# 可以获取到的，关于目标主机的信息（含义及获取方式）

开始对该主机（扫描）的时间（戳） (unsigned long) currenths->StartTime()

结束对该主机（扫描）的时间（戳） (unsigned long) currenths->EndTime()

目标主机ip地址 currenths->NameIP()

目标主机在线状况 (currenths->flags & HOST\_UP) ? "up" : "down"

判断目标主机在线的响应包（比如echo-reply） reason\_str(currenths->reason.reason\_id, SINGULAR)

上一条中响应包的生存时间 currenths->reason.ttl

目标主机ip地址 currenths->targetipstr()

目标主机的地址类型 (o.af() == AF\_INET) ? "ipv4" : "ipv6"

目标主机的mac地址 参见print\_MAC\_XML\_Info(Target \*currenths)函数（base.cpp）

本次扫描的时延 " (%ss latency)",

num\_to\_string\_sigdigits(currenths->to.srtt / 1000000.0, 2)

以下参见printportoutput函数（这部分信息获取比较复杂）

Extraports的状态 "%s", statenum2str(istate)

Extraports的数量 "%d", plist->getStateCounts(istate)

以下参见printportoutput中的print\_xml\_state\_summary函数（portreasons.cpp）

Extraports的reason "%s", reason\_str(currentr->reason\_id, currentr->count)

端口号/端口状态（open/closed）/协议（例如tcp）/服务信息（例如http）

"%d/%s/%s//%s//", current->portno,

state, protocol, serviceinfo

判断该端口活动状态的响应包类型reason（例如：syn-ack） reason\_str(current->reason.reason\_id, SINGULAR)

上述响应包的ttl "%d", current->reason.ttl

以下参见write\_xml\_osmatch函数（osscan.cpp）

操作系统名称（例如 Linksys BEFW11S4 WAP） "%s", match->OS\_name

操作系统探测精准度（例如96） "%d", (int) (accuracy \* 100)

匹配到os数据库文件的行数（例如43941） "%d", match->line

以下参见write\_xml\_osclass函数（osscan.cpp）

设备类型（例如WAP） "%s", osclass->Device\_Type

设备厂商（例如Linksys） "%s", osclass->OS\_Vendor

os家族（例如embedded） "%s", osclass->OS\_Family

os的cpe "%s", osclass->cpe[i]

形同Aggressive OS guesses: Linksys BEFW11S4 WAP (96%%), Microsoft Windows Server 2003 R2 (90%%), Linksys BEFSR41 router (90%%), Asus RT-53N WAP (87%%), Cisco ASA 5510 firewall (PIX OS 8.2) (87%%), Cisco 6506 router (IOS 12.2) (87%%), Cisco C2960 switch (IOS 12.2) (87%%), Cisco NX-OS 4.0(1a)N1(1) (87%%), Citrix NetScaler load balancer (87%%), HP-UX B.11.11 - B.11.23 (87%%)

的输出（osscan.cpp line:4819）

if ((o.osscan\_guess || reason) && FPR->num\_matches > 0) {

// Print the best guesses available

//log\_write(LOG\_PLAIN, "Aggressive OS guesses: %s (%.f%%)",

// FPR->matches[0]->OS\_name, floor(FPR->accuracy[0] \* 100));

std::cout<<"Aggressive OS guesses: "<<FPR->matches[0]->OS\_name<<" ("<<floor(FPR->accuracy[0]\*100)<<"%%)";

for (i = 1; i < 10 && FPR->num\_matches > i && FPR->accuracy[i] > FPR->accuracy[0] - 0.10; i++)

//log\_write(LOG\_PLAIN, ", %s (%.f%%)", FPR->matches[i]->OS\_name, floor(FPR->accuracy[i] \* 100));

std::cout<<", "<<FPR->matches[i]->OS\_name<<" ("<<floor(FPR->accuracy[i]\*100)<<"%%)";

//log\_write(LOG\_PLAIN, "\n");

std::cout<<"\n";

}