drive and only decrypt it on payment, or threaten to publish the users' private data. So far, most reported attacks of this kind were not overly advanced; the attackers made simple mistakes such as using the same password for all infected machines. More advanced attacks are sure to come and will use some of the more complex social engineering attacks [5].

  In the case of the Trojan *Trojan.PgPCoder*, the attacker demanded $200 to release the data; no numbers have been made available on the actual revenue of the attacker. On the non-monetary side, a first case of bullying a user into making explicit pictures has appeared already. So far, however, this seems to be a rather uncommon attack, and the attacker only used the obtained material for private purposes.

Apart from monetary gain, attackers may have different motivations that lead to indirect gain. Most prominent here is the spreading of the attack by using the corrupted platforms to identify and infect further platforms. Furthermore, the attacker can use the platform for anonymity—by providing a proxy for his own online activities or for storage of illegal data such as attack code or pornography.

2.1 Services

In addition to more refined attacks, attackers have diversified into specialized roles; in business terms, there is little vertical integration among the attackers. Rather, a service culture has evolved, including user hotlines. Attack Trojans targeting a special institute (e.g., a bank) can be acquired for about $1000 [12]; for a premium, one can also rent hijacked server space to gather the stolen data and hijacked websites to spread the Trojan. In addition, the attacker community is building up some substantial infrastructure to support the communication needed to connect the various services.

In the early days of cybercrime, the infrastructure largely consisted of messaging boards, such as the now defunct Shadowcrew forum. However, the services offered were already extensive; in a 2003 analysis, researchers describe fully automated services providing any user with (some) valid credit card numbers and user accounts, automatic credit card validity checks, and even merchant addresses susceptible to credit card fraud [17].

After the pressure on cybercrime increased, the infrastructure also started including security software for the protection against government agents; lately, a new instant messenger called CarderIM was distributed, allowing to build up a secure communication network, and anonymity seems to be provided by a TOR-like system.

While this specialization is a sign of more professionalism and thus increases the problem, it also may offer new ways of defense, as it requires additional logistics on the side of the attacker. As different, not necessarily mutually trusting parties are now involved (already, the first cases have been reported on credit card thieves scamming spam distributors), the stolen assets need to be easily transfer- and appraisable. This can be used, for example, by forcing the attacker to perform an online attack (making it harder to sell the assets), or to add fake data that will make it difficult to appraise the validity of the assets.

[1]An exception to the attack economics is the Nigeria or 419 Scam, in which SPAM email is used to find a victim for traditional fraud. Though response rates are low, successful attacks can earn the order of $100000. Examples to attack the economics behind this can be found on `www.419eaters.com`.

2.2 Attack Model

"Crime is often harder work than a regular job. Every time I've done something—it doesn't matter what it is, counterfeiting or whatever—it's always been more work than a regular job would have been. And I would have much rather had a real job than be involved in a criminal act because it's less . . . stress."

(David Thomas, member of 'Shadowcrew')

Our working assumption is that a victim is not subject to a targeted attack, i.e., the victim does not posses a special asset the attacker is after. Rather, the victim is a normal PC user indistinguishable from millions of others, with the same assets millions of others have.

Another working assumption is that critical infrastructures—such as banking portals—are secure. While this may not necessarily be true, one can assume that large service providers are significantly better protected than the average consumer PC, as a successful attack on one has grave consequences. For example, today it is much easier to compromise an arbitrary personal computer on the Internet than to hack into a well-protected server, due to ignorance against security measures and lack of expertise of the average home user. Thus, privately owned computers are much easier targets, likely to be attacked first.

As noted above, the ultimate goal of the attacker is to get some form of profit. In most cases the attacker strives for financial gain, though sometimes it is enough to obtain resources, such as storage and network capacity, for an illegal content distribution network or resources for a targeted attack on a high value target (e.g., the server of a competitor).

To defend against above attackers, we consider a dual strategy. For one, we try to limit the damage an attacker can do in spite of him having full control over the victims' computer. Secondly, we try to make attacks less cost efficient; ideally, one can increase the workload of the attacker in order to prevent economic gain. For consumer PCs, this is not implausible; the earnings on an average consumer PC are relatively low, and the business only pays off as attacks on millions of PCs simultaneously are cheap.

