OpenConflict

Preventing Map Hack in online games

Elie Bursztein, Mike Hamburg, Jocelyn Lagarenne, Dan Boneh
Stanford University
Welcome to the real world

#kartograph / @elie
273
Millions games sold in
2009
Strategy account for 35% of the games sold in 2009
Game Tournaments
Cash prize up to 200,000$
Outline

- Background
Outline

• Background

• Memory based map-hack with Kartograph
Outline

• Background
• Memory based map-hack with Kartograph
• Defending against map-hack
Outline

• Background
• Memory based map-hack with Kartograph
• Defending against map-hack
• Starcraft 2 case study
Outline

• Background
• Memory based map-hack with Kartograph
• Defending against map-hack
• Starcraft 2 case study
• Open conflict benchmark
Background
supernatural powers!

- Learn kungfu
- Infinite money
- Xray vision
- god mode
Memory based attack

Memory
Memory based attack

Memory

Modification
Memory based attack

Memory Modification
Benefits (fast and furious)

- Generic
- Fast
- Invisible
Drawbacks: Needle in a Haystack

- Structure are hard to find
- No control over the flow

Drawbacks: Needle in a Haystack

- Game memory
- Structures
Building

Tankbuster
$ 300  0:05
Anti-Armor
Steely-cold warriors whose personal plasma-cutter cannons can slice through enemy armor.
Units

Tankbuster
$300  0:05

Anti-Armor
Steely-cold warriors whose personal plasma-cutter cannons can slice through enemy armor.
Minimap

Tankbuster
$300  0:05

Anti-Armor
Steely-cold warriors whose personal plasma-cutter cannons can slice through enemy armor.
Tankbuster

$300  0:05

Anti-Armor

Steely-cold warriors whose personal plasma-cutter cannons can slice through enemy armor.
RTS example Starcraft 2
How to cheat at a RTS?
How to cheat at a RTS?

Resources
How to cheat at a RTS?

Resources

units
How to cheat at a RTS?

Resources

units

map
What is a map hack?

Fog of war
What is a map hack?
There is no spoon
Find the information hidden by the game

Understand the data structures

Abuse this knowledge
What is Kartograph?

- Memory analysis techniques
- Visualization techniques
How Kartograph works?
How Kartograph works?
How Kartograph works?

Reduce haystack
How Kartograph works?

Reduce haystack

Find
How Kartograph works?

Reduce

haystack

Find

Understand
How Kartograph works?

Reduce haystack  Find  Understand  Rewrite
Acquiring game memory
Acquiring game memory
Reducing memory

Game memory
Reducing memory

Step 1: play
Reducing memory

Game memory

Step 1 play
Reducing memory

Game memory

Step 1 play

Step 2 discover
Reducing memory

Game memory

Step 1 play

Step 2 discover
Reducing memory

Step 1 play

Step 2 discover

Step 3 play

Game memory
Reducing memory

Game memory

Step 1 play

Step 2 discover

Step 3 play
Reducing memory
Acquiring the game’s memory
Step 1
Removing unrelated memory
Step 2
Discovering the map and keeping relevant memory
Step 3
Removing more unrelated memory
Memory reduction algorithm efficiency

Potential Memory in Megabytes

Launch | Play | Discover | Play More

- Starcraft II
- C&C Red Alert 3
- Civilization IV
- C&C TS
- Age of Empires III
- Anno 1701
- C&C Red Alert 2
- Supreme Commander 2
- Warcraft III
Step 4
Finding the map in the remaining memory
Working assumption
Working assumption
Step 5
Isolating the potential map
In game

In memory
Step 6
Understanding the map’s structure
Step 8
Rewriting game memory for fun and profit
Starcraft 2 mini map
Starcraft 2 mini map
Unexpected effects
Defense requirements
The passive eavesdropping adversary

- Complete control of his machine
- Can understand the game memory structure
- Can identify and parse any data structure
We say that a passive attacker **defeats** the game if the attacker **can write a program** P that **reveals information** about the opponent **beyond what is revealed by the game's rules**. Otherwise we say that the game is **secure against a passive adversary**.
Making games secure

- Use two-party (multi-party) cryptography protocol to ensure that the memory contains only the data the user need
Set intersection protocol

Alice

Bob

\( V_a \cap V_b \)
Set intersection protocol

\[ \bigcap \]

Alice

\[ V_a \]

Bob

\[ V_b \cap V_a \]

\[ V_b \cap V_a \]

Elie Bursztein, Jocelyn Lagarenne, Mike Hamburg, Dan Boneh

OpenConflict
Set intersection protocol

Alice

\( V_a \)

