Recurity Labs

Academia vs. Hackers

2nd International Workshop on Secure Information Systems (SIS'07)

October 15-17, 2007, Wisla, Poland

Invent & Verify

Agenda

- The common goal
- The conception of responsibilities
- The approaches
- Examples from the field
- Generalizing
- Proposal for future work

Disclaimer

- We are a practitioners company
- We have (legal) hacker backgrounds
- We work with academics
- We highly respect them

The common goal

- Information security has three generally accepted goals:
  - **Confidentiality**
  - **Integrity**
  - **Availability**
- Academic research and practitioners' approaches share these goals
  - In a more pragmatic way, we can describe information security as the work towards computer systems we can finally trust.

Conception of Responsibilities

- Academic research in the field of information security is conceived as responsible for defense mechanisms
  - Cryptography
  - Safe programming languages
  - Safe computing environments
- By identifying the root cause of an issue, a general solution is devised and shown to work

Conception of Responsibilities

- Hackers are the evil
- Hackers devise new ways of breaking into computer systems and make the world a less secure place
- When hackers solve a problem, it’s a hack
  - Unreliable
  - Not solving the root cause

Conception of Responsibilities

- Programmers are responsible for writing secure code
  - Validation of input data
  - Failing securely
  - Appropriate and secure application of cryptography
- It's what we could call the "C programming approach": The programmer is responsible for allocating enough memory to store the data he is working with.
  - What if that fails?
  - Who is to blame for the consequences?

Conception of Responsibilities

- The programmer's responsibility has been shown to just not work in practice:

  No programmer can ever consider every possible way his code could fail.

The approaches: Academia

- You know a lot more about the academic approaches to computer security
  - We are not going to tell you how your world looks like
- From our point of view, they:
  - Aim at general solutions
  - Work excellent in theory
  - Have been instrumental in practice

Recurity Labs

The approaches: Hackers

- Hacker approaches are:
  - Pragmatic
  - Strictly result oriented
  - Rarely aim at a general solution
- Often, our goals are very vague
  - “Gain more privileges than we are supposed to have”
  - “Somehow prevent X from happening”
  - “Because we can?”

Invent & Verify

The approaches: Hackers

- The more aspects of a problem the hacker considers, the more successful he will be.
  - This holds true for defense and offense alike.
- Example questions:
  - “Can I make the defensive/offensive action too costly for the other side?”
  - “Can I trick the other side into incorrectly operating the defense/offense mechanisms used?”
  - “Can I appear to be on the other side?”
  - “Can I use a layer above/below the one attacked/defended?”

Examples from the field

- Cryptography:
  Hash performance, fuzzy fingerprints, certificates
- Decompilation:
  Boomerang vs. IDA
- Defense mechanisms:
  RISE vs. ASLR
- Imperfect solutions:
  BinDiff

Recurity Labs

Cryptography: Hash Performance

- Cryptography is as important to hackers as it is to everyone else
- Hash algorithms are the tool of choice for integrity protection and fixed length data block representations
- We were tasked to select a hash algorithm:
  - It had to be secure
  - It had to promise security for a few more years
  - It was to be used in a network protocol