In future work, one may want to consider targeted attacks on corporate networks as well; however, in those cases different economic incentives are at work, as a single attack may yield substantial profit.

3. TOOLS

In this section, we give a preliminary inventory of tools that may be used to limit the damage inflicted on 'occupied' computers, and make it harder for an attacker to extract profit from an attack.

While full protection may not be possible 'under occupation', it is often not needed—we may concentrate on means that make an attack cost-inefficient, e.g., by forcing the attacker to apply human work that interferes with his business model.

3.1 Lies & Deception

There are basically two ways to protect critical data: Either it is not available to the attacker in the first place, or it is not recognizable by him as critical data. If unavailability is not an option, the user can try to provide enough fake data which is indistinguishable from the real data. This makes the whole data set worthless.

For example, the user can store a set of 100 authentic looking credit card numbers on the computer, 99 of which are not linked to any account. Currently, credit card dealers valuate a set of stolen credit card numbers by making test purchases with a small sample. If 99% of the numbers available are fake, this model does not work anymore, and the vendor needs to test every number individually (which also increases the risk of getting caught). With assistance of the credit card companies this model can be extended, and the fake data can be used as a canary. If any of the 99 fake numbers is used, we do know that the users' computer has been harvested for credit card numbers, and that the real number is in danger; this would be a good indicator to block the real number and replace it by a new one. (However, on the downside this opens the possibility of a denial-of-service attack). An unresolved issue here is how this data can be managed without creating too much overhead for the user. Obviously, it is not feasible to store on the platform which numbers are real and which ones are fake, as this information would be available to the attacker as well. Thus, it is not possible to make the system fully transparent to the user. A starting point is to protect easy targets, e.g., add bogus mail folders with credit card related fake mail, and some saved form fields in the browser that relate to non existing or never used websites.

A similar technique can be performed with any privacy relevant or commercial data; by placing a number of random pictures, letters, texts and email addresses on the computers hard-drive, the attacker will have a harder time to figure out what is critical, and has to apply costly human work to achieve his goal. (Note that some proactive detection tools for mailing worms already use bait e-mail addresses to detect infections with unknown malware, e.g. see [10]).

For more sophisticated attacks, an active form of lying is possible. For example, a phishing attack aiming for eBay accounts could be bombarded with realistic looking fake accounts; only on trying to actually use one of those accounts, the attacker would realize the fake. In a more advanced defense, phishers could be beaten with their own weapons. To this end, one could design a complete replica of a commercial site (such as eBay), and redirect all of the fake accounts there. The attacker can then happily perform transactions on the fake site, without ever having any relevance on the real world. Again, this can be combined with a canary technique; if a specific IP address uses a fake account, one can safely assume that this address is currently under control of some malicious entity and thus block all accounts used from this IP address for some time.

Of course none of these techniques are perfect; an attacker can observe the user behavior and thus find out which credit card numbers or email addresses are actually used, and careful testing will tell a good account from a bad one. This does cost time and effort, though, and the attacker may easily trigger a canary and thus render potentially valuable data useless.

3.2 Hardware Assistance

If a PC (and its operating system) cannot be trusted, a safe way seems to be to resort to trusted hardware, which cannot be changed by an attacker. While some approaches are being developed here (such as in the Trusted Computing Group), the adoption in commercial systems is frustratingly slow and full integration on consumer PCs will not be achieved in the coming years.

Thus, we concentrate on technologies that can be used in parallel to the operating system, without need for deep integration.

- **Specialized Firewall.** Most users have a pretty good idea when they need high speed Internet connections and when they don't. A specialized firewall can thus throttle the Internet speed at low hours to the extent that abuse becomes hard; it may still allow access to low resource services (e.g., for security updates or the users mail server), but prevent high throughput operations such as spam bots. By using certified services, the firewall can also selectively kill all packets going the wrong way, or delay them to the extent that the computer becomes too slow to be of any value to the attacker. This, however, would require knowledge of good or bad services, which seems a substantially harder problem to solve. Throttling of the network connection has already been explored in order to restrict spreading of viruses [23]; experimental results show that throttling indeed allows to limit the propagation of fast-spreading worms, while not impeding ordinary traffic. Similar methods can be applied for other types of attacks as well (for an early use see [18]).

