MIS-USE CASES FOR THE GRID*

D. Skow#, FNAL, Batavia, IL 60510, USA

Abstract

Development of expected use cases is a common early step in the design phase of developing new technologies. The author argues for the creation of mis-use cases as well. These help guide the developers in understanding the types of protections and controls needed so the various stakeholders can effectively balance their risk tolerance with expected gain.

This document presents an extrapolation of one current attack pattern to a Grid environment and a discussion of the types of controls and developments needed to provide a robust Grid on today's Internet.

INTRODUCTION

It is clear that the general Internet environment has changed substantially from that which gave birth to the Web in the early 1990's. The user population has exploded and the abuser population has grown with it. For our own system reliability, as well as limiting the liability our resources will be misused to harm others, we have to take measures to defend against and respond to attacks. No system is foolproof and so tradeoffs must be made. We need to "know the enemy" in order to make appropriate tradeoffs based on real assessments of risk rather than wishful thinking. We also need to design ahead as rapid modification of production systems is extremely difficult.

Know the Enemy

The stereotype of the teenage male bent on exploring all corners of the Internet is a common perception of the attacker community. There's evidence from the few successful prosecutions that end up identifying the source of attacks that this is indeed an active population. The attacks are usually characterized by exploration and rarely intentionally harmful. One has a hard time attaching the word "enemy" to this population, "pest" comes more naturally, and many people consider this and annoying but relatively minor activity, like graffiti.

However, just as the automation of vulnerability probes lead to the curse of the Internet worm, so too the Grid offers new possibilities for automation and distribution of malicious software (malware).

We must recognize, as will the bad guys, the attraction of "getting something for nothing" that the Grid offers. If we build a successful infrastructure, they will come. We must be prepared to deal with the pests short of burning down the house.

Who is the "enemy"? Certainly it includes the teenage vandal. It also includes the legitimate user who's made a mistake that is rapidly replicated across the Grid. It includes experimenters trying out new ideas on the production Grid, which induce problems and/or failure. It includes criminals looking to hide their tracks and/or exploit resources. The larger the pool of resources that are available to a successful attack, the more incentive and larger the population of attackers.

How will they attack ?

We can predict the initial attacks by straightforward extrapolation of current trends on the general Internet. In fact, all grids on the Internet are under attack today from exploits using vulnerabilities in common software (e.g. the operating system, common libraries, …).

We can also predict with some confidence that the progression of attack types will likely recapitulate the Internet attacks as the Grid becomes more ubiquitous. First methods will likely be to hijack credentials (analogous to cleartext password attacks), next exploit vulnerabilities in commonly deployed software, and then exploit of grid management/forensics tools and/or custom development of attack tools. (It is interesting to note that there is a great deal of technical overlap between the two. It is probably the case that advances in one helps the other. Good tools however are essential to efficient, affordable operations.)

LIFECYCLE OF A GRID WORM

Consider the example of a Grid worm. Biological analogies are often quite apt in describing malware. A worm is a piece of software that explores its environment and tries to automatically exploit available resources. This is uncomfortably close to a description of an opportunistic Grid job – one of the common use cases given for Grid. We need to understand the mis-use case, how to distinguish it from the legitimate use case, and build in controls lest the parasite kill the host.

A generic worm has three distinct stages of life: birth (the insertion of basic executable), growth (acquisition of privileges), and reproduction (propagation). Each phase offers opportunities for defence and detection.

Birth

Somehow the worm executable has to be initially invoked. This may or may not be the same method that the worm later uses to spread. In fact, several methods are usually tried for both steps. Common methods of insertion include: credential/session hijack, command insertion, trojan software, and user enticement.

If the bad guy can obtain the credentials of a legitimate user, then s/he can start the worm executable directly as

that user with whatever fights the user has. If the badguy can hijack an established session, that suffices as well in most cases. The defence focuses on strong authentication and authorization systems. Authentication (e.g. "Halt! Who goes there?") systems must protect against exposure and/or theft of authentication secrets (or tokens directly). Authorization ("Advance, and be recognized.") systems must allow for compromised identities to be stopped. Death can come from a single hit in a vital organ as well as a thousand nicks.

