Reactive Security and Social Control

Lars Rasmusson, Andreas Rasmusson, Sverker Janson *
Swedish Institute of Computer Science

Untrusted Code

A major security problem for a network oriented environment is executing untrusted code. Private information risks being disclosed or tampered with if unverified remote code manages to gain access to local resources. Since a program only shows to the user what it wants the user to see, it can hide some of its actual actions. This is the essence of a *Trojan horse*.

In the days before global networking the acts of malicious programs mainly perpetrated random acts of vandalism, like erasing files. Now, as computers get increasingly connected, programs can communicate back to their creators. This enables a new range of crimes.

As we begin to use open computer networks to transfer information of more direct economic value, we'll find that programs can to do more malicious things than erasing files. Viruses can be used to snoop passwords to valuable information services or getting hold of e-cash stored on our hard disks. As the trend leads towards where we are down-loading and executing many new programs every day, these problems are only likely to increase.

Certifying every program on the Internet would hinder the introduction of new programs, services and even of bug-fixes. The essence of the open net is that new information is put there almost instantaneously. Hence we can't do away with the concept of an untrusted program.

Cryptographic methods like digital signatures can be used to authenticate the sender and/or guarantee that the program hasn't been tampered with. However, the program's hostility cannot be decided by any level of cryptography.

For the untrusted software to be useful it may have to be granted access to information that it potentially can misuse. There is a notion of *risk* involved in dealing with untrusted code, and this is not well supported in conventional computer security. Two ways to deal with the risks are

- to use a system where trust/distrust is an integral part of the system
- runtime monitoring of the untrusted code and decision support for the user

These methods belong to a class of security mechanisms we call *soft* security. Soft security, as opposed to *hard*, means that privileges are granted as they are needed, with the current risks taken into consideration. Hard security denotes methods that don't reevaluate granted privileges.

Soft security is related to reactive, "after-the-fact", security and intrusion detection. The term reactive emphasizes the *when* the analysis is done whereas *soft* is an indication of on what grounds resource access is granted, hence the new term.

Why are there malicious programs?

Some crimes (like occasional speeding, or some white collar crimes) can be said to be *rational* in a game theoretic sense. This means that the expected net payoff is greater than zero, after considering risk of being caught, expected punishment and expected gain [1]. If we radically change the properties of an economic system (as ours will be changed by Internet) we might find that a number of new crimes will be economically "sound."

But apart from programs written in evil spite, like Trojan horses or viruses, a program can also start to misbehave because of bugs or because it is being used in a context for which it was not designed. An interconnected ever-changing program environment makes it virtually impossible

*Email: {lra, ara, sverker}@sics.se

for the programmer to understand all the effects of his/her program. Therefore it seems wise to treat all programs with a little caution, regardless of the author's intentions.

Reputation and anonymity instead of blind trust

One way to help a user to minimize the risk of using untrusted software is to use reputation mechanisms [2]. Reputation enables us to dare to take larger risks with the programs we believe are benevolent. Microsoft's reputation makes us dare to install the annual version of MS Word without first verifying its source code. But for Internet systems dealing with lots of untrusted code reputation mechanisms need to be made explicit.

The importance of being able to trust one's business partner is beginning to be acknowledged on the Internet. Certification companies and authorities that act as *Trusted Third Part*s are proliferating on the net. A trusted third part acts as a guarantor for the seriousness of the other part. However, authoritative trust has some drawbacks. It is centralized and hierarchical and it puts both parties in the hands of the trusted third. It ends up in a circular reasoning; "How do I trust the trusted part?"

What we are looking for is a system where all parties actively cooperate to build up reputation, and where reputation is built on rational grounds. Further, it should not be necessary to keep global registers over every person on the Internet, since it is both impossible and violates personal integrity.

Anonymity

Part of tomorrow's business will be conducted by programs. Unlike ordinary companies or persons, a program does not have any physical manifestation that guarantees that it will be around for a longer time. It can be copied infinitely, or changed into unrecognizability. If a program put on the Internet cannot be traced back to an originator, it is effectively an anonymous program. No-one can be held responsible for its actions. With complete anonymity, selling stolen information or goods, computer break-ins without risk of being caught, or plain vandalism (making other peoples computers crash, etc.) can be safely committed.

