Full Disclosure

What is full disclosure? Full disclosure refers to the computer security concept that releasing complete details of a computer vulnerability to the public is the best way to get vendors to fix security problems.

"Full disclosure is, in and of itself, a means to an end. Many people participate in it, not out of malice but out of the hope that, with enough public scrutiny vendors, they *(sic)* will finally take responsibility for the software they write. Make no mistake about it: Full disclosure is ugly. People get hurt. However, I see no reasonable alternative. For every responsible bug reporter out there, it's likely he / she has a counterpart who will most likely keep the information for themselves and use to ends that we would all rather avoid. In a perfect world, vendors would perform rigorous security audits, previous to market release. In a perfect world we would still have buggy software (this is something we will never lose) but we would also have vendors who make security a pre-emptive consideration as opposed to a forced reaction." - Al Huger (Security Focus)

There are several distinct steps that occur during a vulnerability discovery.

- The vulnerability exists in the software, but it has not been discovered.
- The vulnerability is discovered (by a researcher, company, attacker, etc.), but the information has not yet been disclosed beyond the original researchers.
- The vulnerability is announced either to a public security mailing list or has become common knowledge among the security and/or underground community.
- The vulnerability is coded into an easy to use exploit.
- The vendor develops a patch for the vulnerability
- Users and administrators patch the vulnerability.

There is a *window of exposure* and significant danger to users when an easy to use exploit exists, but the existing systems are not patched. This window of exposure is what computer security professionals are striving to eliminate.

The Opponents of Full Disclosure

Most users are not security conscious and they are connected to the Internet with minimal defenses. If a security vulnerability is released to the public with all the details it spreads quickly through the underground and there is a flood of attacks against the system. If users ever bother to patch their systems, it takes them a long time.

Full disclosure allows “script kiddies” without any real knowledge to use point-and-click attacks of the latest exploit to wreak havoc on regular users. It also gives attackers with real knowledge the information that they need to develop an exploit. If the vulnerability details had not been published, there may have never been an attack performed using the specific vulnerability.

Most of the security holes that are being exploited today are categories of holes that are already well known and very seldom does full disclosure of a vulnerability add new information about the types of holes that exist.

It may be possible to limit vulnerability information and exploit tools to security professionals. If we make the tools illegal and hold the creator of a tool responsible for what is done with the tools, we can dramatically reduce the number of "script kiddies" attacking our networks. Individuals and corporations who irresponsibly distribute point-and-click attacks and exploit code should be held accountable for damages that can be traced back to their tools. These guys are not part of the solution.

We can also reduce window of exposure by limiting vulnerability information available to public. Allow a vendor to develop and distribute a patch. Inform administrators that there is a security vulnerability, how to check if they are vulnerable, and how to patch it.

We all want to protect the users and more users are harmed by full disclosure than are helped.

The Supporters of Full Disclosure

Many vendors refuse to acknowledge a vulnerability without the pressure of full disclosure. Without full disclosure, if a vendor refuses to issue a patch for a vulnerability, you'd be left with no recourse and stuck with an insecure system. Full disclosure is not a panacea. Some users will be harmed during the window of exposure that we discussed earlier. The goal of full disclosure is to motivate vendors and to encourage companies to work with security researchers on reducing the window of vulnerability.

Product vulnerabilities need to be documented and published. If the vulnerability details are published, then developers all over the world can learn from them and try not to make the same mistakes. If a vendor publishes a patch for a vulnerability without disclosing the details of the vulnerability, an attacker can spend the time to reverse engineer the patch and discover a way to exploit the vulnerability. An overworked administrator usually doesn't have the time to do much more that slap on a patch that "does something to improve security". System administrators need some form of exploit code to test the vulnerability and also test the vendor solution. If the details of the vulnerability were published, an administrator might notice that a similar vulnerability exists in other systems on their network.

Who's responsible for fixing software that the vendor no longer supports or even if the vendor no longer exists (abandoned code)? This is a much more difficult situation because there is no vendor to pressure into fixing the problem. If the vulnerability can be fixed through some sort of administrative change, then it is appropriate to release the vulnerability. If source code to the program is available, you can probably get the problem fixed. If the source code is not available, the only real solution is to switch to a different program. This is not always a realistic solution and you need to balance the thin thread of obscurity against the probability that, even after the vulnerability is revealed, a large percentage of the users will continue using the insecure program.