Bob

\( V_a \)
Set intersection protocol

Alice

Bob

Va ∩ Ub
Set intersection protocol

Alice

Va

Va ∩ Ub

Bob

Va

Va ∩ Ub
Constraints

• Bob learn nothing about $V_a$

• Alice learn nothing about $U_b$ other than $V_a \cap U_b$
Constraints

• Bob learn nothing about Va

• Alice learn nothing about Ub other than $Va \cap Ub$

Computing with these constraints is called the oblivious intersection set problem
Chosen oblivious intersection protocol

- Due to Jarecki and Liu
- Use an oblivious function evaluation as sub-protocol
- Adapted and optimized for our problem
For one unit

unit key:

$$o_k(v) := H_1(v)^k \in \mathbb{G}$$
unit key:

\[ o_k(v) := H_1(v)^k \in \mathbb{G} \]
We would like a protocol whereby "lice learns the value of the flows. Formally, the security requirement is that information about Bob's units in that cell. We encapsulate many protocols for this problem have been proposed. Any player within her visibility region gives the set $B^4$. For each pair of opposing players, let's call them "lice on view of the protocol wand therefore Bob learns nothing. "lice should learn nothing about Bob's units, but not others. We want Bob to send "lice his data, the union of all of "lice's visibility regions gives the set $A^4$, and this will be accomplished by running the protocol in $V$, the union of all of "lice's visibility regions gives the set $A^4 \cap B^4$. Let $M^4$ denote the game needs to show "lice all the units belonging to $A^4 \cap B^4$ as a function $o_k(v) := H_1(v)^k \in \mathbb{G}$.

Proof: $H_1(v)^r \rightarrow H_1(v)^{rk}$

Likewise, we want Bob to learn $A^4 \cap B^4$ as required. Likewise, we want Bob to learn $A^4 \cap B^4$ as required. Given $A^4$, $B^4$, and $V^4$, we will assume a passive threat model $w_1a_1a_1 honest but corrupted players. Constraints stated informally as "lice's exponentiations are relative to a fixed base $s$.

Suppose "lice can find some oblivious function evaluation protocol when $(v, w) \rightarrow (o_k, o_{k'})$. Now, the oblivious function evaluation protocol runs as $H_1(v)^r \rightarrow H_1(v)^{rk}$.

Simple variants of this can be more efficient for certain cases. In particular, a subgroup of the points on an elliptic curve and a hash function $H$ are defined as $H(v) := g^v \in \mathbb{G}$.

We found that a good starting point for our settings is an oblivious function evaluation protocol $o_k(v) := H_1(v)^k \in \mathbb{G}$.

Hence, Bob learns nothing, and there is a simulator that simulates Bob's machine can correctly render the output of the protocol.

Subject to the two constraints above is called oblivious intersection protocol due to Jarecki and Liu. We describe the oblivious function evaluation protocol when $(v, w) \rightarrow (o_k, o_{k'})$. Proof: $H_1(v)^r \rightarrow H_1(v)^{rk}$.

For our function $o_k(v) := H_1(v)^k \in \mathbb{G}$, for any $o_k(v) := H_1(v)^k \in \mathbb{G}$, $H_1(v)^r \rightarrow H_1(v)^{rk}$.

Given $A^4 \cap B^4$, we will refer to some domain. Bob holds the secret key $k$.

Therefore, $\exists 1$.

Hence, Bob learns nothing.

For our function $o_k(v) := H_1(v)^k \in \mathbb{G}$, for any $o_k(v) := H_1(v)^k \in \mathbb{G}$, $H_1(v)^r \rightarrow H_1(v)^{rk}$.

Given $A^4 \cap B^4$, we will refer to some domain. Bob holds the secret key $k$.

Therefore, $\exists 1$.

Hence, Bob learns nothing.

Now, the oblivious function evaluation protocol runs as $H_1(v)^r \rightarrow H_1(v)^{rk}$.

Simple variants of this can be more efficient for certain cases. In particular, a subgroup of the points on an elliptic curve and a hash function $H$ are defined as $H(v) := g^v \in \mathbb{G}$.

We found that a good starting point for our settings is an oblivious function evaluation protocol $o_k(v) := H_1(v)^k \in \mathbb{G}$.

