Introducing constructive vulnerability disclosures

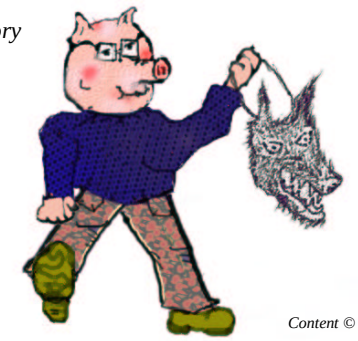
Marko Laakso

Secure Programming Group

Computer Engineering Laboratory

University of Oulu

Finland



Motivation

- Software vulnerabilities prevail:

  "Fragile and insecure software continues to be a major threat to a society increasingly reliant on complex software systems."
  - Anup Ghosh [Risks Digest 21.30]

- A disclosure debate rages:

  "Today, there are appeals to put the genie back into the bottle; that is, to stop the publishing of new vulnerabilities."
  - Rik Farrow [Network Magazine 2000-10-05]

- Our approach:
  - From the grey area between the extremes of *no-disclosure* and *full-disclosure* emerges our (radical) concept of *constructive vulnerability disclosures*.

Introducing constructive vulnerability disclosures

- Presentation contents:
  - Chapter 1: Orientation
    - The Problems - as we see them
    - Our goals
    - Our approach
  - *Chapter 2: A Case Study - PROTOS/c04-wap-wsp-request*
  - *Chapter 3: The big(ger) picture*

Orientation

- Not a vulnerability disclosure per se:
  - No disclosure
    - E.g. vendor quietly fixing things internally
- Classic public vulnerability disclosure models:
  - Partial Disclosure
    - Limited information scope, e.g. no exploit
  - Full Disclosure
    - All available information, down to the dirty details of exploitation
- → Constructive Vulnerability Disclosure
  - A partial disclosure concept
  - Alternative to the more extreme disclosure models

The Problems - as we see them

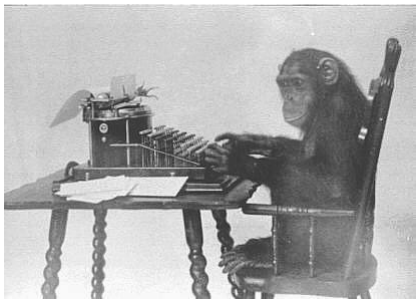
- Poor software quality:
  - The sheer number of information security vulnerabilities
  - Caused mainly by well-known trivial programming errors
  - End-result is a vicious patch-and-penetrate cycle
- Inefficiencies of traditional vulnerability process:
  - Volume of communication, reproduction problems
  - A slight variant of the same exploit may bite multiple vendors
  - Reappearance of same bugs, regression testing
  - Caveat emptor - customers should be able to evaluate
- No-disclosure vs. Full disclosure debate:
  - Effort might be better spent on resolving the actual issues

Our goals

- Through constructive disclosures we aim to support:
  - Low-cost black-box vulnerability analysis
  - Early elimination of some of the most trivial vulnerabilities
  - Vendor awareness beyond one particular vulnerability
  - Regression testing of the future versions
  - Customer-driven product evaluation

Our approach - in a nutshell

Today, thousands of gifted and patient, but uncoordinated monkeys are pounding different products in order to reveal vulnerabilities.



Visual by
http://www.PDImages.com

Think of us as rather dumb monkeys using a monkey-machine and systematic methodology to eliminate the most trivial ones.

