Attacker “Math” 101

Professor Dai Zovi
Institute for the Advancement of Memory Corruption
What is this all about?

- Thinking like an attacker
- Modeling their choices
- Predicting future behavior
- Making better defense decisions
Attacker “Math”

- If the cost to attack is less than the value of your information to the attacker, you will be attacked.
- Mass malware must be financially profitable for the profit-driven attackers.
- APT campaigns must scale according to the resources at the attacker's disposal.
Attack Graphs

- Informal tool to visualize and analyze how to attack a system (software, network, etc)
- Nodes represent levels of access/positions or actions to perform
- Nodes can be weighted with a cost, calculated in terms of capital, skill, risk, opportunity, or time/effort required
- Actors can be modeled in similar terms
Adversary Modeling

- Different groups/types of attackers have different intents, capabilities, strategies, and tactics.

- Most organizations are not concerned with all of them.

- Mass malware

- APT

- ZFO / Anonymous / LulzSec

- Stuxnet
Conjecture

- Attackers will take the least cost path through an attack graph from their start node to their goal node, where:
  - Cost is a multi-variable equation
  - Start nodes represent some level of access or position
  - Goal nodes represent a consequence that is good for attacker, bad for defender
Mass Malware

Internet Access  ???  Profit
Internet Access

Mass compromise and infect
Malicious Ads
SEO

Malicious HTML/JS

Drive-by Download
Social Engineering

Installations

Banking Creds
Stolen CC#
Stolen PII

Profit
Malicious HTML/JS Execution

- Chrome 10
  - WebKit Vulnerability
  - ASLR Bypass
  - DEP Bypass
  - Sandboxed Low Integrity Native Code Execution
- IE 8/9
  - IE Vulnerability
  - ASLR Bypass
  - DEP Bypass
  - Low Integrity Native Code Execution
- FF 4
  - Java Vulnerability
  - ASLR Bypass
  - DEP Bypass
  - Medium Integrity Native Code Execution
- Firefox Vulnerability
  - ASLR Bypass
  - DEP Bypass
Sandboxed Low Integrity Native Code Execution

Sandbox escape

Medium Integrity Native Code Execution

Local Privilege Escalation

Admin User RCE

M-H Integrity Escalation

High Integrity Privileged RCE

Install Rootkit

Privileged Host Persistence

Low Integrity Native Code Execution

Integrity Escalation
Attacker Math

\[
\text{Cost(Medium Integrity RCE)} = \text{Min}( \\
.10 \times (\text{WebKit vuln} + \text{ASLR/DEP} + \text{Sandbox}), \\
.60 \times (\text{IE vuln} + \text{ASLR/DEP} + \text{IE PM}), \\
.20 \times (\text{FF vuln} + \text{ASLR/DEP}), \\
.95 \times (\text{Flash vuln} + \text{ASLR/DEP} + \text{IE PM}), \\
.75 \times (\text{Java vuln}) \\
) 
\]
Exploits are Hard

- Mass malware wants to go from injected content to installations at the least cost.
- If drive-by downloads become unprofitable, they will increasingly shift to social engineering (self-signed applets, rogue AV, etc).
- If no one published exploits, they would just repurpose exploits from targeted attacks (they are already doing this).
Lessons

- Exploiting Java is the cheapest path to Medium Integrity Native Code Execution or User-privileged Remote Command Execution

- Therefore, attackers will prefer exploiting Java over browser vulnerabilities

- Exploiting the kernel is the cheapest path from Unprivileged Native Code Execution to Privileged Code/Command Execution

- Therefore, attackers will deploy kernel exploits before sandbox evasions (and already have)
JailbreakMe 2.0
iOS 4.0 Runtime Security Features

- Mandatory Code Signing
  - All executables must be signed by Apple or a provisioned code signing certificate

- Code Signing Enforcement
  - All executable memory pages must have a valid signature

- Runtime sandbox
  - The actions that the app may perform are restricted by the kernel at runtime
MobileSafari HTML → Memory corruption vulnerability → Return-oriented execution → Code signing enforcement bypass → Unprivileged native code execution

- Sandbox evasion
- Mandatory code signing evasion
- Privilege Escalation
- Privileged native code execution
MobileSafari HTML → Memory corruption vulnerability → Return-oriented execution → Code signing enforcement bypass → Unprivileged native code execution

- Sandbox evasion
- Mandatory code signing evasion
- Privilege Escalation
- Privileged native code execution

Kernel exploit
Response

- Apple released iOS 4.0.1 to patch vulnerabilities within 2 weeks
  - Speed of response discourages similar 0day jailbreaks
  - JB community shifted focus back to boot ROM exploits
- Press and users largely celebrated the release of the jailbreak
  - What would the response have been if the same techniques were branded as an exploit (bad) rather than jailbreak (good) ?
- Jailbreak was quickly adapted into a PoC rootkit by Eric Monti
Lessons

- Jailbreak developers’ use of exploits mimics malicious attackers
  - They are resource constrained, just like defenders
  - Desire maximum return on investment for their exploits
  - Deploy exploits strategically
    - Preservation of “SHAtter” in favor of “Limera1n” exploit
  - Choose target attack surfaces for maximum return
    - Boot ROM (unpatchable) vs. iOS (quickly patchable)
Conjecture

- The level of security offered by a path through an attack graph is measured by the cost required for an attacker to traverse it.