Hence, Bob learns nothing, and there is a simulator that simulates Bob's machine can correctly render the output of the protocol.
We would like a protocol whereby 'lice learns the value of 'lice's machine needs to determine curious meaning that 'lice and Bob will execute the protocol For each cell in the intersectionfi 'lice wants to learn many protocols for this problem have been proposed and this will be accomplished by running the protocol in contain unitsfi buildings and other objects" we will refer her own unitsxfi but not others1 We want Bob to send 'lice and Bobxfi 'lice can see some of Bob's units wthose close to Bob's unitsx1 

\[ o_k(v) := H_1(v)^k \in \mathbb{G} \]

\[ H_1(v)^r \]

\[ H_1(v)^k = H_1(v)^{rk}r^{-1} \]

\[ H_1(v)^{rk} \]
Multi-units

With a straightforward implementation, Bob would need to test each cell in order to minimize computation tiles. Then Alice can decompose of this property, we can construct multiple levels of grid cells shapes rather than disconnected point sets. To take advantage E. Hypergrids queries, so the chaff increases his workload. However, Bob must respond to all tell the difference between these random queries and Alice’s sending meaningless, random queries. Bob won’t be able to know an upper bound of the number of Bob’s units. Since random chunks and units that she can’t see so she only infers by adding not possible to completely hide the total number of Bob’s about the total number of units that Bob has. While it is lengths of

D. Chaff Visibility regions are almost always large, continuous Conversely, Alice can hide the size of her visibility map by encrypting each chunk separately. The chunks from all sized chunks possibly padding the last chunk and Our system ensures this by breaking encryption of Note that we must ensure that the "fi

\( U_B \times V_A \rightarrow U_B \)

encrypted chunks from \( U_B \)

\( V_A \rightarrow v_1^{r_1}, \ldots, v_n^{r_n} \rightarrow U_B \)

Note that Bob learns nothing about protocol with Bob to obtain Alice then uses the oblivious function evaluation

\[ v_1^{kr_1}, \ldots, v_n^{kr_n} \]

multi-units

\[ V_A \rightarrow V_A \]

Figure 1

Starcraft visible cells vs. hypergrid cells, second buckets

http://ly.tl/p19

Elie Bursztein, Jocelyn Lagarenne, Mike Hamburg, Dan Boneh

OpenConflict
Chaff

• The basic protocol leaks information
  • The number of Alice visible cell
  • The number of bob unit (nb encrypted chunk)
• Both are resolved by adding a chaff
  • Bob and Alice add random values that “pad” the data
Design a protocol that is fast enough so the added game latency is imperceptible to users.
Establishing a baseline
Methodology

• 1000 Starcraft 2 replays from pro-gamer
  • Game duration
  • Number of units by players
  • Number of visible cells
  • Number of actions by seconds
• Map playable size:
  • min 15180 cells
  • max 24640 cells
Analysis difficulty

1. Replays use a proprietary file format (MQP)
   • Wrote a custom parser

2. Replay only record players actions
   • Wrote a minimal game engine
   • return an upper-approximation
Game duration

(duration (sec) vs. number of games)

- Number of games decreases as duration increases.
- The graph shows that most games are completed within 500 seconds, with fewer games taking longer durations.

Data: Elie Bursztein, Jocelyn Lagarenne, Mike Hamburg, Dan Boneh

OpenConflict

http://ly.tl/p19
Actions per second

- Game related actions
- Camera related actions
- Total actions

Actions per second vs. duration (sec)
Map cell visible by duration

![Graph](image-url)
• Hypergrid: 4 grid instead of 1
  • 1x1 cells
  • 2x2 cells
  • 3x3 cells
  • 4x4 cells
• Lest visible cells
• 4 times more units
• Hypergrid: 4 grid instead of 1
  • 1x1 cells
  • 2x2 cells
  • 3x3 cells
  • 4x4 cells
• Lest visible cells
• 4 times more units
• Hypergrid: 4 grid instead of 1
  • 1x1 cells
  • 2x2 cells
  • 3x3 cells
  • 4x4 cells
• Lest visible cells
• 4 times more units
• Hypergrid: 4 grid instead of 1
  • 1x1 cells
  • 2x2 cells
  • 3x3 cells
  • 4x4 cells
• Lest visible cells
• 4 times more units
- Hypergrid: 4 grid instead of 1
  - 1x1 cells
  - 2x2 cells
  - 3x3 cells
  - 4x4 cells
- Lest visible cells
- 4 times more units
• Hypergrid: 4 grid instead of 1
  • 1x1 cells
  • 2x2 cells
  • 3x3 cells
  • 4x4 cells
• Lest visible cells
• 4 times more units
• Hypergrid: 4 grid instead of 1
  • 1x1 cells
  • 2x2 cells
  • 3x3 cells
  • 4x4 cells
• Lest visible cells
• 4 times more units
Impact of the hypertiled grid

