

به نام خدا

مهندسی معکوس نرم افزار

تألیف: سید داود ملک حسینی

(گروه امنیتی هامان)

انتشارات پندار پارس

سرشناسه	: ملک حسینی، سید داود، ۱۳۶۷-
عنوان و نام پدیدآور	: مهندسی معکوس نرم افزار/ تالیف داود ملک حسینی.
مشخصات نشر	: تهران: پندار پارس، ۱۳۹۸.
مشخصات ظاهری	: ۲۷۰ ص.: مصور، جدول.
شابک	: 978-600-8201-66-3
وضعیت فهرست نویسی	: فیبا
یادداشت	: کتابنامه.
موضوع	: مهندسی معکوس-- نرم افزار
موضوع	: Reverse engineering-- Software
رده بندی کنگره	: ۱۳۹۷ م۶۶/۵/۱۶۸TA
رده بندی دیویی	: ۰۰۴۲۰۲۸۵/۶۲۰
شماره کتابشناسی ملی	: ۵۴۶۱۴۰۰

انتشارات پندار پارس



دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ www.pendarepars.com
 تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۱۲۲۴۵۲۳۴۸
info@pendarepars.com



نام کتاب	: مهندسی معکوس نرم افزار
ناشر	: انتشارات پندار پارس
تالیف	: سید داود ملک حسینی
چاپ نخست	: بهمن ۹۸
شمارگان	: ۵۰۰ نسخه
طرح جلد	: سارا یعسوبی
چاپ، صحافی	: روز

قیمت : ۲۵۰.۰۰۰ تومان شابک : ۹۷۸-۶۰۰-۸۲۰۱-۶۶-۳



* هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد *

فهرست

۳	بخش نخست؛ کیژن (Keygen).....
۵	فصل نخست؛ دو روش در ساخت کیژن مربوط به الگوریتم MD5.....
۵	مقدمه.....
۶	کیژن MD5.....
۶	۱-۱ پیشگفتار.....
۶	۱-۲ پیش از شروع.....
۸	۱-۳ آمادگی لازم برای آغاز کار.....
۱۰	۱-۴ تجزیه و تحلیل الگوریتم.....
۲۱	۱-۵ پیدا کردن یک کلید معتبر.....
۲۶	۱-۶ معکوس کردن الگوریتم.....
۲۷	۱-۷ حرف آخر.....
۲۸	روش دوم کیژن نمودن MD5.....
۲۸	۱-۸ چکیده.....
۲۸	۱-۹ هدف.....
۲۸	۱-۱۰ تجزیه و تحلیل هدف.....
۲۹	۱-۱۱ پیدا کردن روال و الگوی تأیید (verification).....
۳۰	۱-۱۲ پیدا کردن شروع الگوی MD5.....
۳۱	۱-۱۳ پیدا کردن رشته هایی که هش شده اند.....
۳۲	۱-۱۴ استفاده از MD5 hash.....
۳۲	۱-۱۵ خلاصه مطلب.....
۳۳	۱-۱۶ نتیجه گیری.....
۳۵	فصل دوم؛ Keygenning GameShield.....
۳۵	مقدمه.....
۳۶	موارد مورد نیاز.....
۳۶	۲-۱ نرم افزار آزمایشی.....
۳۶	۲-۲ ابزارهای مورد نیاز.....
۳۶	۲-۳ تحقیقی در مورد ساختار گواهی.....
۳۷	۲-۴ Authorization Definitions (AD).....
۳۷	۲-۵ AUTHORIZATIONS REQUEST CODES (ARC).....
۳۷	۲-۶ ACTIVATION CODES (AC).....
۳۸	۲-۷ SERIAL NUMBERS.....
۳۹	۲-۷-۱ بازیابی Password Global و License Password.....
۳۹	۲-۸ Dump کردن DLL.....
۳۹	۲-۸-۱ از مشکل Stack Overflow می گذریم.....
۴۰	۲-۹ موقعیت یابی پروسه آنپک.....
۴۱	۲-۱۰ بدست آوردن DLL.....