  - Measuring the precise cost of a path requires spending exactly that amount to traverse it.

  - However, we can estimate or bound costs of some subpaths by proxy or observation.
Observable Cost Measurements

- Fuzzing statistics
  - Fuzzing stats measure cost to find a crash in a particular product

- Bug bounties
  - Anonymous ZDI submissions measure cost to find a vulnerability in that product
  - Pwn20wn measures cost to develop an exploit against that product
Lies, Damn Lies, and Fuzzing Statistics
Charlie Miller’s Fuzzing Stats

- "Dumb fuzzing"
- 12-25% of unique crashes deemed "exploitable"
- 33-50% of unique crashes deemed "exploitable" or "probably exploitable"

Meditate On These Numbers

- 300 file format parsers
- 1,000,000 fuzz iterations
- 1600 unique bugs
- 200-800 likely exploitable vulnerabilities

Bugs as natural resources

- Don't just count quantity of bugs, measure the drilling depth required to extract them.
- Can they be refined (exploited) using current technology and processes?
- Estimate size of discovered fields.
Theorem

Cost to discover a vulnerability in a particular product is less than the sum of a claimed bug bounty for that type of vulnerability plus the value of credit to that particular researcher

\[ \text{Cost(Vuln)} \leq \text{Value(Bounty)} + \text{Value(Credit)} \]
Your credit is no good here

- The value of credit to different researchers is variable, so let's remove it from the equation:
  \[ \text{Cost(Vuln)} \leq \text{Value(AnonBounty)} \]
ZDI/iDefense
Anonymous Submissions

- No exploit is required, just a verifiable security vulnerability in a product significant enough that ZDI would care to pay for the bug

- ZDI bounties paid are confidential, but we will assume that they are less than Pwn2Own

- So we don't actually get Cost(Vuln), just the products for which:
  Cost(Vuln) <= Value(Anon Bounty)
<table>
<thead>
<tr>
<th>ZDI-CAN-855</th>
<th>Microsoft</th>
<th>CVSS: 10</th>
<th>2010-06-30 (121 days ago)</th>
<th>2011-02-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovered by: Anonymous</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZDI-CAN-852</th>
<th>Mozilla Firefox</th>
<th>CVSS: 10</th>
<th>2010-06-30 (121 days ago)</th>
<th>2011-02-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovered by: washi of team509</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZDI-CAN-843</th>
<th>RealNetworks</th>
<th>CVSS: 10</th>
<th>2010-06-30 (121 days ago)</th>
<th>2011-02-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovered by: Anonymous</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZDI-CAN-842</th>
<th>Adobe</th>
<th>CVSS: 10</th>
<th>2010-06-30 (121 days ago)</th>
<th>2011-02-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovered by: Abdullah Ada</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZDI-CAN-829</th>
<th>Microsoft</th>
<th>CVSS: 10</th>
<th>2010-06-30 (121 days ago)</th>
<th>2011-02-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovered by: Anonymous</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZDI-CAN-828</th>
<th>Apple</th>
<th>CVSS: 10</th>
<th>2010-06-30 (121 days ago)</th>
<th>2011-02-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovered by: Anonymous</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZDI-CAN-798</th>
<th>Microsoft</th>
<th>CVSS: 10</th>
<th>2010-06-30 (121 days ago)</th>
<th>2011-02-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovered by: Anonymous</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Anonymous ZDI Advisories in 2011

- HP Client Automation/Radia (ZDI-11-105), Cisco Secure Desktop (ZDI-11-092, ZDI-11-091)
- Microsoft PowerPoint (ZDI-11-125, ZDI-11-124, ZDI-11-123), Excel (ZDI-11-043, ZDI-11-042, ZDI-11-041, ZDI-11-040)
- WebKit (ZDI-11-104, ZDI-11-101, ZDI-11-097)
- Adobe Flash (ZDI-11-081), Shockwave (ZDI-11-079), Reader (ZDI-11-075)
Corollary

- The cost to discover and reliably exploit a vulnerability in a particular product is less than the sum of a claimed Pwn2Own prize for that product, the value of the laptop, and the value of fame to that researcher.

  \[ \text{Cost(Exploit)} \leq 15\,\text{k} + \text{Value(Laptop)} + \text{Value(Fame)} \]

- It is harder to remove fame from the equation here, but what can we measure?