Number of visible cells vs duration (sec)

- Standard grid
- Hypertiled grid
## Open-Conflict Benchmark

<table>
<thead>
<tr>
<th>$v$</th>
<th>$u$</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>5ms</td>
<td>6ms</td>
<td>8ms</td>
<td>9ms</td>
<td>11ms</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>8ms</td>
<td>9ms</td>
<td>11ms</td>
<td>12ms</td>
<td>14ms</td>
</tr>
<tr>
<td>300</td>
<td>300</td>
<td>11ms</td>
<td>13ms</td>
<td>14ms</td>
<td>16ms</td>
<td>17ms</td>
</tr>
<tr>
<td>400</td>
<td>400</td>
<td>14ms</td>
<td>16ms</td>
<td>17ms</td>
<td>19ms</td>
<td>20ms</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
<td>17ms</td>
<td>19ms</td>
<td>20ms</td>
<td>22ms</td>
<td><strong>24ms</strong></td>
</tr>
</tbody>
</table>
Conclusion

• Developed a generic method to perform memory based attack

• Established a defense performance baseline based on real world data

• Designed and implemented an oblivious set intersection protocol that prevents passive attacks

• Future work: defending on active attacks
Thank you!

Paper and video available from [http://ly.tl/p19](http://ly.tl/p19)

Twitter: @elie
Finding units information
When things become harder

• Unit lists are very small
• Visualization won’t work this time to find it :(  
• Solely based on memory shape analysis algorithms
Stack detection heuristics

- Only one new integer by unit
- Each integer is a valid pointer
Unit hack Step

Game memory
Unit hack Step

Game memory
Unit hack Step

Game memory
Unit hack Step

Game memory
Unit hack Step

Game memory
Unit Hack shape

![Unit Hack shape](image)
Understanding unit structure
Understanding unit structure
Understanding unit structure

make it move
Understanding unit structure

make it move
Understanding unit structure

make it move
Understanding unit structure

make it move

make it bleed
Understanding unit structure

make it move

make it bleed
Kartograph performance

The following examples illustrate how diverse the data structures are:

- **Age of Empires 3** uses separate structures to store resource information and visibility information. The information visible to each player is encoded using a bit fields.
- **Supreme Commander 2** encodes the visibility as a short integer representing how many units are able to see a given cell of the map. An example of this cumulative visibility map is visible in figure 13.
- **Civilization 4** uses a filter-based approach. Its visibility map contains for each cell a color that is applied by the game to create the fog of war. Each cell's color filter is encoded as a 32-bit integer with 8 most significant bits used as an alpha channel. Accordingly, unexplored cells contain a straight black mask (0xFF000000) and explored but currently unobserved cells are masked with a gray mask (0xFF6D6D6D).

### B. Unit analysis

While the unit list is an order of magnitude smaller than the map, it is still possible to quickly narrow down its location as shown in Table II. As we find more units, the number of possible locations for the unit list shrinks rapidly. We expected to see units stored in a linked list, which is far from trivial to reverse-engineer. However, most games use a stack either with pointers to units (e.g., Warcraft 3 and Starcraft 2) or with a pointer and a unit ID (e.g., Age of Empires 3). Consequently, instead of using the complex linked-list analysis tools we developed originally, we ended using the simplified algorithm described in the previous section. The last point worth mentioning is that unit hit points are often obfuscated to prevent an attacker from searching memory for the hit point value shown on screen. However, our Adversarial Game Instrumentation technique nullifies this defense, allowing us to quickly reverse-engineer unit structures despite simple obfuscation.

### C. Using the game as a map hack

Since we know the map structures' locations and format, we can take Kartograph one step further and trick the game into displaying the entire map and lifting the fog of war. For example, in Supreme Commander 2, we can lift off all the visibility restriction by changing all the fl's in the cumulative game map into a positive number. Because with Kartograph we are able to rewrite precisely the map structure with a meaningful value, we can not only turn the game into a map hack but also create all sorts of strange effects. For instance, by re-writing only part of the Civilization visibility filter map, we can selectively reveal only part of the map, as shown in figure 14. Note that during online play, some games do static checks on visibility map consistency. In this case, we must either rewrite network packets or write the map hack as an external program that overlays itself on top of the game.