  Unusual behavior (e.g., massive increase of bandwidth use without user interaction) may trigger an alarm and/or a limitation of the available bandwidth to the point that only basic services (e.g., the network timing, email polling) function properly; in a more complex variation, the firewall can also recognize protocols and thus block specially critical protocols such as SMTP.

  Another function can be the insertion of passwords and other critical data. In this case, the firewall serves as a proxy for secure applications; if a user connects to a secure website, it can replace placeholders with the appropriate password or the users credit card number. This way, critical data never needs to reside on the untrusted computer.

  One of the problems with a hardware module monitoring and manipulating Internet streams is encrypted traffic—if the hardware module cannot read a stream, it cannot operate on it. There are two basic ways to allow an external module to deal with encrypted traffic. For one, the module can build an encrypted channel to the application and a separate channel to the service provider. While this is relatively easy to implement, it has the disadvantage that the application itself cannot securely identify the service provider anymore; to this end, a separate application would be needed to allow the hardware module to check the certificates of the service provider and display it to the user. However, the approach has the advantage that it is implemented on protocol rather than TCP/IP level, so the hardware module can receive a full protocol message (e.g., an http request) before sending it on.

  The second possibility is that the application is modified to reveal its communication keys to the hardware module. As most standard protocols (such as SSL) generate random session keys, this needs to be an ongoing activity. The hardware module then can decrypt the data stream, and thus analyze and modify at leisure.

  In both cases, the hardware module has to reject all traffic it cannot decrypt, at least for known protocols (such as SOAP).

- **Write-protected memory.** Write protected memory is available rather cheaply today; examples are optical discs, or USB memory sticks with a physical write protection. More expensive solutions for PC hard drives exist as well, though they do not seem to penetrate the mass market. Write-protected memory can be used to safely store critical applications, which can be simple programs like a secure shell implementation or even a full operating system. In a more advanced setup, it may be possible to store the static parts of an operating system on a physically write-protected disc, while keeping the dynamic parts in ordinary re-writable memory. A virtual file system along the lines of UnionFS can merge the two and present a unified view on which standard operating systems can work.

- **Hardware based authentication.** Hardware-based authentication tokens, i.e., tokens that perform a small challenge response protocol with the service provider and thus prove that the authenticating party has physical access to the token, are already widely available. While currently mostly used for banking or cooperate networks, a recent field trial by PayPal raises hope that such tokens may become a commodity. Use of such tokens makes it impossible for an attacker to steal authentication information, forcing him to perform a significantly more complex online attack (i.e., manipulate valid transactions performed by the victim).

- **Hardware Backups.** A user who needs to ensure data availability will require a backup system that cannot possibly be disrupted by any software running on the PC itself. While it should be possible for software to put data onto the backup medium, deletion or manipulation must not be possible without physical access.

  A central problem with such hardware backups is the protection of the channel to the backup medium. If a file is transferred to an external storage device, it is always possible for an attacker to disrupt the communication, e.g., by encrypting the backup file. We see two ways to avoid this problem:

  - **Source backups.** The backup is made directly from incoming data, e.g., the TCP/IP packets or the keyboard strokes. While this destroys data structures and is inappropriate for some applications (e.g., Powerpoint documents whose creation require a lot of mouse action), it may supply a sufficient emergency backup to recover most data at reasonable cost, which would allow the owner to not be hold ransom in blackmailing attacks.
  - **Verified Backups.** In order to protect the channel to the backup medium, the backup server may offer a means of verification: the server tests that

the backup is still valid. An easy to implement, though not completely secure, way is to verify that data on the backup medium is syntactically valid; this allows the attacker to alter the data, but not to encrypt it. For full security, the backup server needs to report (and authenticate) the data back to the user; for example, it could convert parts of the backup documents into JPEG images which are shown to the user. In combination with a CAPTCHA-like challenge, as described below, this allows for effective data authentication.