  - Time-to-exploit
Malicious HTML/JS Execution

- Chrome 9
  - WebKit Vulnerability
- IE 8
  - IE Vulnerability
- FF 3
  - Firefox Vulnerability
- Safari 5
  - WebKit Vulnerability

ASLR Bypass

WebKit Vulnerability

IE Vulnerability

Firefox Vulnerability

WebKit Vulnerability

64-bit NX Bypass

ASLR Bypass

DEP Bypass

Sandboxed Low Integrity Native Code Execution

Low Integrity Native Code Execution

Medium Integrity Native Code Execution

Native Code Execution

File write access

Chrome Sandbox Escape

Windows Kernel Exploit

IE PM Escape
After building the tools from scratch, it took him about two weeks to find the bug and set out to exploit it. The result was an attack that reliably commandeers a Mac when Safari visits a website that hosts the malicious code.
"I spent about six weeks finding the vulnerabilities and engineering the exploits," Fewer said. "Then I decided to give [Pwn2Own] a go, and bought a plane ticket."
IE9 released today: we have at least 5 unpatched use-after-free vulnerabilities in IE9. Details shortly available via VUPEN TPP

Google has fixed -in less than 24hrs- a WebKit vuln. affecting Chrome and exploited at Pwn2own to pwn the BlackBerry http://bit.ly/eeVwai

Reportedly targeted attacks taking advantage of the recently disclosed & unpatched Internet Explorer MHTML vuln are ongoing

Nobody has challenged Chrome at this year's pwn2own. If next year Google offers 40k to pwn Chrome, maybe we will do the magic

We released 3 of our Safari exploits (ASLR/DEP bypass) fixed in Safari v5.0.4, not the pwn2own code (still unpatched) http://bit.ly/cCdpcN

MS has confirmed that IE9 RC is not affected by the pwn2own exploit and IE9 will be out on Monday. Good but we already have IE9 ODS
Lessons

- Requiring evasion of mitigations or exploitation of additional vulnerabilities in the chain increases time to develop a full exploit linearly.

- And therefore, it also increases the cost to develop such an exploit linearly.
Hypothetical: Browser Exploit Powerball

- Once a year, public bounty is posted for “reliable enough” exploits against dominant desktop configurations
  - Must gain enough privs to accomplish attacker objectives
- Prices gradually increase until first winning submission is received for a particular target
  - Incentivizes submitting as early as possible
- Contestants MUST sign NDA on disclosing participation and submission, vulnerability is reported to vendor anonymously
  - Forcibly removes credit and fame from the equation
Armchair APT Analysis
Conjecture

- “APT” attacks must scale according to resources at the attacker’s disposal
- “Aurora” campaign wasn’t just against Google, or only 34 targets, but apparently against thousands of organizations (Reuters)
Cloppert’s APT Kill Chain Model

- Recon
- Vulnerability weaponization
- Exploit delivery
- Host exploitation
- Host persistence
- Command and control
- Actions on Objectives
<table>
<thead>
<tr>
<th>Phase</th>
<th>Does it scale?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recon</td>
<td>NOT SO MUCH</td>
</tr>
<tr>
<td>Weaponization</td>
<td>HELL YES</td>
</tr>
<tr>
<td>Exploit delivery</td>
<td>HELL YES</td>
</tr>
<tr>
<td>Host exploitation</td>
<td>HELL YES</td>
</tr>
<tr>
<td>Host persistence</td>
<td>HELL YES</td>
</tr>
<tr>
<td>Command and control</td>
<td>HELL YES</td>
</tr>
<tr>
<td>Actions on Objectives</td>
<td>NOT AT ALL</td>
</tr>
<tr>
<td>ID</td>
<td>Disc Date</td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
</tr>
<tr>
<td>71254</td>
<td>2011-03-14</td>
</tr>
<tr>
<td>68987</td>
<td>2010-11-03</td>
</tr>
<tr>
<td>69160</td>
<td>2010-11-03</td>
</tr>
<tr>
<td>68932</td>
<td>2010-10-28</td>
</tr>
<tr>
<td>68905</td>
<td>2010-10-25</td>
</tr>
<tr>
<td>68921</td>
<td>2010-10-25</td>
</tr>
<tr>
<td>68803</td>
<td>2010-10-21</td>
</tr>
<tr>
<td>67988</td>
<td>2010-09-14</td>
</tr>
<tr>
<td>68024</td>
<td>2010-09-13</td>
</tr>
<tr>
<td>67849</td>
<td>2010-09-07</td>
</tr>
<tr>
<td>66387</td>
<td>2010-07-16</td>
</tr>
<tr>
<td>65141</td>
<td>2010-06-04</td>
</tr>
<tr>
<td>62810</td>
<td>2010-03-09</td>
</tr>
<tr>
<td>61697</td>
<td>2010-01-14</td>
</tr>
</tbody>
</table>
Lessons

- Focusing defensive countermeasures on the cheapest (for the attacker) phases of the attack is not as effective as focusing on the expensive.
- If your defense is cheaper than their offense, you will gain the advantage.
Conclusion

- Think like an attacker to predict what they will do and how they will attack you
  - Model your understanding of their intent, capabilities, and constraints
- Adjust your threat model based on new information on attackers and their capabilities
  - i.e. Anonymous pre- and post-Gawker
Questions?

@dinodaizovi / ddz@theta44.org

http://blog.trailofbits.com