KEYSIGHT TECHNOLOGIES 看雪

“疫情”时代
网络安全新风险、新机遇

企业信息安全线上沙龙

数学视角看CTF逆向题

大龙猫 看雪学院

内容

1，CTF逆向题的数学结构

2，CTF逆向题的局部算法套路

3，数学视角看反破解手段

CTF的数学结构

从代码到数学

数学结构的意义

1，解决实际问题。

看雪举办的 KCTF 大赛，经常有防守方大佬，因为题目多解被罚分。

还有因此痛失第一名宝座的。我们现在有数学模型来避免多解。

2，结合下一节，从整体到局部，把题目数学化。

CTF题目，就是以输入为变量，经过一系列计算，得出答案正确或答案错误这个二值结果，结果集合记作 {true, false}.

通用数学形式为：
答案正确与否 = f(input)

常见的表现形式是：
答案正确与否 = f(username, key)

等价形式：
答案正确与否 = equal(f(username, key), 常数)
答案正确与否 = equal(f(username, key), g(username))

equal(x, y) 是取值 {true, false} 的函数，当且仅当 x 等于 y 时取值为 true.

把CTF题目表示成映射

f : A -> B
A = 输入的集合，B = {true, false}

f 可以分解为映射的复合。一个具有代表性的例子：
f(username, key) = equal(g(username, key), h(username))

g, h 可以进一步分解为映射的复合，比如：
h(username) = map[to_number(用户名) % 1024)], map为数组

h可以分解为：　　h = h2 h1 T
其中，T(x) = to_number(x), h1(x) = x % 1024, h2(x) = map[x]

T 往往为转码，把字符串转化为可以计算的数据往往是第一步。

对于任意题目，我们写成映射的形式

$f : A \to B \qquad f = f_n\ f_{n-1} \cdots f_3\ f_2\ f_1\ f_0$

$$A \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} A_3 \xrightarrow{f_3} \cdots \xrightarrow{f_{n-1}} A_n \xrightarrow{f_n} B$$

避免多解，就是让 B 取值为 true 时，对应唯一的 x in A.

f : A -> B　　f = fn fn-1 … f3 f2 f1 f0

f0　f1　f2　f3　fn-1　fn
A -> A1 -> A2 -> A3 -> … -> An -> B

B = {true, false}
在 An 和 B 之间构造双射是没有意义的。如果定义成双射，那么 An 就只有两个元素，穷举空间的大小为2，无论算法如何花哨，没有任何难度，这一步映射也就没有任何实际意义。
我们去除没有任何实际意义的映射，不失一般性，我们认为 An 中的元素数量是足够大到使蛮力穷举不可行的。B 中的 false 必然对应多个 An 中的元素。再考虑到 An 中只能由一个元素对应 B 中的 true.
我们可以得到：fn = equal

一般情况下，我们还可以认为 f0 = T，即字符串转数据

由之前的讨论，我们得到如下数学表达

```
f : A -> B          f = equal fn-1 ⋯ f3 f2 f1 T
```

```
  T      f1     f2     f3       fn-1          equal
A -> A1 -> A2 -> A3 -> ⋯    ->    An      ->    B
```

这就是 CTF 题基本的数学结构。

我们把 B 中的 true 的原象，和原象的原象，一直到最初的正确输入，组成的链，称之为酷链，链上的所有元素，称为酷元素。因为由这个链可以拿到 flag，这很酷。

这个定义不仅酷，也是非常有用的！

```
f : A -> B          f = equal fn-1 … f3 f2 f1 T

  T      f1     f2     f3       fn-1          equal
A -> A1 -> A2 -> A3 -> …    ->    An      ->    B
```

CTF 题要注意以下几点：

1，唯一解的充要条件是酷链唯一。

2，酷链不唯一，要分析构造碰撞的难度。碰撞就是，对于某个 fk，存在 a != b，使得 fk(a) = fk(b).