۴۴	۲-۱۱ بررسی خروجی‌ها.....
۴۵	۲-۱۲ بدست آوردن گواهی‌های توابع.....
۴۶	۲-۱۳ نوشتن رباینده پسورد.....
۴۷	۲-۱۴ تعریف الزامات.....
۴۷	۲-۱۴-۱ آغاز کردن پروسه.....
۴۸	۲-۱۵ تنظیم Virtual Protect و PAGE-NOACCESS.....
۴۸	۲-۱۶ جست‌وجو کردن توابع مربوط به آنپک.....
۵۰	۲-۱۷ انتظار برای عملکرد نقاط ترمز (Break points).....
۵۰	۲-۱۸ پاک کردن و بستن.....
۵۱	۲-۱۹ خروجی Gameshield 4.6.....
۵۱	۲-۲۰ بدست آوردن پسورد مجاز و معتبر.....
۵۱	۲-۲۰-۱ رمز گشایی گواهی.....
۵۲	۲-۲۰-۲ ساختار فایل گواهی.....
۵۵	۲-۲۱ استفاده از Delphi Parser.....
۵۶	۲-۲۲ انتخاب یک Authorization Definition.....
۵۸	۲-۲۳ ساختن یک کلید.....
۵۸	۲-۲۴ رمزگشایی یک ARC.....
۶۱	۲-۲۵ ساختن یک کد فعال‌سازی.....
۶۳	۲-۲۶ نتیجه‌گیری.....
۶۵	بخش دوم: آنپک کردن (Unpack).....
۶۷	فصل سوم: آنپک کردن HASP SL.....
۶۷	چکیده.....
۶۷	۳-۱ یک نرم‌افزار آزمایشی.....
۶۷	۳-۲ چگونگی انجام پروسه آنپک.....
۷۴	نتیجه‌گیری.....
۷۵	فصل چهارم: آنپک کردن PECompact v2.79 beta.....
۷۵	مقدمه.....
۷۵	۴-۱ آنپک کردن.....
۸۳	۴-۲ گزینه‌ها و موارد بیشتر.....
۸۹	فصل پنجم: آنپک molbox.....
۹۷	۵-۲ نگاهی کلی به Virtual DLL#2.....
	۵-۳ بحثی در مورد OEP مربوط به Molebox، تصحیح صدا زدن‌ها، مخفی کردن فایل
۱۰۰	ها.....
۱۰۳	۵-۴ فایل‌های پنهان شده.....
۱۰۹	فصل ششم: آنپک کردن Private exe protector (PEP).....
۱۱۰	۶-۱ مقدمه.....
۱۱۲	۶-۲ تعیین مکان و موقعیت OEP.....
۱۱۳	۶-۳ OEP.....

۱۱۷	۶-۴ محافظت وارد شده
۱۱۹	۶-۵ محافظت anti dumpt
۱۲۰	۶-۶ PE Header
۱۲۱	۶-۷ Code Section Section Header
۱۲۲	۶-۸ ابهام سازی
۱۲۶	۶-۹ بازکردن محافظ
۱۲۷	۶-۱۰ پارامترها
۱۲۷	۶-۱۰-۱ hModule
۱۳۶	۶-۱۰-۲ MORPH macro
۱۴۳	۶-۱۱ Main handler
۱۵۰	۶-۱۲ 0x00 (VM_EXIT) handler
۱۵۱	۶-۱۳ 0x0D handler
۱۵۳	۶-۱۴ 0x0E handler
۱۵۴	۶-۱۵ 0x12 handler
۱۵۵	۶-۱۶ 0x13 handler
۱۵۸	۶-۱۷ نکات پایانی
۱۵۸	چه مواردی باقی مانده ؟
۱۵۸	از Trial استفاده کنیم یا Demo ؟
۱۶۱	فصل هفتم: ExeCryptor
۱۶۲	۷-۱ ابزارهای مورد نیاز
۱۶۲	۷-۲ آنپک کردن و به هم ریختن (dumping)
۱۷۱	۷-۳ بازسازی ورودی‌ها (imports)
۱۷۶	۷-۴ حذف محدودیت‌ها
۱۸۱	۷-۵ loader در بهترین حالت
۱۹۱	۷-۶ نتیجه‌گیری
۱۹۳	بخش سوم: مبحث دانگل کرکینگ
۱۹۵	فصل هشتم: برداشتن قفل سخت‌افزاری Sentinel SuperPro از نرم‌افزارها
۱۹۵	۸-۱ روش‌ها و رویکردهای ممکن: تقلید کننده در مقابل شبیه ساز
۱۹۵	۸-۲ دانگل چگونه کار می‌کند
۱۹۸	۸-۲-۱ تقلید کننده یک دانگل
۱۹۸	۸-۲-۲ تقلید کننده چگونه کار می‌کند؟
۲۰۰	۸-۲-۳ شبیه‌ساز چگونه کار می‌کند؟
۲۰۱	۸-۳ دیس اسمبل کردن یک برنامه محافظت شده توسط Sentinel
۲۰۲	۸-۳-۱ ت دیس اسمبل کردن توسط IDA
۲۰۷	۸-۳-۲ دیس اسمبل کردن توسط OllyDbg
۲۰۹	۸-۴ برخی جزئیات در مورد برنامه‌نویسی رابط کاربری Sentinel Applications
۲۰۹	۸-۴-۱ Sentinel SuperPro چیست؟
۲۱۰	۸-۴-۲ ساختار حافظه کلید