Hiding, denying, and obfuscating the vulnerability does not make the vulnerability go away! Neither does attacking the security researcher. The vendor created the security vulnerability, not the researcher. The vulnerability exists and burying our heads in the sand will not protect us.

The vulnerability details may already be in circulation in the "computer underground". Full disclosure will even the odds in the battle between attackers and defenders in this case. It is possible that someone will publish the vulnerability details for personal fame, but the attackers won't be giving them to the vendors anytime soon.

New developers need to learn from the mistakes others have made. Vendors need to take security vulnerabilities seriously. This is not happening because vendors continue to escape legal liability and responsibility for a total lack of security. If computer security is going to continue improving, then vendors must be held accountable for their poor security practices.

"If the vulnerability details had not been published, there may never have been a single attack performed using the specific vulnerability."

> The problem is that you can never be sure of that until there are no more affected systems in use anywhere. There is also a significant risk that the really dangerous attackers are never noticed, because they are careful to hide their presence.

"It may be possible to limit vulnerability information and exploit tools to security professionals."

> It's impossible to keep a secret among thousands of security professionals and once someone in the underground has the vulnerability information, it spreads like wildfire. Information naturally disseminates and more people are helped by responsible disclosure than are harmed by it.

"Full disclosure allows "script kiddies" without any real knowledge to use point-and-click attacks of the latest exploit to wreak havoc on regular users. If the details had been kept secret, they would not have the resources to do it."

> The script-kiddies are really only a decoy in the security game. Current patches will protect against most "script kiddies". Since they, by definition, don't have deep knowledge about security or computer systems they are limited to simple attacks. Your most sensitive information and systems should not be vulnerable to a single point-and-click exploit. The most dangerous attackers will be able to create their own exploits without the complete vulnerability details. These are the people who will be going after the "crown jewel" of your organization.

We can reduce the window of exposure by encouraging and supporting vendors who take security seriously, educating security professionals and system administrators about the latest exploits, and by refusing to tolerate vendors who have a history of poor security response.

Researchers who discover vulnerabilities are doing all of us an important public service. They help keep software vendors honest and help everyone learn to avoid similar mistakes.

What is a Reasonable Middle Ground?

Most people want the same thing, systems and networks that are secure, but disagree on the solution. How and when we disclose the facts is the most crucial part of full disclosure. Many factors come into play, the seriousness and implications of the bug, how long has it been since it was discovered, how responsive is the vendor and how dependant are we on the vendor for a patch.

I would encourage everyone to read Rain Forest Puppy's RFPolicy (Appendix A). It clearly defines reasonable guidelines for communication between security researchers and vendors. It also gives recommendations on how to contact vendors, educate the public, and timeframes to disclose vulnerabilities.

Vendors should be given a reasonable chance to provide a patch or new version before the vulnerability details are made public. When releasing the vulnerability details they should be released *completely*. The attackers usually have a lot of spare time to figure out the missing parts, but busy administrators usually don't. The vulnerability details should be published in places where they reach the largest possible group of security people. Publishing the information only in obscure places, increases the risk that attackers will exploit it before anybody has the chance to fix the problems.

The vendors need to provide sufficient information to the public so people who discover vulnerabilities can contact them. If a security researcher contacts a vendor, please take them seriously and treat them with respect. These researchers are often working on their own time to check out your product. A poor response (theoretical vulnerability, not our problem, not crediting researchers) will discourage researchers from coming to vendors in the future. Vendors need to develop a security response procedure. At a minimum, create the security email addresses mentioned in RFPolicy and monitor them.

We need to understand that attackers have access to just as much information as security professionals. As our networked environment becomes more complex, we will continue to have vulnerabilities. However, a single vulnerability should not expose all of our information and vulnerability response is just one component of a layered security environment.

Resources

Rancum, Marcus. “The Slaughter of Innocents.” URL: http://pubweb.nfr.net/~mjr/usenix/ranum_5.pdf (20 November 2000).

Rancum, Marcus. “Have a cocktail: Computer Security Today.” URL: http://pubweb.nfr.net/~mjr/usenix/ranum_elx_cocktail.pdf (20 November 2000).

Weld Pond. “Do security holes demand full disclosure?” 16 August 2000. URL: http://www.zdnet.com/zdnn/stories/comment/0,5859,2615973,00.html (20 November 2000)

McClure, Stuart. Scambray, Joel. “Anti-hacking method of full disclosure under attack from a part of the security industry.” 11 August 2000. URL: http://www.infoworld.com/articles/op/xml/00/08/14/000814opswatch.xml (20 November, 2000).