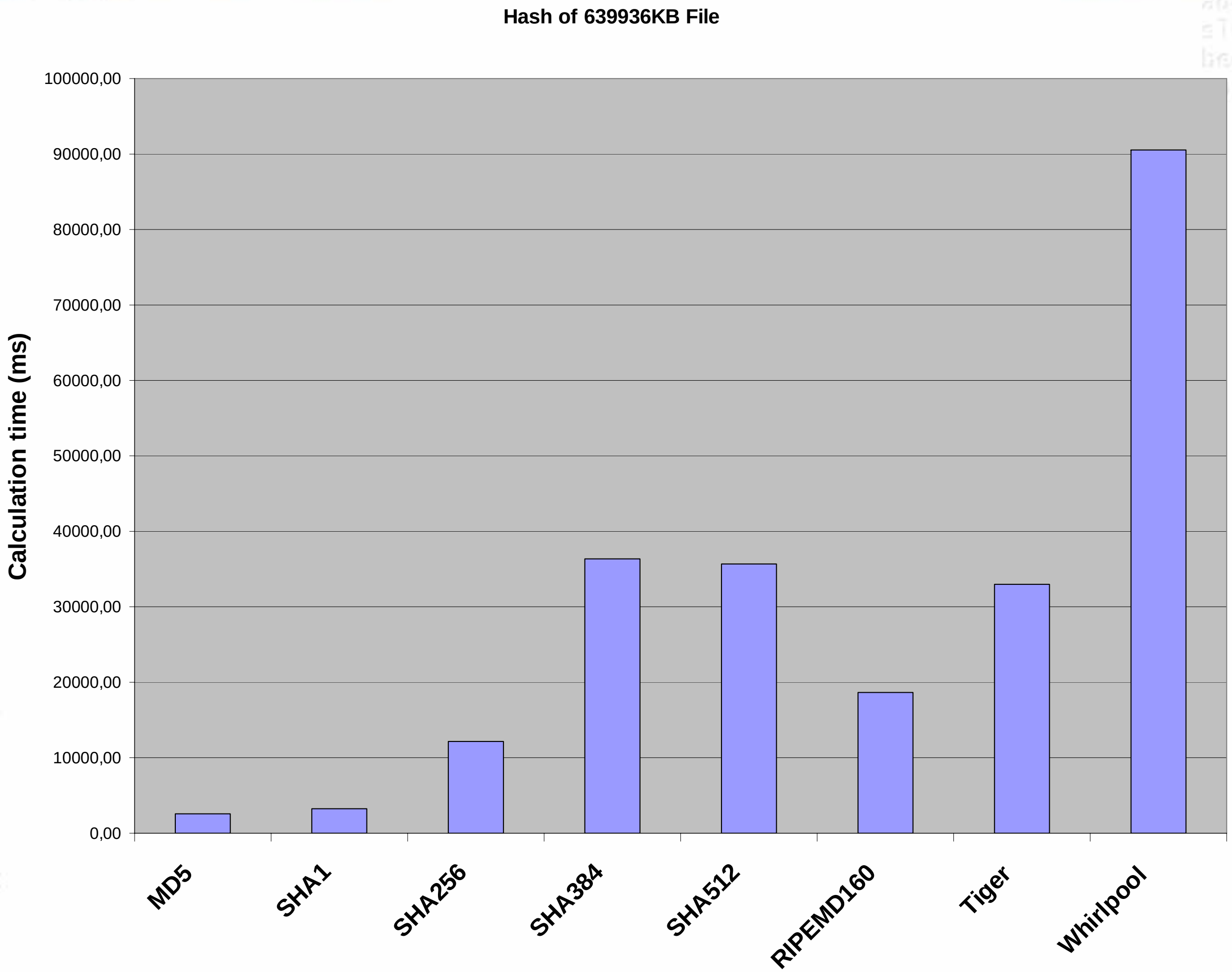
Invent & Verify

Cryptography: Hash Performance

- Selection:
  - MD5 is dead
  - SHA1 appears to die shortly
  - SHA256 – SHA512 are “just” longer versions
- Our favorite cryptographer named **Whirlpool** as the algorithm of choice.
  - So we considered it. But…

Recurity Labs

Cryptography: Hash Performance



Invent & Verify

Cryptography: Hash Performance

- The insecure algorithms were fast
- The secure algorithms were too slow
- Security (collision resilience) is an important design goal when developing hash algorithms
- Speed is an optional design goal

➔In practice, people will use insecure algorithms because they cannot use the secure ones.

  - The collision resilience requirement is overrated while the speed requirement is underrated.

Cryptography: Fuzzy Fingerprints

- In cryptography, fingerprints are the most common way to check valid signatures
  - Unfortunately, some times by humans
  - Example: verify the host identity with SSH
- So, instead of breaking the crypto, a hacker [1] came up with a method of generating look-alike fingerprints.
  - In crypto, they are not the same
  - For the human brain, they may well be

Cryptography: Fuzzy Fingerprints

```
---[Current State]------------------------------------------
0d 00h 02m 00s | Total: 2216k hashs | Speed: 18469 hashs/s
-------------------------------------------------------------
Best Fuzzy Fingerprint from State File /var/tmp/ffp.state
Hash Algorithm: Message Digest 5 (MD5)
Digest Size: 16 Bytes / 128 Bits
Message Digest: d1:bc:df:32:a2:45:2e:e0:96:d6:a1:7c:f5:b8:70:8f
Target Digest:  d6:b7:df:31:aa:55:d2:56:9b:32:71:61:24:08:44:87
Fuzzy Quality: 47.570274%
```

Invent & Verify

Recurity Labs

Cryptography: Fuzzy Fingerprints

- An example of attacking a different layer:
  - Ignore the crypto
  - Fool the crypto user

- Side note:
  A lot of academic work went into this attack for generating the fuzzy fingerprints

Invent & Verify

Recurity Labs

Cryptography: Certificates

- A lot of research went into authentication and authorization using certificates and PKI
  - They all depend on the secrecy of the secret key
  - Therefore, they assume that the secret key is rarely lost to a third party
- Current certificate / PKI systems are unmanageable, since in reality, people are loosing their secret keys to attackers ***all the time***
  - ➔ Spend more time researching fast and scalable revocation ☺