Buffer overflow is the most commonly known method of exploiting and application to insert an unintended command. There are several other standard exploits to achieve the same end (e.g. replacement of temporary files, spoofed data, etc.). The defences against these at the developers' level are largely well known also (e.g. check all inputs for validity, check return codes, avoid race conditions, etc.). They must become common practice in all widely deployed software to close this pathway. From as system management standpoint the least vulnerable service is one you don't run. Unneeded services should be turned off and those running should be regularly monitored and patched.

Alternately, the badguy can attempt to have an authorized user execute his/her program. Common methods are to replace an expected executable with modified copy (the Trojan horse) or to somehow entice the user to execute code provided by the bad guy (e.g. embedded weblinks in email or on other webpages). Defence relies on educated and alert users. Detection is also hard and focuses on accounting turning up unexpected modifications and/or usage.

Growth

In most cases, the initial insertion is a limited toehold into the compromised system. Quite often the worm needs to obtain more information or privileges to effectively spread. Many worms have very small initial executables, which then download more extensive attack software. Privilege collection may be "getting root," looking for unprotected user credentials (e.g. private key or proxy collection) or both. This information collection phase often remains running until the worm is found and removed, occasionally sending on information it collects

Defensive measures are similar to the previous phase, but one now has to also defend against those exploits that are not effective over the network (local exploits). One can frequently find these ongoing collection processes by monitoring accounting logs and looking for unusual duration or network connections. Of course, this requires that logs are secure against tampering (a commonly attempted trick worms use to hide their traces) and are monitored.

Reproduction

A worm which doesn't rapidly reproduce is of limited concern – though the first clouds of mosquitoes of summer remind the author that if the initial infestation is prolific enough, there's no need for reproduction to drive one crazy. Current worms often attempt multiple methods of reproduction and they may well learn (utilizing locally collected information or updates from "mother"). Biological parasites have learned that too aggressive propagation destroys the host environment (themselves along with it) and there are indications that the electronic varieties are learning this too.

Defences require defendable points. Not all bottlenecks are bad. They provide points where defenders can ward off more numerous attackers and contain the losses. Strategically placed throttle points in software and in networks can provide places where alarms on unusual utilization can be raised and where controls can be imposed. These controls will need to be automated to be effective against automated attacks and they will need to be applied both inbound (to defend against infection) as well as outbound (to contain spread). There may be possibilities in the wider use of IPSEC and/or dynamic network access control to help in this area.

Similarly, diversity increases survival odds of the species. Yes, this has costs in efficiency, but homogenous systems are particularly vulnerable to wide-scale exploit of common vulnerabilities. A moderate level of diversity should be encouraged both to aid in the discovery of errors and vulnerabilities as well as increasing the odds of continued functionality of some portion of the ensemble.

Death ?

Individual biological worms eventually die, and yes, most electronic worms are eventually found and killed as well and the biological analogy continues to be apt. We may, with heroic effort and a global program, be able to eradicate some type of worm from the world, but we are unlikely to be able to eliminate parasites and we reintroduction remains a constant threat whenever there is an un-inoculated population.

CONCLUSIONS

Like the first day of school, participation in the Grid holds great promise, but it also exposes the participants to a much wider world of "germs". We should expect that "colds" will be exchanged and we should prepare for dealing with them while the antibodies are developed.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the benefit from a great number of conversations with colleagues particularly in the GGF, DOE and HEPiX meetings. Matt Crawford is particularly thanked for presenting this talk when the author was unable to attend the conference. The book "Beyond Fear"[1] is recommended as a particularly good overview of security analysis.

REFERENCES

[1] B. Schneier, Beyond Fear, Copernicus Books, ISBN 0-387-02620-7, 2003.