Our approach - in practice

- Constructive disclosure **test-suite** development:
  - The area of interest (implementations of a specific protocol)
  - Generation of test-cases through syntax testing
  - Result collection & verification via demonstration exploits
  - Packaging abstracted test-results and test-material
- Test-suite pre-release:
  - Bug-reports and vendor communication
    - 3rd party co-ordination (AusCERT & CERT/CC)
  - ~45-day grace period
- Test-suite release:
  - Vendor initiated advisories and patches
  - Available to all current and future vendors and evaluators

Introducing constructive vulnerability disclosures

- Presentation contents:
  - *Chapter 1: Orientation*
  - Chapter 2: A case study - PROTOS/c04-wap-wsp-request
    - Test-suite results
    - Vulnerability process statistics
    - Vendor feedback
  - *Chapter 3: The big(ger) picture*

A case study - PROTOS/c04-wap-wsp-request

- We applied the proposed model in practice:
  - Wireless Application Protocol (WAP) Suite
    - The mobile counterpart of WWW
  - A test-suite was developed:
    - For the WAP Wireless Session Protocol (WSP) akin to HTTP
    - More specifically for the WSP-requests akin to HTTP-requests
    - WSP-requests are processed by WAP-gateways akin to HTTP proxies and servers
    - Over 4000 test-cases were generated through syntax testing:
      - Test-cases consist of exceptional WSP-requests aimed to reveal robustness problems such as buffer overflows
    - Seven easily available WAP gateway products were tested

[http://www.ee.oulu.fi/research/ouspg/protos/]

Test-suite results

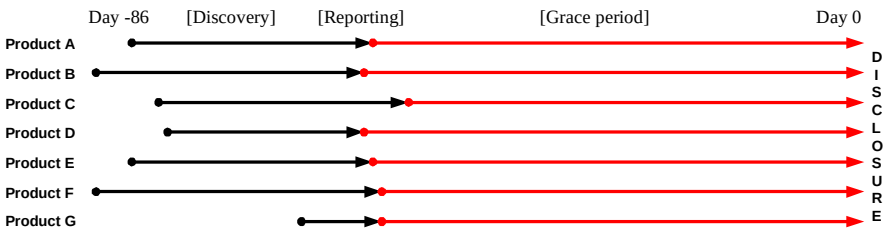
- Results for the 7 WAP gateway products:

Testrun #	Total test-cases	Failed test-cases	Total groups	Failed groups
tr-001	4236	569	39	10
tr-002	4236	141	39	18
tr-003	4236	10	39	2
tr-004	4236	385	39	16
tr-005	4236	664	39	8
tr-006	4236	622	39	14
tr-007	4236	148	39	20

- 7 out of 7 implementations failed
  - Some implementations failed in 50% of anomaly groups
    - 50% chance of monkeys creating a Shakespeare sonnet!
  - Failures (read vulnerabilities) were reported to the vendors

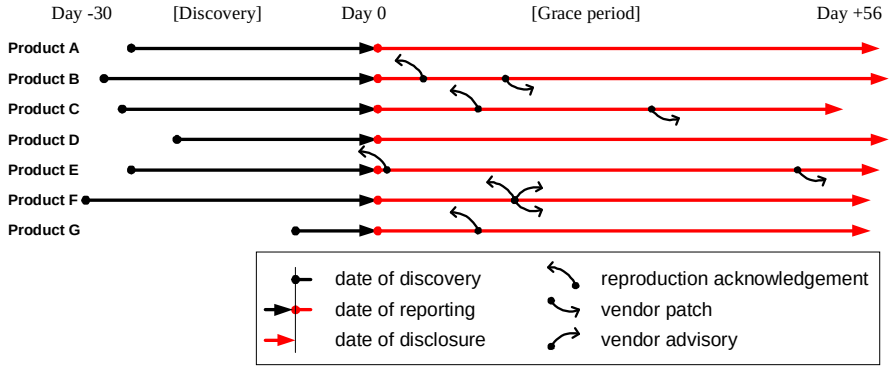
Vulnerability process statistics

- From the vulnerability process point of view:
  - The whole process took 86 days
  - Reports were backed up (in private by):
    - (for 4 products) arbitrary code BoF exploits, (for 3 products) DoS
  - Vendors had a grace-period of at least 51 days
    - Prior to public release of the test-material



Vulnerability process statistics

- A more detailed look:
  - 4 successful product patches (+1 incomplete)
  - 1 vendor initiated advisory + AusCERT security bulletin



Vendor feedback

"Very interesting. We were extremely careful, but there was a deeply embedded support routine that was not doing proper bounds checking on the host portion of the URL."