Invent & Verify

Program code recovery

- In reality, there are many legitimate reasons to recover the code of a program from its binary representation:
  - The source code might be lost
  - The compiler might no longer exist
  - The platform might no longer be available

But the most important one is:

  - You simply don't have the source

Boomerang [2]

- Since 2002, the Boomerang decompiler is a successful project to recover source code from binary code
  - Developed by a team at the University of Queensland, Australia
  - Widely recognized for significant advancements in the field of decompilation
  - Team: 6 developers

IDA Pro [3]

- A disassembler, not a decompiler
- The tool of choice for reverse engineering
  - Interactive: accepts the fact that experience is the best pattern recognition
  - Imperfect: Fails often, but allows the user to correct the failure
  - Helps the reverse engineer to recover the source code manually
  - Team: 1 developer

Enter: OOP

- Both tools (should we say approaches)
  - Have been around for a while
  - Have been used and adapted
  - Fail with code generated from C++
- For people in the field, this means:
  - Adjusting IDA took a few days (plugin)
  - Adjusting Boomerang would take ???

➔ While the decompiler attempts to tackle the root cause of the problem (lost source), reality gives the advantage to the quick-and-dirty solution (the disassembler).

Recurity Labs

Exploitation Prevention: RISE [4]

- "Randomized Instruction Set Emulation"
- Encrypts binary code at load time
- Decrypts binary code during execution (instruction fetch) in an emulator environment
  - Since attacker provided code is not encrypted "correctly", it will produce gibberish when decrypted and fail
  - The process will crash, or, in supposedly rare circumstances, loop infinitely
  - Execution of attacker code is prevented

Invent & Verify

Exploitation Prevention: RISE

- Obvious performance impact
- Binary programs can no longer be debugged if anything fails
- And by the way:
  Killing sparrows with nuclear warheads, including AWACS support, was never a wise move.

Exploitation Prevention: ASLR [5]

- Executing attacker provided code needs an important piece of information: the **address**
- Address Space Layout Randomization just loads the program and its segments into different virtual memory regions every time
  - Performance impact: none (regular operation)
  - Result: better than RISE
- Concept developed by hackers who did not want to get compromised by their peers
  - Linux GRSEC/PAX patch
  - OpenBSD
  - Now also in Windows Vista

RISE vs. ASLR

- The inventors of RISE only considered successful exploits in their research
- The inventors of ASLR considered why exploits are often unsuccessful and how to force this situation

Imperfect Solutions: BinDiff

- Comparing two binary programs to determine if they share functionality is hard
  - From an academic point of view, it's impossible
- Comparing two binary programs that are known to be very much alike has advantages for a hacker
  - Think before and after a security patch
- So, someone [6] came up with a structural comparison method that works *most of the time.*

Recurity Labs

Imperfect Solutions: BinDiff

- Turns out:
  The method works well with any two binaries!
  - Even when both are for different CPU platforms.
  - Even when only a few functions are common
- The method also allows to give similarity measures for any two binaries.

➔ Although the problem is still provably unsolvable, for any practical consideration, it is solved.

Invent & Verify

Generalizing

- From our point of view, the scientific way of defining requirements is too strict for real world use.
  - It seems to be common to consider the initial goal of the research a hard requirement while neglecting the environmental aspects
  - What is “given” (the premise) does not always hold true
  - The better approach might be to take several possible combinations of “given” as base for the research

Generalizing

- Assume that things go wrong in the worst possible way
- Never start reasoning with: “Nobody would ever…”
  - Someone will
- Usability in production environments should play a bigger role in the development of defense technologies

Proposal for future work

- The discussion has been too arrogant from both sides in the past
  - Academia is per se right
  - Hackers are per se successful
  - Both is wrong
- At the end of the day, we are all in the same boat here.

Proposal for future work

- Most serious hackers will be honored and delighted to review research concepts (before they are done)
  - Yes, most of them do that for free, even if they are professionals
- Most would also be delighted to show their ideas to academic researchers for feedback and discussions
  - If they don't find their concepts on ACM with someone else's name on it

References

1. http://freeworld.thc.org/papers/ffp.html
2. http://boomerang.sourceforge.net/
3. http://www.datarescue.com/idabase/
4. http://www.cs.unm.edu/~forrest/publications/rise-tissec.pdf
5. http://pax.grsecurity.net/docs/aslr.txt
6. http://www.zynamics.de/products/bindiff.html

Recurity Labs

Jörn Bratzke
Vulnerability Researcher

Phone: +49 30 69539993-0
FAX: +49 30 69539993-8
E-Mail: joern@recurity-labs.com

Recurity Labs GmbH, Berlin, Germany
http://www.recurity-labs.com/

Invent & Verify