۲۱۲	سلول‌های انتخاب شده و قابل برنامه نویسی
۲۱۲	کدهای دسترسی (Access Code)
۲۱۴	انواع سلول
۲۱۴	۸-۴-۳ مرجع تابع API
۲۱۸	۸-۵ نوشتن مجدد روی API‌های Sentinel
۲۱۸	۸-۵-۱ SproFormatPacket
۲۲۰	۸-۵-۲ SproFindFirstUnit
۲۲۱	۸-۵-۳ SproOverWrite
۲۲۱	۸-۵-۴ SproFindNextUnit
۲۲۲	۸-۵-۵ SproRead
۲۲۹	۸-۵-۲ رویکرد SproRead
۲۳۱	۸-۵-۳ SPROQuery
۲۳۶	۸-۶ موارد بیشتر
۲۳۷	فصل نهم: یک بررسی عمیق‌تر - HASP SL
۲۳۷	مقدمه
۲۳۷	۹-۱ بدست آوردن نرم‌افزار آزمایشی
۲۳۷	۹-۲ ترفندهایی برای دور زدن آنتی-دیبگ به صورت دستی
۲۴۶	۹-۳ چه نسخه‌ای را بکار ببریم؟
۲۴۷	۹-۴ ترفندهای جدید HASP در Minitab 15

پیشگفتار مؤلف

کتابی که در دستان شماست شامل کلیدی‌ترین و تخصصی‌ترین مباحث مهندسی معکوس نرم‌افزار در شاخه کرک می‌باشد که پس از تست کامل و دقیق همه مثال‌ها و دستورات مندرج در کتاب، اقدام به چاپ آن نمودیم. جلد دوم آن نیز در دست آماده‌سازی است که امیدوارم به زودی به عرصه چاپ و نشر برسد.

اینجانب سید داود ملک حسینی دارای بورس تخصصی هک و امنیت و مهندسی معکوس نرم‌افزار می‌باشم و نیز مسلط به زبان‌های برنامه‌نویسی سطح بالا و سطح پایین و متخصص در شبکه‌های مخابراتی و ارتباطی هستم. این کتاب، حاصل دانش چندین ساله و سال‌ها تجربه‌ام در این زمینه است و آنرا به تمام اعضای جامعه مهندسی کامپیوتر تقدیم می‌کنم و امیدوارم روزی کشور عزیزمان به سمت پیشرفت واقعی در مسیر علوم کامپیوتر برود.

جا دارد از همه عزیزانی که طی این سال‌ها حامی و پشتیبانم بوده‌اند، به‌ویژه پدر و مادر گرانقدرم تقدیر و تشکر نمایم. همچنین، بدون انجام زحمات سرکار خانم مهندس مریم قارونی، فارغ التحصیل کارشناسی ارشد امنیت اطلاعات از مالزی، این کتاب به مرحله تألیف و چاپ نمی‌رسید و لازم می‌دانم از ایشان قدردانی کنم و امیدوارم در تمام مراحل زندگی موفق و پیروز باشند.

به امید رسیدن به فردایی بهتر و ایرانی پیشرفته و قدرتمند

در پناه حق

سید داود ملک حسینی

گروه امنیتی هامان

بخش نخست

کیژن (Keyggen)

فصل نخست

دو روش در ساخت کیژن مربوط به الگوریتم MD5

مقدمه

مجاز هستید همه کدهای ارائه شده در این آموزش را استفاده کرده و تغییر دهید. تنها از شما می‌خواهیم به مکانی که آنها را پیدا کرده‌اید اشاره‌ای داشته باشید. به علاوه، این آموزش با تمام الحاقیات و مکمل‌هایش و در همین قالب فعلی و بدون تغییر، جهت توزیع، مجاز و آزاد محسوب می‌شود.