Schneier, Bruce. “Full Disclosure and the Window of Exposure.” CRYPTO-GRAM. 15 September 2000. URL: http://www.counterpane.com/crypto-gram-0009.html#1

Appendix A

////// **Full Disclosure Policy (RFPolicy) v2.0** //////
This policy is available at http://www.wiretrip.net/rfp/policy.html
\\\ **Executive overview for vendors and software maintainers** \\\
This policy states the 'guidelines' that an individual intends to follow. You basically have 5 days (read below for the definitions and semantics of what is considered a 'day') to return contact to the individual, and must keep in contact with them *at least* every 5 days. Failure to do so will discourage them from working with you and encourage them to publicly disclose the security problem.
This policy is not set in stone--in fact, it is encouraged that all parties regularly communicate with each during the process, adjusting as situations arise.
\\\ **Table of contents** \\\
Purpose of this policy
Policy definitions
Policy
Detailed/commented explanation of policy
Difference between version 1 and version 2 of RFPolicy
RFPolicy FAQ
Using this policy
Credits
\\\ **Purpose of this policy** \\\
This policy exists to establish a guideline for interaction between a researcher and software maintainer. It serves to quash assumptions and clearly define intentions, so that both parties may immediately and effectively gauge the problem, produce a solution, and disclose the vulnerability.
First and foremost, a wake-up call to the software maintainer: the researcher has chosen to NOT immediately disclose the problem, but rather make an effort to work with you. This is a choice they did not have to make, and a choice that hopefully you will respect and accept accordingly.
The goal of following this policy, above all else, is education:
Education of the vendor to the problem (ISSUE, as defined below).
Education of the researcher on how the vendor intends to fix the problem, and what caveats might cause a solution to be delayed.
Education of the community of the problem, and hopefully a resolution.
With education, through continued communication between the researcher and software maintainer, it allows both parties to see where the other one is coming from. Coupled with compensation*, the experience is then beneficial to the researcher, vendor, and community. Win/win/win for everybody. :)
(*Compensation is meant to include credit for discovery of the ISSUE, and perhaps in some cases, encouragement from the vendor to continue research, which might include product updates, premier technical subscriptions, etc. Monetary compensation, or any situation that could be misconstrued as extortion, is highly discouraged.)
\\\ **Policy definitions** \\\

The ISSUE is the vulnerability, problem, or otherwise reason for contact and communication.
The ORIGINATOR is the individual or group submitting the ISSUE.
The MAINTAINER is the individual, group, or vendor that maintains the software, hardware, or resources that are related to the ISSUE.
The DATE OF CONTACT is the point in time when the ORIGINATOR contacts the MAINTAINER.
All dates, times, and time zones are relative to the ORIGINATOR.
A work day is generally defined in respect to the ORIGINATOR.
\\\ **Policy** \\\
A. The ORIGINATOR will send email regarding the ISSUE to the MAINTAINER; the point in time when email is sent from the ORIGINATOR is considered the DATE OF CONTACT.
It is important that the ORIGINATOR review any documentation included with the object of the ISSUE for indication of a proper method of contact. That failing, the ORIGINATOR should check the web site of the MAINTAINER for methods of contact. Should the ORIGINATOR not be able to locate a suitable email address for the MAINTAINER, the ORIGINATOR should address the ISSUE to:
security-alert@[MAINTAINER]
secure@[MAINTAINER]
security@[MAINTAINER]
support@[MAINTAINER]
info@[MAINTAINER]
regardless of their existence. Anyone who could be deemed as a 'MAINTAINER' is encouraged to populate at least some of the above email addresses. Email auto-responses should not be considered as a message from the MAINTAINER.
Note: addressing the ISSUE to InterNIC handles may cause the email to be misdirected (for example, to a virtual hosting company who happens to host the MAINTAINER's web site). Addressing the ISSUE to the above listed email addresses may cause the email to be received by non-authoritative persons (for example, to an online service provider who happens to have an user named 'security-alert').
B. The MAINTAINER is to be given 5 working days (in respects to the ORIGINATOR) from the DATE OF CONTACT; should no contact occur by the end of 5 working days, the ORIGINATOR should disclose the ISSUE. Should the MAINTAINER contact the ORIGINATOR within the 5 working days, it is at the discretion of the ORIGINATOR to delay disclosure past 5 working days. The decision to delay should be passed upon active communication between the ORIGINATOR and MAINTAINER.
C. Requests from the MAINTAINER for help in reproducing problems or for additional information should be honored by the ORIGINATOR. The ORIGINATOR is encouraged to delay disclosure of the ISSUE if the MAINTAINER provides feasible reasons for requiring so.
D. If the MAINTAINER goes beyond 5 working days without any communication to the ORIGINATOR, the ORIGINATOR may choose to disclose the ISSUE. The MAINTAINER is responsible for providing regular status updates (regarding the resolution of the ISSUE) at least once every 5 working days.