...

"I was wondering if you are going to post your results anywhere for us to look through? We would be interested to see how we compared to the other products you have been testing."

...

"It is always good to receive reports on the performance of our products, especially when they provide details on how to reproduce problems."

...

"There are a number of problems involved in these tests. The most common one is running out of memory (because of some strange code in an exception path) which (if not handled properly) results in an access to location 0. The most serious problem (from a security point of view) might cause the gateway to transmit some of its memory contents as an HTTP header name to the HTTP server, though you may not have noticed it doing that."

…

All this nice feedback and more from vendors who were rather new to the vulnerability scene. ;)

Introducing constructive vulnerability disclosures

- Presentation contents:
  - *Chapter 1: Orientation*
  - *Chapter 2: A Case Study - PROTOS/c04-wap-wsp-request*
  - Chapter 3: The big(ger) picture
    - Lessons learned - vendor dialog
    - Lessons learned - neutral 3rd party
    - On the necessity of policies and documentation
    - Our view on the future
    - About us

Lessons learned - vendor dialog

- We have observed most of these:
  - No response?
    - keep on trying via alternative channels
  - Response? (Oh my God!)
    - deny everything or downplay
      - *"How To Hide^H^H^Handle Security Problems in Your Products"* [http://www.sockpuppet.org/tqbf/bug-reports.html]
    - sue the messenger (never happened to us)
    - patch quietly
    - patch and release an advisory
    - more thorough review of the implementation
- Dialog is crucial to minimise the risk exposure:
  - If you are a vendor, make sure you have a security contact

Lessons learned - neutral 3rd party

- A neutral co-ordinator is crucial to the process:
  - A trusted party who can assess, arbitrate and advice
  - A link between the reporters and repairers (vendors)
  - Existing organisations (over-worked as they are):
    - FIRST members such as AusCERT and CERT/CC
    - Government movements such as FedCIRC and NIPC
  - Future involvement:
    - Insurance companies?
    - Governments?
  - Are their 'customers' willing to pay for mostly invisible pro-active work rather than for reactive incident response?

On the necessity of policies and documentation

- Vulnerabilities will matter even more in the future
- In need of adaptation and development:
  - Professional codes of ethics
    - Yet to be applied in this context
  - Vendor policies for handling vulnerabilities
    - Since Microsoft has it, others will follow?
  - Policies set out by the reporters and co-ordinators
    - Made at least by Rain Forest Puppy (RFP) and CERT/CC
    - Vendors and the public will know what to expect
  - Maybe even FAQs for the newcomers and main players
  - Media support - buzz on the bugs
    - Respected sources are acting like the yellow press
    - Could we help to avoid "numbness" caused by the media?

Our view on the future

- ... all software is broken!
  - I wouldn't blindly trust a product based on the lack of publicly announced vulnerabilities - it is broken anyway
  - Rather, I would pay attention to the Vendor's track record on handling the inevitable vulnerability disclosures
  - Complexity growth vs. quality improvement?
- Wistfully,
  - Professional handling of vulnerabilities is well supported
  - Customers will have more means to evaluate the quality
  - The standardisation organisations will expand beyond conformance requirements and set a baseline via robustness test-suites

About us

- Our bunch of bananas:
  - This work has been conducted in PROTOS project:
    - a government-funded (3 year) research partnership between the University of Oulu and VTT Electronics
    - supported by three partner companies from the telecommunication industry
  - AusCERT & CERT/CC have been lending a helping hand!
  - Standard disclaimer:
    - Test-suites cover a very limited set of potential vulnerabilities
- Unless we get badly beaten by an outraged mob:
  - We shall develop and release more test-suites during our last project year (2001)

As easy as making the pigs fly?

- Real Life™ processes are complex
- Professional approach required and promoted

Questions and Feedback:

http://www.ee.oulu.fi/research/ouspg
ouspg@ee.oulu.fi