تمام برنامه‌های تجاری استفاده شده در این آموزش تنها به جهت اثبات راه کارها و تئوری‌های موجود، آورده شده‌اند و هیچ توزیعی از نرم‌افزارهای وصله نرم‌افزاری^۱ شده تحت هیچ رسانه‌ای^۲ انجام نشده است. نرم‌افزارهای استفاده شده بیشتر وقت‌ها توسط دیگر افراد، قبلاً وصله نرم‌افزاری شده‌اند و نسخه‌های کرک شده آنها از مدت‌ها پیش قابل استفاده و دسترسی بوده‌اند. گروه "هامان" یا نویسندگان این کتاب نباید بابت خسارت وارد شده به حقوق شرکت‌های آن برنامه‌ها مسئول شناخته شوند. هدف اصلی این مطلب همانند دیگر آموزش‌های گروه هامان، انتشار و اشتراک دانش و آموزش چگونگی پیچ کردن نرم‌افزارها، نحوه دور زدن محافظت‌های مختلف و به طور کلی توضیح این است که چگونه هنر مهندسی معکوس را ارتقاء دهیم. در واقع ما هیچ نرم‌افزار کرک شده‌ای را منتشر نمی‌کنیم.

¹ patch
² media

کیژن MD5

۱-۱ پیشگفتار

در طول زمان برنامه‌ها و الگوریتم‌های محافظتی زیادی آمدند و رفتند و هر کدام به نوعی قابل شکسته شدن و آنپک شدن بودند اما در این میان تکنیک کیژن برای ما راهکاری جدید و جدی محسوب می‌شود. این آموزش راه کاملی را به شما نشان نمی‌دهد در واقع افرادی که مقداری دانش در زمینه مهندسی معکوس دارند قادرند که این موضوع محافظت را متوجه شوند اما اگر شما یک تازه وارد هستید این آموزش نمی‌تواند مناسب شما باشد.

ابزارهای مورد استفاده عبارتند از یک pc و نسخه v1.10 نرم افزار OllyDebug.

۱-۲ پیش از شروع

در آموزش ارائه شده، خواهید دید که توضیحاتی را در مورد دیس اسمبل^۱ ارائه می‌دهیم. در این حالت نیاز است که نرم‌افزار هدف ما از الگوریتم MD5 hashing استفاده کند. اما چگونه متوجه شویم؟ در ابتدا نیاز است که برنامه را خط به خط دنبال کنیم و نهایتاً به قطعه‌های مشخصی از کد خواهیم رسید که شما را به سمت جوابتان راهنمایی می‌کنند.

در زیر نخستین نشانه را که از RFC1351 گرفته شده است (<http://tools.ietf.org/html/rfc1321>)

ببینید:

```

00CFA540 | $ 8B4424 04 | MOV EAX,DWORD PTR SS:[ESP+4]
00CFA544 | . 33C9      | XOR ECX,ECX
00CFA546 | . C700 01234567 | MOV DWORD PTR DS:[EAX],67452301
00CFA54C | . C740 04 89AB | MOV DWORD PTR DS:[EAX+4],EFCDA889
00CFA553 | . C740 08 FEDC | MOV DWORD PTR DS:[EAX+8],98BADCFE
00CFA55A | . C740 0C 7654 | MOV DWORD PTR DS:[EAX+C],10325476
00CFA561 | . 8948 10     | MOV DWORD PTR DS:[EAX+10],ECX
00CFA564 | . 8948 14     | MOV DWORD PTR DS:[EAX+14],ECX
00CFA567 | . 8948 58     | MOV DWORD PTR DS:[EAX+58],ECX
00CFA56A | . C3        | RETN
00CFA56B | . 00        | NOP

```

¹ Disasmble

```

/* MD5 initialization. Begins an MD5 operation, writing a new context.
*/
void MD5Init (context)
MD5_CTX *context; /* context */
{
    context->count[0] = context->count[1] = 0;
    /* Load magic initialization constants.
*/
    context->state[0] = 0x67452301;
    context->state[1] = 0xefcdab89;
    context->state[2] = 0x98badcfe;
    context->state[3] = 0x10325476;
}

```

این هم از اثبات نهایی:

00CFAS70	\$	8B5424 0C	MOV EDX,DWORD PTR SS:[ESP+C]	
00CFAS74	.	8B4C24 04	MOV ECX,DWORD PTR SS:[ESP+4]	
00CFAS78	.	53	PUSH EBX	
00CFAS79	.	8BC2	MOV EAX,EDX	
00CFAS7B	.	8B59 08	MOV EBX,DWORD PTR DS:[ECX+8]	
00CFAS7E	.	55	PUSH EBP	
00CFAS7F	.	8B69 04	MOV EBP,DWORD PTR DS:[ECX+4]	
00CFAS82	.	4A	DEC EDX	
00CFAS83	.	56	PUSH ESI	
00CFAS84	.	8B71 0C	MOV ESI,DWORD PTR DS:[ECX+C]	
00CFAS87	.	85C0	TEST EAX,EAX	
00CFAS89	.	0F84 C606000	JE 00CFAC55	
00CFAS8F	.	8B4424 14	MOV EAX,DWORD PTR SS:[ESP+14]	
00CFAS93	.	57	PUSH EDI	
00CFAS94	.	83C0 38	ADD EAX,38	
00CFAS97	.	42	INC EDX	
00CFAS98	.	895424 18	MOV DWORD PTR SS:[ESP+18],EDX	
00CFAS9C	>	8B78 C8	MOV EDI,DWORD PTR DS:[EAX-38]	
00CFAS9F	.	8BD6	MOV EDX,ESI	
00CFASA1	.	33D3	XOR EDX,EBX	
00CFASA3	.	23D5	AND EDX,EBP	
00CFASA5	.	33D6	XOR EDX,ESI	
00CFASA7	.	03D7	ADD EDX,EDI	
00CFASA9	.	8BF8	MOV EDI,EBX	
00CFASAB	.	8BF2	MOV ESI,EDX	
00CFASAD	.	8B11	MOV EDX,DWORD PTR DS:[ECX]	
00CFASAF	.	33FD	XOR EDI,EBP	
00CFASB1	.	8D9416 78A46	LEA EDX,DWORD PTR DS:[ESI+EDX+D76AA478]	Look at this
00CFASB3	.	8B70 CC	MOV ESI,DWORD PTR DS:[EAX-34]	
00CFASB8	.	C1C2 07	ROL EDX,7	
00CFASBE	.	03D5	ADD EDX,EBP	
00CFASC0	.	23FA	AND EDI,EDX	
00CFASC2	.	33FB	XOR EDI,EBX	
00CFASC4	.	03FE	ADD EDI,ESI	
00CFASC6	.	8B71 0C	MOV ESI,DWORD PTR DS:[ECX+C]	
00CFASC9	.	8DB437 56B7C	LEA ESI,DWORD PTR DS:[EDI+ESI+E8C7B756]	Look at this
00CFASD0	.	8BFD	MOV EDI,EBP	
00CFASD2	.	C1C6 0C	ROL ESI,0C	
00CFASD5	.	03F2	ADD ESI,EDX	
00CFASD7	.	33FA	XOR EDI,EDX	
00CFASD9	.	23FE	AND EDI,ESI	
00CFASDB	.	33FD	XOR EDI,EBP	
00CFASDD	.	8B68 D0	MOV EBP,DWORD PTR DS:[EAX-30]	
00CFASE0	.	03FD	ADD EDI,EBP	
00CFASE2	.	8BEE	MOV EBP,ESI	
00CFASE4	.	33EA	XOR EBP,EDX	
00CFASE6	.	8DBCF 0B702	LEA EDI,DWORD PTR DS:[EDI+EBX+242070DB]	Look at this
00CFAC50	.	8B59 08	MOV EBX,DWORD PTR DS:[ECX+8]	

این شکل نیز از RFC1351 گرفته شده است.

```

/* MDS basic transformation. Transforms state based on block.
*/
static void MD5Transform (state, block)
UINT4 state[4];
unsigned char block[64];
{
    UINT4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];

    Decode (x, block, 64);

    /* Round 1 */
    FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
    FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
    FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
    FF (b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
    FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
    FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
    FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
    FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
    FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
    FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
    FF (c, d, a, b, x[10], S13, 0xfffff5bb1); /* 11 */
    FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
    FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
    FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
    FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
    FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */

```

این مدرکی است که ثابت می‌کند ما در نخستین دور از MD5 هستیم. همچنین می‌توانید از پلاگین IDA findcrypt نیز استفاده کنید. البته ابزارهای دیگری نیز وجود دارند که به صورت مشابه عمل می‌کنند اما اینکه شما قادر باشید خودتان MD5 را تشخیص دهید در وقتتان صرفه جویی کرده‌اید.