The converse, complete identification in all steps, is also susceptible to new crimes. Complete logging of someone can generate a computer shadow that could be used for annoying advertising or blackmailing, for break-ins ("locate persons who have bought a new VCR and who are at work") etc.

To anonymity, two approaches are possible. Either say "let's just forbid anonymous programs - everything must be traceable back to the originator," a common view. Or say "we can't prevent anonymous programs - we must therefore design our system so that it doesn't collapse if anonymous programs slip in."

The former view is unacceptable since there is no way to "forbid" some programs in an open network such as Internet to which anyone can connect their computer. If we insisted and designed such a system, it would be very vulnerable if someone released such a program in the system. It's not a good idea to construct open systems so they only work if all the other components work as intended.

Reputation demands identity

In a reputation based system identity is something valuable. Reputation coupled to an identity enables two parties to make business together, something from which they both benefit. Loosing one's reputation equals a loss of income from other business. It becomes irrational and costly to waste one's reputation by malicious behaviour.

Unforgeable identities don't have to be created by an identity issuing authority. They can be created by anyone using digital signatures, or zero-knowledge (interactive) protocols. The id works as an unforgeable trade mark, and reputation is

established when the same id is used more than once. Anonymity is achieved by creating a new id for every transaction.

If we manage to construct a system where the interacting programs are "suspicious" to one another and don't expect other programs to behave nicer than their reputation guarantees, we will in fact have a system that can support anonymous programs. It will not have drawbacks such as complete logging of every person, or centralized trust servers. It acts cautiously if new (possibly buggy or malicious) components are added, but once they have merited a certain amount of trust, they are integrated into the system.

Behavior-based resource granting

Traditionally operating systems limit user access to system resources by enforcing access rules in system calls. Ordinary read/write access control and *capability systems* are examples of this. Once granted permission, the program has free access to the resource.

These security barriers are necessary but not sufficient in a networked environment where programs are exchanged promiscuously between computers. Capability systems do not solve the problems of denial-of-service attacks or of leaking information, since they are just a means to decrease granularity for the privilege assignment.

Many useful restrictions are not possible to enforce before runtime. For instance, forbidding simultaneous access to a shared resource by two or more programs inhibits potential covert channels between the programs. Whether covert communication will take place or what information might be leaked depends on the actual situation. It probably doesn't matter as much if your word processor leaks the contents of your shopping list as if it leaks your business mail.

Assigning correct privileges to a program requires the user's afterthought and skill and will be very cumbersome as the number of programs we interact with each day increases. Instead of just classifying the resources, untrusted programs could be classified by their expected behavior. This would constrain the range of expected "normal" actions and making it harder for a program to undetected do something unexpected/malicious. We are studying how to give automated support for deciding if and how a program deviates from its expected behaviour [3].

Different implementations of a service could, if similar enough, be classified as belonging to the same class. Behaviour classes reduce the number of choices the user has to make since the user needn't be aware of all rules a particular behavior implies. Since a violated rule can be explained in behavioral terms it is easy to understand and giving the user decision-support for how to handle the situation.

Automatically communicating and updating behaviour descriptions, will make the society of Internet hosts more resilient to malicious programs.

Conclusions

We need to remove the obstacles for an open Internet with commercial interests without using centralized solutions. Trust relationships could be used to assess the economical risks of engaging in activities with unknown parties. To inhibit malicious programs from covert activities we suggest runtime monitoring for constraining the allowed behavior of programs.

References

[1] Economic Times *The Economics of Crime*, journal, vol 4, no.1, 1995. Addison-Wesley.

[2] Rasmusson, Lars & Janson, Sverker *Simulated Social Control for Secure Internet Commerce* New Security Paradigms Workshop '96, 1996.

[3] Rasmusson, Andreas & Janson, Sverker *Personal Security Assistant for Secure Internet Commerce* New Security Paradigms Workshop '96, 1996.