### Table II

<table>
<thead>
<tr>
<th>Game</th>
<th>Unit 1</th>
<th>Unit 2</th>
<th>Unit 3</th>
<th>Unit 4</th>
<th>Unit 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supreme Commander 2</td>
<td>176454</td>
<td>13546</td>
<td>428</td>
<td>55</td>
<td>12</td>
</tr>
<tr>
<td>Age of empire 3</td>
<td>3443</td>
<td>177</td>
<td>48</td>
<td>29</td>
<td>10</td>
</tr>
</tbody>
</table>

V. Preventing Passive Maps

In the previous sections, we discussed Kartograph, an effective tool for extracting strategic information from game memory. The attack is passive in that it only extracts information that the client already has. We now turn to defending against passive information extraction attacks. We first define the passive eavesdropper threat model and then describe OpenConflict, our system that defends against such attacks. We discuss active attacks in section VIII.
Kartograph performances

<table>
<thead>
<tr>
<th>Game</th>
<th>Launch</th>
<th>Play</th>
<th>Discover</th>
<th>Play more</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starcraft 2</td>
<td>850</td>
<td>725</td>
<td>2</td>
<td>1.3</td>
</tr>
<tr>
<td>C&amp;C Tiberium Sun</td>
<td>75M</td>
<td>73M</td>
<td>400K</td>
<td>400K</td>
</tr>
<tr>
<td>C&amp;C Red Alert 2</td>
<td>101M</td>
<td>100M</td>
<td>935K</td>
<td>915K</td>
</tr>
<tr>
<td>C&amp;C Red Alert 3</td>
<td>660M</td>
<td>635M</td>
<td>4.4M</td>
<td>1.6M</td>
</tr>
<tr>
<td>Age of Empire 3</td>
<td>245M</td>
<td>243M</td>
<td>2.7M</td>
<td>2.5M</td>
</tr>
<tr>
<td>Supreme Commander 2</td>
<td>1.2G</td>
<td>629M</td>
<td>2.5M</td>
<td>1.5M</td>
</tr>
<tr>
<td>Civilization IV + ext</td>
<td>340M</td>
<td>293M</td>
<td>2M</td>
<td>1.9M</td>
</tr>
<tr>
<td>Anno 1701</td>
<td>432M</td>
<td>413M</td>
<td>1.9M</td>
<td>1.8M</td>
</tr>
<tr>
<td>Warcraft 3</td>
<td>129M</td>
<td>124M</td>
<td>1.9M</td>
<td>1.8M</td>
</tr>
</tbody>
</table>
Network based maphack
Rewriting network traffic

- Resync the game or get caught
- Use LSP (Layer service provider) to rewrite network traffic
Understanding the network traffic
Understanding the network traffic
Understanding the network traffic

Bucket

Visualize
Understanding the network traffic

Bucket  Visualize  Understand
Understanding the network traffic

Bucket | Visualize | Understand | Resync
Civilization 4 visualization
Civilization 4 visualization

LSP listener

http://ly.tl/p19
Civilization 4 visualization

- **Buckets**
- **LSP listener**

![Civilization 4 visualization with LSP listener and Buckets highlighted]
Civilization 4 visualization
Civilization 4 visualization

Bucket visualization

LSP listener

Buckets
Civilization 4 visualization

Bucket visualization

Buckets

LSP listener

Length

time
Civilization 4 visualization

Bucket visualization

Trace diff map

LSP listener

Buckets

Length

time
Civilization 4 visualization

Bucket visualization

Trace diff map

Fixed value

LSP listener

Buckets

Length

time

Fixed value

Table:

<table>
<thead>
<tr>
<th>Size</th>
<th>Recorded</th>
<th>Isolate Change</th>
<th>Count unchanged</th>
<th>Visualize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size 11</td>
<td>561</td>
<td>Isolate</td>
<td>6</td>
<td>Isolate (o)</td>
</tr>
<tr>
<td>Size 37</td>
<td>1965</td>
<td>Isolate</td>
<td>26</td>
<td>Isolate (o)</td>
</tr>
<tr>
<td>Size 10</td>
<td>13</td>
<td>Isolate</td>
<td>10</td>
<td>Isolate (o)</td>
</tr>
<tr>
<td>Size 25</td>
<td>2</td>
<td>Isolate</td>
<td>29</td>
<td>Isolate (o)</td>
</tr>
<tr>
<td>Size 5</td>
<td>17</td>
<td>Isolate</td>
<td>5</td>
<td>Isolate (o)</td>
</tr>
</tbody>
</table>
Civilization 4 visualization

Bucket visualization

Trace diff map

Fixed value

Counter value

LSP listener

Buckets

Length

time

Fixed value

Counter value

Deferred value

Buckets
Civilization 4 visualization

Bucket visualization

Trace diff map

Fixed value

Counter value

random /crypted value

LSP listener

Buckets

Length

time