- **Reinstallations.** The Swiss defense plan in case of an attack is rather unusual. Rather than stopping the attacking army, the plans suggest for the Swiss army to retreat to the mountains, where it will be impossible to beat. From there, it poses a constant threat to any occupying force, being able to reconquer the country as soon as the occupying army reduces numbers.

  Translated to PC security, the cyber-equivalent to the Swiss mountains can be a secure hardware part which is able to restore the operating system to its old, uncorrupted state as soon as the active attack is over. In an easy version, the secure hardware can simply be a CDROM; every once in a while, the entire operating system is replaced with a clean image from the write protected CD, leaving only the user data as it is. (Some institutions already follow this approach and reinstall PCs at risk periodically).

On a cautionary note, one should keep in mind though that a well funded attacker may well be in the position to perform some manipulations directly at the production site of the hardware; if such a system is to be widely used, precautions have to be made to ensure its trustworthiness.

3.3 CAPTCHAs

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are little tasks that should pose no problem for a human to solve, but which are assumed to be hard for a computer [1]; examples are the recognition of distorted letters or the content of a picture. Currently, CAPTCHAs are mainly used to assure that a service provider interacts with a human and not with an automated software bot. However, CAPTCHAs can also be used to establish a low-bandwidth communication channel between a human and a computer over an untrusted device. The CAPTCHA can encode, for example, authentication information solely intended for the human user, without the displaying computer being able to determine the embedded information. (Note that a similar technique is currently used by attackers: SPAM mail increasingly contains CAPTCHA-like images carrying the SPAM message in order to avoid automated filtering).

Unfortunately, CAPTCHAs have weaknesses as well. Apart from improvements in Artificial Intelligence that require more complicated CAPTCHAs, attackers have found and implemented a simple way to harness human computing power: To this end, the attackers offer free porn sites that only require the user to solve a small CAPTCHA to gain full access. The task given to the viewer of the site is the CAPTCHA that the attacker wants to have solved.

Though most CAPTCHAs can be broken if the attacker has enough time, we can still use them to authenticate data to the user (such as in the verified backup scenario described above). Suppose, for example, that a user has a physical security token that displays a time dependent pseudorandom value. Suppose further that a server wants to authenticate a certain confidential piece of data to the user, which should not be available to the displaying computer. The server can encode the data to be authenticated in a CAPTCHA-like manner, thereby adding a CAPTCHA challenge that is dependent on the user's pseudorandom value. This way, only a human user is able to view the authenticated information and only the human user who has access to the physical token can reply with the correct answer to the challenge. To spoof the authenticated information, the attacker would have to be able to embed the same CAPTCHA challenge in a different image; this has to be done fast enough to match the changing pseudorandom code, which forces the attacker to break the CAPTCHA quickly.

Other ways of human based authentication are possible, see for example [6, 11].

3.4 Neighborhood Watch

A central problem for protecting a computer from being abused as a SPAM sender is that the owner has little incentive to take action. A well written bot does not need to disrupt normal operations, and the user has little damage from sending SPAM. Thus, there are few reasons for an individual user to spend money to protect his computer from being infected by a spam bot. It should be noted though that it is not always necessary for security measures to be economical for an organization. For example, in a case of credit card fraud, the UK legislation forced banks to implement security measures whose price exceeded the losses it prevented. This was done as the overall cost to society (i.e., funding criminal organizations) was considered excessive.

One potential way of increasing user awareness could be a system of 'neighborhood watches'. Here, we utilize the fact that the attacks on computers are often independent of the geographic distribution—the way in which attackers gain access to the platform requires certain configurations or user behavior, and it is difficult for an attacker to coordinate the locations of the attacks. Thus, there are platforms that share one physical connection (e.g., a wireless node), but have a reasonable probability that not all of them are corrupted by the same organization. On a low level of the communication layer, they see each others data, even if they then choose to ignore it. This feature could be used for PCs on the line to watch each other. By analyzing traffic (i.e., looking for spoofed IP addresses or analyzing content or timing), the neighboring PCs can detect potential sources of SPAM and alert either the owner of the PC or the ISP to trigger a closer investigation (in effect, this would implement a kind of intrusion detection system, without requiring dedicated machines to observe the network).