E. In respect for the ORIGINATOR following this policy, the MAINTAINER is encouraged to provide proper credit to the ORIGINATOR for doing so. Failure to document credit to the ORIGINATOR may leave the ORIGINATOR unwilling to follow this policy with the same MAINTAINER on future issues, at the ORIGINATOR's discretion. Suggested (minimal) credit would be:
"Credit to [ORIGINATOR] for disclosing the problem to [MAINTAINER]."
F. The MAINTAINER is encouraged to coordinate a joint public release/disclosure with the ORIGINATOR, so that advisories of problem and resolution can be made available together.
G. If the ISSUE is publicly disclosed, by a third-party, the ORIGINATOR is encouraged to discuss the current status of the ISSUE with the MAINTAINER; based on that discussion, the ORIGINATOR may choose to disclose the ISSUE. The MAINTAINER is encouraged to credit the ORIGINATOR for discovering the ISSUE. Should the MAINTAINER disclose the ISSUE, or items supporting/relating to the ISSUE (patches, fixes, etc), the ORIGINATOR may choose to disclose the ISSUE.

\\\ **Detailed/commented explanation of policy** \\\

This section serves to elaborate on the items in the policy, for better understanding.
A. Pretty self explanatory--the ORIGINATOR is to email the MAINTAINER about the problem. The ORIGINATOR should do their homework and try to find the correct address to email (by checking the MAINTAINER's web site, by looking in documentation distributed with the software/product, etc). Emailing InterNIC handles or addresses such as 'postmaster' or 'webmaster' is not good, since they are most likely IT support staff and not the proper representatives to handle such a situation.
B. The MAINTAINER has 5 work days respond. Note that all times of work days are relative to the ORIGINATOR, not the MAINTAINER. Suggestion to the MAINTAINER: sooner is better than later--just because you have 5 days does not mean you need to take them all. The ORIGINATOR is technically free to do whatever they want to do after 5 work days--however, they should be fair and wait if the MAINTAINER shows adequate initiative to fix the ISSUE.
C. Just as the MAINTAINER shouldn't ignore the ORIGINATOR, neither should the ORIGINATOR ignore the MAINTAINER. The ORIGINATOR should help the MAINTAINER recreate the problem, if necessary. It's probably in the best interest of the ORIGINATOR to help the MAINTAINER confirm the problem--otherwise, the ORIGINATOR stands to disclose a potentially false ISSUE.
D. The MAINTAINER has to actively give status reports. Note that it's the MAINTAINER's responsibility to do so, and not the ORIGINATOR's responsibility to request them.
E. If the ORIGINATOR does indeed take the time to follow this policy, they should be acknowledged not only for doing so, but in general, acknowledged for finding the problem. There are proper ways to cite references, credit sources, and otherwise respect the origination of information--I suggest vendors do the same. If you can not respect the ORIGINATOR enough for taking the time to notify you of the ISSUE, the ORIGINATOR (and possibly others) may feel reluctant to follow this policy with the same MAINTAINER in the future.

F. Making the problem and solution advisories available together allows the community to have immediate access to both the problem description and the appropriate fix.
G. If the MAINTAINER feels it's appropriate to alert the public of the issue, then there's no reason why the ORIGINATOR should not. Traditionally, alerting the community of a problem (but not providing full exploit details) has proven to be futile; other researchers are then just as likely to discover the problem as well--and they may not bide by the guidelines set by this policy. Therefore, if the issue is to be disclosed, all aspects of it should be disclosed. If a third-party discovers and publishes the vulnerability, the MAINTAINER and ORIGINATOR should evaluate the status of a fix, and act accordingly. No matter what, the MAINTAINER should always credit the ORIGINATOR.