۱-۳ آمادگی لازم برای آغاز کار

فایل drg2pdf.exe را اجرا کنید و به قسمت Registration dialog بروید. در شکل زیر پنجره‌ای را می‌بینید.



وقتی قسمت نام (name) و سریال (serial) را پر کردید دکمه ok را بزنید. پیامی¹ را دریافت خواهید کرد با این مضموم که سریال نامعتبر است. بنابراین باید بریک پوینت‌هایی² را در MessageBOXA و MessageBOXEXA و MessageBOXEXW قرار دهیم و دوباره برنامه را اجرا کنیم. ما آنرا در MessageBOXAEXW به دام انداختیم. دوباره برنامه را جست‌وجو³ کنید و در قسمت code به انتها برسید.

00C72B76	. 804D EC	LEA ECX,DWORD PTR SS:[EBP-14]	
00C72B79	. 8845 F3	MOV BYTE PTR SS:[EBP-D],AL	
00C72B7C	. 6A 11	PUSH 11	
00C72B7E	. E8 BEB1600	CALL 00DDE741	[Arg1 = 00000011 edocpdfp.00E1E741
00C72B83	. 399E AC00000	CMF DWORD PTR DS:[ESI+AC],EBX	
00C72B89	.. 76 6D	JBE SHORT 00C72BF8	From here
00C72B8B	. 385D F3	CMF BYTE PTR SS:[EBP-D],BL	
00C72B8E	.. 74 68	JE SHORT 00C72BF8	From here
00C72B90	. 8B86 C400000	MOV EAX,DWORD PTR DS:[ESI+C4]	
00C72B96	. 3D 88000000	CMF EAX,88	
00C72B98	.. 74 0E	JE SHORT 00C72BAB	
00C72B9D	. 3D 80160000	CMF EAX,1680	
00C72BA2	.. 75 54	JNZ SHORT 00C72BF8	From here
00C72BA4	.. 68 FD000000	PUSH 0FD	
00C72BA9	.. EB 45	JMP SHORT 00C72BF0	
00C72BAB	> 68 FE000000	PUSH 0FE	
00C72BB0	.. EB 3E	JMP SHORT 00C72BF0	
00C72BB2	> 8B45 B8	MOV EAX,DWORD PTR SS:[EBP-48]	
00C72BB5	. 3BC3	CMF EAX,EBX	
00C72BB7	.. 75 05	JNZ SHORT 00C72BBE	
00C72BB9	. B8 EC88E000	MOV EAX,00E088EC	
00C72BBE	> 50	PUSH EAX	
00C72BBF	. 6A 06	PUSH 6	
00C72BC1	. E8 81E7FEFF	CALL 00C61347	[Arg1 = 00000006 edocpdfp.00CA1347
00C72BC6	. 59	POP ECX	
00C72BC7	. 8BF8	MOV EDI,EAX	
00C72BC9	. 59	POP ECX	
00C72BCA	. E8 29990500	CALL 00CCC4F8	
00C72BCF	. 85C0	TEST EAX,EAX	
00C72BD1	.. 75 39	JNZ SHORT 00C72C0C	
00C72BD3	. 81FF 3730000	CMF EDI,3037	
00C72BD9	.. 74 10	JE SHORT 00C72BEB	
00C72BD8	. 81FF 3830000	CMF EDI,3038	
00C72BE1	.. 74 08	JE SHORT 00C72BEB	
00C72BE3	. 81FF 3E30000	CMF EDI,303E	
00C72BE9	.. 75 21	JNZ SHORT 00C72C0C	
00C72BEB	> 68 CE000000	PUSH 0CE	
00C72BF0	> 804D EC	LEA ECX,DWORD PTR SS:[EBP-14]	
00C72BF3	. E8 49BB1600	CALL 00DDE741	[Arg1 = 00000010 edocpdfp.00E1E741
00C72BF8	> 6A 30	PUSH 30	
00C72BFA	. 8BCE	MOV ECX,ESI	
00C72BFC	. FF75 E8	PUSH DWORD PTR SS:[EBP-18]	
00C72BFF	. FF75 EC	PUSH DWORD PTR SS:[EBP-14]	
00C72C02	. E8 339E1600	CALL 00D0C93A	
00C72C07	.. E9 95000000	JMP 00C72CA1	
00C72C0C	> 6A 10	PUSH 10	
00C72C0E	. 804D EC	LEA ECX,DWORD PTR SS:[EBP-14]	
00C72C11	. E8 30001600	CALL 00DDE741	[Arg1 = 00000010 edocpdfp.00E1E741

Jumps from 00C72B89, 00C72B8E, 00C72BA2

در نظر داشته باشید از هر سه موقعیت موجود، منبع اصلی مورد نظر است و صدا زده می‌شود. بنابراین اجازه دهید کمی جست‌وجو کنیم تا به بهترین قسمت برخورد کنیم.