Systems for neighborhood watch have been proposed recently to cope with worm infections, e.g., in Vigilante [7] or Sweeper [22]. In this context, several hosts collaboratively detect new vulnerabilities and distribute filters to block further infections. In a similar concept, collections of large numbers of instances of the same application ('application communities') may share the burden of monitoring for flaws and attacks and notify the rest of the community [14]; the same concept can also be applied to enhance attack software [15].

In addition, it is also possible to distribute certain critical tasks on different computers; this way, no single computer needs to store all critical data needed for some computation, requiring the attacker to perform an online attack rather than an offline one. Also, the different platforms can perform plausibility tests on each others operations. As a concrete example, access control information can be secret shared between the platforms, requiring every platform to do an operation for the number to be sent out. An attacker that has compromised only some of the nodes may be able to start a transaction (and ask the other platforms to finalize the computation), but cannot get the credentials from the platform. In addition, the secondary platforms can run some sort of intrusion detection, and raise a red flag if, for example, the credentials are used at a time the user usually is asleep.

While there is a vast literature on Byzantine fault tolerance and grid computing that addresses collaboration between mutually untrusted parties, little work has been done on applying those techniques to such mundane tasks as a credit card payment; while it seems plausible, this application does raise new questions such as dealing with unreachable parties (as opposed to normal peer-to-peer protocols it seems difficult here to choose random peers in the network, as the peer needs to have some secret related to the original platform) and privacy issues.

3.5 Storage Space

As long as an attacker in possession of a bot-net can distribute storage and computation effort onto the net, he has practically unlimited computation and storage power. However, if stolen data is sold on, it needs to be gathered at a central place; the customer of a data harvester will barely want the data distributed over a number of hacked machines, to which connecting to always poses some risk of being caught.

Therefore, space (and to some extent, computation time) can be used as a bottleneck; this has for example been proposed for e-cash [4]. In our case, we can make the authentication to some service requires a one gigabyte key. This is easy to implement, e.g., by using some huge lookup-table in a challenge–response authentication protocol. In this case, storage and transportation of 10,000 stolen authentication keys is a major problem, undermining the ability to gather authentication data and sell it on to a professional exploiter. The performance overhead on users side can be kept reasonably limited though, as it is not necessary to access the entire key every time; rather, every time the key is used, some different entries in the lookup table are accessed; this also allows scaling up the key size if storage becomes cheaper.

While storage price may not be a problem for the attacker, this mechanism makes efficient reselling access data difficult. In any cases, illegally obtained data is temporarily stored on hacked servers; increasing the storage requirement to the order of hundreds of Gigabyte (for a reasonably large set of access data) makes it harder to find appropriate servers, and increases the risk of detection, both during storage and transfer.

3.6 Software Security

In the worlds of DRM and online game security, a number of technologies are being developed to secure software programs running on a host that potentially cannot be trusted. The basic methods used are obfuscation (i.e., writing the code in a way that makes reverse engineering hard) and self checking code (i.e., code that detects tampering and shuts itself down if necessary). Unfortunately, it is hard to make solid statements on the security gained by such technologies, and to our knowledge no analysis has been performed on the effort required to break modern obfuscation and self-protection schemes; it has even been conjectured that secure obfuscation technologies do not exist [3].

It does seem possible, however, to automatically generate a large number of different obfuscated instances of a particular program in a way that no automatic reverse engineering of all instances is possible. An attacker, who wants to extract a secret out of the program (such as a private signature key) is thus forced to perform manual work for every program instance he attacks, rather than implementing one attack software, which automatically extracts secrets from all software images. This will make large-scale attacks uneconomical, as each individual piece of software needs to be manually broken. Examples for software that can be protected this way are banking access software or VPN clients containing critical keys.

4. CONCLUSIONS

In this paper, we made the challenging assumption that, given recent advances in attack technology, we may end up in a situation where a large portion of all consumer PCs is under control of some malicious entity. We proposed to investigate two new paradigms, namely how to provide some limited form of protection even on ‘occupied’ computers and how to reduce the economic value of an attacked system. The authors believe that a systematic study of these two questions will become a central issue in next-generation Internet security architectures.