3，非单射的使用可以增加题目的灵活性和难度。

4，非单射可以转化为单射，方法是缩小定义域，也就是缩小该步骤的有效输入范围。

5，满射保证有解，这个对于题目规则来说不是必要的。

例子

```
char input[256] = {0};
GetDlgItemTextA(hDlg, nIDDlgItem, input, 255); // 从文本框获取用户输入
char x = atoi(input);
if (x + 1 == ‘B’)
    return true;
else
    return false;
```

不考虑函数失败，能找到多少解？

能找到多少类型的解？

例子

```
char input[256] = {0};
GetDlgItemTextA(hDlg, nIDDlgItem, input, 255); // 从文本框获取用户输入
char x = atoi(input);
if (x + 1 == 'B')
    return true;
else
    return false;
```

我们分解成映射链进行分析。

```
f = equal200  I  Add1  I  T
T(x) = atoi(x) = to_int( to_number(x) )
I(x) = x mod 0x100,     Add1(x) = x + 1,     equal200(x) = equal(x, 'B')
```

攻击T：字符串开头添0，小数，追加无效字符，4字节溢出
攻击I：1字节溢出

5 种花式多解，解有无穷个，只有 +1 是双射。

```
f : A -> B          f = equal fn-1 ··· f3 f2 f1 T

  T      f1     f2     f3       fn-1          equal
A -> A1 -> A2 -> A3 -> ···    ->    An      ->    B
```

CTF 题要注意以下几点：

1，唯一解的充要条件是酷链唯一 -> 映射不要漏，逐个进行双射分析，数学算法要推敲

2，酷链不唯一，要分析构造碰撞的难度 -> 攻方有几个数学高手 / sage 库先碰到那个解

局部算法套路

题目的硬核部分

运算 f 的选取和组织，是题目的核心！

运算可以分为四类：

第一类，密码学算法的有漏洞版本。求密钥或找不动点。

第二类，自定义运算和密码学套路。比如，不进位加法，多项式除法，幂模运算。往往基于群论。

第三类，真·数学题。以丢番图方程为首。

第四类，小游戏。比如，游戏出招，走迷宫。

混淆(obfuscation)给所有的运算提供保护，比如代码膨胀、虚拟机等。

运算的选取和组织，是题目的核心！

在通用混淆之外，难度的来源在哪儿？

第一类和第三类，破解者的密码学/数学水平。

第二类，自定义双目运算和密码学套路，很容易逆向。要使用密码学思想来组织代码。

第四类，自定义游戏类操作。难度在于让人一时难以理解，亦在于寻找游戏对策的难度。大佬摸到门道是很快的。使用密码学思想组织代码可以大大增加难度。

追求趣味性和冷僻性，建议使用第二类和第四类。

我们来欣赏一下重要的密码学思想

混淆和扩散的思想，由信息论之父香农提出。主要应用于对称密钥加密算法（加密密钥和解密密钥相同）。著名的加密算法 AES 基于混淆和扩散的思想。思想亦适用于广义的反破解构造。

密文 = 加密算法(明文，密钥)

混淆(confusion)，密文与（明文，密钥）的关系足够复杂。

扩散(diffusion)，明文中的每一位影响密文中的较多位数，密文中的每一位受到明文中较多位数的影响。

注意：confusion 和 obfuscation 不是一回事。加密和反破解不是一回事。
混淆和扩散的思想不仅限于加密，可以有效的指导反破解工作。

混淆(confusion)，查表，非线性，难以建立高中式的数学表达式。

穷举无视复杂!

扩散(diffusion)，扩大穷举空间，避免局部穷举。

局部穷举练手，可以试试自定义 base64 编码。

局部穷举的存在也是大佬会跌倒的地方。

混淆、扩散思想的弱点在于边界。

对称密钥的弱点在于边界。

在边界上下断点，可以获得密钥。（已知算法，已求得逆运算）

定位了重要边界反推可以获得正确输入。（不需要明确求得逆运算）

数学视角看反破解　前两节的应用

反破解，目的是最大程度上增加破解的劳动量，制造劳动和回报的不对等。

检验方法的有效性 优先于 方法本身

如何检验有没有做到劳动和回报的不对等？

利用混淆、扩散的思想，我们可以做劳动量的量化尝试：

使用代码膨胀、虚拟机等工程手段来处理代码，按混淆和扩散的方法来分析处理结果。

劳动量 = 我们制造的最小的可单独分析的有算法特殊性的代码块的可能的模式数量

（理想评估）劳动量 = 香农熵 = $-\Sigma p*\log(p)$

我们做不到理想评估，无法确定p. 我们只能近似，且无法避免主观，但工程上有作用。

—— 劳动量就是分析这种最小单元的难度。与整体无关。

—— 可拆解（不是最小单元），可识别（可能模式太少）

代码混淆的逆向难度是人力范畴。做的好，也往往是防通解不防特解。

数学算法的逆向难度是机器范畴，破解时间随位数成指数增长，轻松突破宇宙年龄。

尽可能使用数学算法来保护数据，用混淆、扩散的思想保护数学算法的边界。

数学视角看CTF逆向题

大龙猫　看雪学院