¹ messagebox

² Break point

³ trace

00C72B98	. 8D45 D4	LEA EAX, DWORD PTR SS:[EBP-2C]	
00C72B9B	. 50	PUSH EAX	
00C72B9C	. 8D45 C4	LEA EAX, DWORD PTR SS:[EBP-3C]	
00C72B9F	. 50	PUSH EAX	
00C72BA0	. E8 EEEBFEEF	CALL 00C61733	
00C72BA5	. 83C4 20	ADD ESP, 20	Let the fun begin
00C72BA8	~ EB 14	JMP SHORT 00C72B5E	
00C72BA9	> 8D45 B4	LEA EAX, DWORD PTR SS:[EBP-4C]	
00C72BAE	. 50	PUSH EAX	
00C72BAE	. 8D45 D4	LEA EAX, DWORD PTR SS:[EBP-2C]	
00C72B51	. 50	PUSH EAX	
00C72B52	. 8D45 C4	LEA EAX, DWORD PTR SS:[EBP-3C]	
00C72B55	. 50	PUSH EAX	
00C72B56	. E8 7EEBFEEF	CALL 00C613D9	
00C72B5B	. 83C4 0C	ADD ESP, 0C	
00C72B5E	> 3AC3	CMP AL, BL	
00C72B60	~ 75 50	JNZ SHORT 00C72BB2	
00C72B62	. 8D45 B4	LEA EAX, DWORD PTR SS:[EBP-4C]	
00C72B65	. 50	PUSH EAX	
00C72B66	. 8D45 D4	LEA EAX, DWORD PTR SS:[EBP-2C]	
00C72B69	. 50	PUSH EAX	
00C72B6A	. 8D45 C4	LEA EAX, DWORD PTR SS:[EBP-3C]	
00C72B6D	. 50	PUSH EAX	
00C72B6E	. E8 66EBFEFF	CALL 00C613D9	
00C72B73	. 83C4 0C	ADD ESP, 0C	
00C72B76	. 8D4D EC	LEA ECX, DWORD PTR SS:[EBP-14]	
00C72B79	. 8845 F8	MOV BYTE PTR SS:[EBP-D], AL	
00C72B7C	. 6A 11	PUSH 11	
00C72B7E	. E8 BEBB1600	CALL 00DDE741	[Arg1 = 00000011 edocpdfa.00E1E741
00C72B83	. 399E AC000000	CMP DWORD PTR DS:[ESI+AC], EBX	
00C72B89	~ 76 6D	JBE SHORT 00C72BF8	From here
00C72B8B	. 385D F3	CMP BYTE PTR SS:[EBP-D], BL	
00C72B8E	~ 74 68	JE SHORT 00C72BF8	From here
00C72B90	. 8B86 C4000000	MOV EAX, DWORD PTR DS:[ESI+C4]	
00C72B96	. 3D 88000000	CMP EAX, 88	
00C72B9B	~ 74 0E	JE SHORT 00C72BAB	
00C72B9D	. 3D 80160000	CMP EAX, 1680	
00C72BA2	~ 75 54	JNZ SHORT 00C72BF8	
00C72BA4	. 68 FD000000	PUSH 0FD	
00C72BA9	~ EB 45	JMP SHORT 00C72BF0	From here

صدا زدن^۱ تابع در 0C72B40 در واقع می‌تواند یک صدا زدن قابل توجه باشد. پس به شروع این پروسه می‌رویم و اتفاقات رخ داده را پیدا می‌کنیم. فعلا عمل جست‌وجو در 0C72B40 را انجام ندهید.