Acknowledgements. The authors thank Michael Locasto, the anonymous reviewers and all participants of the NSPW’07 workshop for valuable comments that helped to sharpen the paradigm put forth in the paper.

5. REFERENCES

[1] The CAPTCHA project. `http://www.captcha.net`.

[2] Hemavathy Alanandam, Pravin Mittal, Avichal Singh, and Chris Fleizach. Cybercriminal activity. `http://www.cs.ucsd.edu/~cfleizac/WhiteTeam-CyberCrime.pdf`, 2006.

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, Salil Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology—CRYPTO’01*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[4] Bob Blakley. The emperor’s old armor. In *Proceedings of the 1996 Workshop on New Security Paradigms (NSPW’96)*, pages 2–16, New York, NY, USA, 1996. ACM.

[5] M. Bond and G. Danezis. A pact with the Devil. In *Proceedings of the 2006 Workshop on New Security Paradigms (NSPW’06)*. ACM Press, 2006.

[6] William Cheswick. Johnny can obfuscate: Beyond mother’s maiden name. In *First USENIX Workshop on Hot Topics in Security*, pages 31–36, 2006.

[7] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham. Vigilante: End-to-end containment of internet worms. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP'05)*, pages 133–147. ACM Press, 2005.

[8] Richard Ford and Sarah Gordon. Cent, five cent, ten cent, dollar: hitting botnets where it really hurts. In *Proceedings of the 2006 Workshop on New Security Paradigms (NSPW'06)*, pages 3–10, New York, NY, USA, 2007. ACM.

[9] Peter Gutmann. The commercial malware industry. `http://www.cs.auckland.ac.nz/~pgut001/pubs/malware_biz.pdf`.

[10] R. Hu and A. Mok. Detecting unknown massive mailing viruses using proactive methods. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004*, volume 3224 of *Lecture Notes in Computer Science*, pages 82–101. Springer, 2004.

[11] Collin Jackson, Dan Boneh, and Jon Mitchel. Transaction generators: Root kits for web. In *Second USENIX Workshop on Hot Topics in Security*, 2007.

[12] Don Jackson. Gozi trojan. `http://www.secureworks.com/research/threats/gozi/`, 2007.

[13] Carl Landwehr. Secure grid computing: An empirical view. `http://www.laas.fr/IFIPWG/Workshops&Meetings/48/WS1/10-Landwehr.pdf`, 2005.

[14] M. Locasto, S. Sidiroglou, and A. D. Keromytis:. Software self-healing using collaborative application communities. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2006)*, 2006.

[15] M. Locasto, A. Stavrou, and A. Keromytis. Dark application communities. In *Proceedings of the 2006 Workshop on New Security Paradigms (NSPW'06)*. ACM Press, 2006.

[16] McAfee. Virtual criminology report. `http://www.mcafee.com/us/local_content/white_papers/threat_center/wp_virtual_criminology_report_2007.pdf`, 2006.

[17] Bill McCarty. Automated identity theft. *IEEE Security and Privacy*, 01(5):89–92, 2003.

[18] R. Nelson. Unhelpfulness as a security policy or it's about time. In *Proceedings of the 1995 Workshop on New Security Paradigms (NSPW'95)*, pages 29–32. IEEE Press, 1995.

[19] C. Raiciu, M. Handley, and D. Rosenblum. Exploit hijacking: Side effects of smart defenses. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense (LSAD '06)*, pages 123–130, 2006.

[20] Stefan Savage. Unwanted traffic: Roots of the problem. `http://www.iab.org/about/workshops/unwantedtraffic/Session2_Stefan.pdf`, 2006.

[21] Symantec Internet security threat report, trends for July–December 2006. `http://www.symantec.com`.

[22] J. Tucek, S. Lu, C. Huang, S. Xanthos, Y. Zhou, J. Newsome, D. Brumley, and D. Song. Sweeper: A lightweight end-to-end system for defending against fast worms. In *Proceedings of the 2007 European Conference on Computer Systems (EuroSys'07)*, pages 115–128. ACM Press, 2007.

[23] M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *18th Annual Computer Security Applications Conference (ACSAC 2002)*, pages 61–68. IEEE Computer Society, 2002.