Better Security Tool Designs: Brainpower, Massive Threading, and Languages

Golden G. Richard III

Professor and University Research Professor
Department of Computer Science
University of New Orleans

Founder and Owner, Arcane Alloy, LLC

golden@cs.uno.edu
golden@arcanealloy.com
Who?

Professor of Computer Science and University Research Professor, Director, GNOCIA, University of New Orleans
http://www.cs.uno.edu/~golden
Digital forensics, OS internals, reverse engineering, offensive computing, pushing students to the brink of destruction, et al.

Founder, Arcane Alloy, LLC.
http://www.arcanealloy.com
Digital forensics, reverse engineering, malware analysis, security research, tool development, training.

Co-Founder, Partner / Photographer, High ISO Music, LLC.
http://www.highisomusic.com
Rock stars. Heavy Metal. Earplugs.
Why?

With “Big Data”, “floppy thinking” is no longer appropriate.

Computer architectures have moved on. Have we? Or are we still “programming like it’s 1999?” Or 1979?

“Hell with patience, I’m gonna go out and kill something”.

320KB floppies
750GB
1TB

1.44MG floppies
750GB
1TB

4TB
4TB

hourglass

Intel Core™ i7

Intel Core™ i7
Important Trends

• Big hard drives
  – Huge impact on investigations using traditional “storage forensics”
  – Can’t (and don’t want to) wait days for answers
• Big RAM
  – Impact on live forensics and memory analysis
  – Processing 128GB is far different than 512MB
  – Memory analysis tools should give instantaneous results
• Multicore CPUs
  – Very significant—we need to change our ways
• GPUs
  – Significant for some applications—worthy of attention
• Clusters of multicore, GPU-equipped machines
Multicore CPUs

- Modern CPUs are bad a$$—but different
- Gone: ever-increasing clock rates
- Replacement: lots of cores
- Dual-core / Quad-core / 6-core / 12-core / Beyond
- What's next?
- 100's of cores in a single processor
- Programmer doesn't care → software slow

6809: 9,000 transistors
Z80: 8,500 transistors
8086/8: 29,000 transistors
Xeon E5-2600: 4.5 billion transistors
Modern GPUs

2007:
G80 GPU
768MB Device Memory
128 compute cores, ~1GHz each

2009:
G200 GPU
1+GB Device Memory
200+ compute cores, ~1GHz each

2013:
NVIDIA Tesla / Radeon HD 7990
6+GB Device memory
2000+ compute cores

Hardware thread management,
can schedule millions of threads

Thanks, gamers!!
GPU Horsepower

GPU Scalpel (2007)

- K40 @ 5.3TFLOPS
- HD 8990 @ 8.2TFLOPS
- Xeon E5 @ 250GFLOPS

**Table:**
- GT200 = GeForce GTX 280
- G71 = GeForce 7900 GTX
- NV35 = GeForce FX 5950 Ultra
- G92 = GeForce 9800 GTX
- G70 = GeForce 7800 GTX
- NV30 = GeForce FX 5800
- G80 = GeForce 8800 GTX
- NV40 = GeForce 6800 Ultra

**Graph Details:**
- Peak GFLOP/s
- NVIDIA GPU
- Intel CPU
- NV30, NV35, NV40, NV40
- Jan 2003 to Jun 2008
- G80 Ultra
- G80
- G92
- GT200
- 3.2 GHz Harpertown
- 3.0 GHz Core2 Duo
Increasing Tool Performance

• Design matters
  – Think like an OS designer
  – Not knowing what that means…
  – …is part of the problem!
  – Optimization
  – Proper data structures
  – Minimize unnecessary copies / locking
  – Asynchronicity (it’s not a Police album)
Increasing Tool Performance (2)

• Use all available computational resources
  – Where appropriate, massively threaded applications
  – Specifically program to multicore CPUs
  – GPUs where appropriate

• Asynchronous I/O
  – Hide limited disk bandwidth with aggressive prefetching and caching
  – Overlap disk I/O and computation within the application
Challenges: Experience

• Most programmers not familiar with massively threaded software designs or distributed computing

• Potentially complicated synchronization issues

• GPU programming harder
  – Generally requires application to be broken into distinct host / GPU components
  – GPU component is SIMT (SIMD)
  – Complicated memory hierarchy
  – Components must bulk copy data between host and GPU
  – [Though now “zero copy” transfer between host and GPU]
  – Portability issues
Challenges: Languages

- Poor language support for massively threaded designs

- Python? 😞 ➜ dreaded GIL
- Ruby? 😞 ➜ C impl: uses non-thread-safe libraries
- Java? 😞 ➜ grrrr..
- C / C++? 😊 ➜ does this make you happy?
- Go? 😞 - 😊 ➜ upcoming talk by Vico
- Erlang? (!) ➜ maybe
loop(Users, N) ->
    receive
    {connect, Pid, User, Password} ->
        io:format("connection request from:~p ~p ~p~n", [Pid, User, Password]),
        case member({User, Password}, Users) of
            true ->
                Max = max_connections(),
                if
                    N > Max ->
                        Pid ! {ftp_server,
                            {error, too_many_connections}},
                        loop(Users, N);
                    true ->
                        New = spawn_link(?MODULE, handler, [Pid]),
                        Pid ! {ftp_server, {ok, New}},
                        loop(Users, N + 1)
                end;
            false ->
                Pid ! {ftp_server, {error, rejected}},
                loop(Users, N)
        end;
    {'EXIT', Pid} ->
        io:format("Handler ~p died~n", [Pid]),
        loop(Users, lists:max(N-1, 0));
    Any ->
        io:format("received:~p~n", [Any]),
        loop(Users, N)
    end.
## Headaches Do Pay Off: Massively Threaded Scalpel

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>RUN TIME</th>
<th>BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalpel v1.60</td>
<td>448s</td>
<td>45MB/s</td>
</tr>
<tr>
<td>Scalpel v1.91MT-multicore</td>
<td>178s</td>
<td>111MB/s</td>
</tr>
<tr>
<td>Scalpel v1.91MT-multicore-asyncc</td>
<td>146s</td>
<td>140MB/s</td>
</tr>
<tr>
<td>Scalpel v1.91MT-gpu-asyncc</td>
<td>77s</td>
<td>265MB/s</td>
</tr>
</tbody>
</table>

Custom binary string search + async I/O + massively threaded design with GPU / multicore overlap

(Doesn’t make use of zero-transfer memory access in modern NVIDIA GPUs)

20GB disk image, 25 file types targeted for carving. Quad-core Dell XPS 720 w/ 4GB RAM, 8 x 15K SCSI disk array (max bandwidth ~600MB/sec), G200 GPU w/ 896MB device RAM, 192 compute cores.
Thanks, And:

An Introduction to Digital Forensics: Privacy, Practice, and Research
Friday (TOMORROW) 12/6 @ 3pm
Tulane University
Stanley-Thomas 302
Free, obviously

tutorial M3: Introduction to Reverse Engineering Malware
ACSAC 2013
Monday 12/9 (All Day)
http://www.acsac.org/2013/
Not free, but it’ll be fun

golden@cs.uno.edu
golden@arcanealloy.com