\\\ **Difference between version 1 and version 2 of RFPolicy** \\\

Version 1 required a 2 day initial contact period, and then a 5 day wait before disclosure. Due to all the possible ways '2 days' could be mishandled, it was removed in favor of a solid 5 day period.
The email section in version 2 was reworked to discourage emails to InterNIC handles, and encourage trying to locate the correct email address (RTFM :)
Version 2 better defines what should happen at the end of the initial 5 day waiting period.
Version 2 adds the provision for sustained contact from the MAINTAINER.
Version 2 defines possible actions should the ISSUE become public before disclosure by the originator.
"This is not a legal contract" mumbo-jumbo removed from version 2.

\\\ **RFPolicy FAQ** \\\

Q. This policy uses dates and times for gauging responses. How do time zones/holidays/weekends/cultural differences factor in?
A. First off, as noted above, all dates and times are relative to the ORIGINATOR. Now, it is quite possible that a difference in date/time perspective occurs, due to: the ORIGINATOR being on a different continent than the MAINTAINER, the MAINTAINER having a different work week than the ORIGINATOR, the MAINTAINER being sick, the MAINTAINER taking an extended weekend, the MAINTAINER having a holiday, etc. Therefore the initial contact period was extended to 5 days--we feel that 5 days should be adequate to surmount any date/time differences.
Q. I'm a software maintainer, and I can't possibly fix the problem in 5 days....
A. You don't have to. If you (re)read the above, you have 5 days to establish communication. Provided you cooperate with the researcher and keep them 'in the loop', they should provide you with whatever time necessary to resolve the ISSUE (within fair reason).
Q. I'm a software maintainer, and I want more than 5 days!
A. Well, considering that, in general, you don't have *anything* technically, this document hopes to provide you with at least 5. Be on your best behavior, cooperate with the ORIGINATOR, and you should get more. :)
Q. You mention compensation--do ORIGINATORs expect to be paid?
A. NO! (Well, they shouldn't...I can't definitely predict the expectations of people) Compensation, as mentioned in this policy, is meant first-and-foremost to be PROPER CREDIT. Academia has historically and religiously provided credit when referencing all

types of works and research; the ISSUE provided by the ORIGINATOR should also be thought of as research, and the ORIGINATOR should be credited accordingly. Now, beyond that, it may be in the vendor's best interest to promote good relations with the researcher, and one suggested way is to provide updates and product licenses. A lot of research is done on evaluation and trial versions of software--providing a single, full license/copy should produce little impact on the vendor, but greatly help the researcher. Another suggestion is to allow access to support sites/technical content, such as TechNet (if you happen to be Microsoft :)

\\\ **Using this policy** \\\

This policy is free for anyone to modify, republish, sell, or otherwise use. The goal is to establish communication and interaction amongst the security community (users, researchers, and vendors)--not hamper it with copyrights and trademarks.

People are encouraged to use this policy or derivatives. You can make use this policy by supplying the URL (found at the top of this document) in the initial vendor contact email, and giving indication that you intend to following the guidelines stated.

If you intend to be an ORIGINATOR, we suggest you prefix your advisory sent to the MAINTAINER with something similar to:

"This advisory is being provided to you under the policy documented at http://www.wiretrip.net/rfp/policy.html. You are encouraged to read this policy; however, in the interim, you have approximately 5 days to respond to this initial email. This policy encourages open communication, and I look forward to working with you on resolving the problem detailed below."

In addition, should the ORIGINATOR and MAINTAINER arrive at a unified resolution and disclosure, it may be of interest to contact the CVE officials (http://cve.mitre.org) to assign a CVE identifier to the vulnerability. Doing so allows the vulnerability to be referenced and cataloged, facilitating it's acceptance and use into the community.

\\\ **Credits** \\\

Since this is an important part of what this policy attempts to achieve, I should follow the same advice. :)

Version 2 was drafted after extensive input of the community (some 400+ individual suggestions were received). Apologies for not listing all 400+.

Thanks to the following people for initial concepts and input (version 1):

Aleph1 [aleph1-at-securityfocus.com]
Steve Manzuik [steve-at-secureresolutions.org]
Weld Pond [weld-at-atstake.com]
Russ Cooper [russ.cooper-at-rc.on.ca]

Special thanks to Russ Cooper for the large amounts of feedback that helped shape version 1 of this policy.

- rain forest puppy [rfp-at-wiretrip.net]