۴-۱ تجزیه و تحلیل الگوریتم

هنگامی که متوجه عملکردمان شدیم در واقع به اینجا رسیده‌ایم:

¹ call

00C72AD4	. 8BCF	MOV ECX,EDI	
00C72AD6	. E8 76541600	CALL 00DD7F51	Get Navn
00C72ADB	. 8B3F	MOV EDI,DWORD PTR DS:[EDI]	
00C72ADD	. 8D4D C4	LEA ECX,DWORD PTR SS:[EBP-3C]	
00C72AE0	. FF77 F8	PUSH DWORD PTR DS:[EDI-8]	
00C72AE3	. 57	PUSH EDI	
00C72AE4	. E8 51CBFEFF	CALL 00C5F63A	Name to Unicode
00C72AE9	. 8B86 A0000000	MOV EAX,DWORD PTR DS:[ESI+A0]	
00C72AEF	. 8D4D A4	LEA ECX,DWORD PTR SS:[EBP-5C]	
00C72AF2	. FF70 F8	PUSH DWORD PTR DS:[EAX-8]	
00C72AF5	. 50	PUSH EAX	
00C72AF6	. E8 3FCBFEFF	CALL 00C5F63A	Companyname to Unicode
00C72AFB	. 8B86 98000000	MOV EAX,DWORD PTR DS:[ESI+98]	
00C72B01	. 8D4D D4	LEA ECX,DWORD PTR SS:[EBP-2C]	
00C72B04	. FF70 F8	PUSH DWORD PTR DS:[EAX-8]	
00C72B07	. 50	PUSH EAX	
00C72B08	. E8 2DCBFEFF	CALL 00C5F63A	Serial to Unicode
00C72B0D	. 399E AC000000	CMP DWORD PTR DS:[ESI+AC],EBX	
00C72B13	~ 76 35	JBE SHORT 00C72B4A	
00C72B15	. 8A86 C8000000	MOV AL,BYTE PTR DS:[ESI+C8]	
00C72B1B	. 50	PUSH EAX	
00C72B1C	. 8D45 B4	LEA EAX,DWORD PTR SS:[EBP-4C]	
00C72B1F	. FFB6 C4000000	PUSH DWORD PTR DS:[ESI+C4]	
00C72B25	. FFB6 C0000000	PUSH DWORD PTR DS:[ESI+C0]	
00C72B2B	. FFB6 BC000000	PUSH DWORD PTR DS:[ESI+BC]	
00C72B31	. FFB6 B8000000	PUSH DWORD PTR DS:[ESI+B8]	
00C72B37	. 50	PUSH EAX	
00C72B38	. 8D45 D4	LEA EAX,DWORD PTR SS:[EBP-2C]	
00C72B3B	. 50	PUSH EAX	
00C72B3C	. 8D45 C4	LEA EAX,DWORD PTR SS:[EBP-3C]	
00C72B3F	. 50	PUSH EAX	
00C72B40	. E8 EEBFEFF	CALL 00C61733	Let the fun begin
00C72B45	. 83C4 20	ADD ESP,20	
00C72B48	~ EB 14	JMP SHORT 00C72B5E	
00C72B4A	> 8D45 B4	LEA EAX,DWORD PTR SS:[EBP-4C]	
00C72B4D	. 50	PUSH EAX	
00C72B4E	. 8D45 D4	LEA EAX,DWORD PTR SS:[EBP-2C]	
00C72B51	. 50	PUSH EAX	
00C72B52	. 8D45 C4	LEA EAX,DWORD PTR SS:[EBP-3C]	
00C72B55	. 50	PUSH EAX	
00C72B56	. E8 7EESFEFF	CALL 00C613D9	
00C72B5B	> 83C4 0C	ADD ESP,0C	
00C72B5E	> 3AC3	CMPL BL	
00C72B60	~ 75 50	JNZ SHORT 00C72BB2	

برخی از توضیحات، توضیحاتی هستند که هنگام دیباگ کردن نرم‌افزار هدف، خودمان اضافه کرده‌ایم. شما می‌توانید آنها را جست‌وجو کنید بدون اینکه بخواهید وارد جزئیات شوید.

جالبترین قسمت دستورات عمل صدا زدن در 0C72B40 و CMP، در 0C72B5E است. اگر عمل جست‌وجو را روی صدا زدن در 0C72B40 انجام دهید خواهید دید که این صدا زدن کلمه FALSE را برمی‌گرداند. این مقدار برگردانده شده سپس در 0C72B5E چک می‌شود. بنابراین عمل صدازدن در 0C72B40 برای روشن (ON) شدن مجبور است که TRUE را برگرداند. جست‌وجو کردن صدا زدن در 0C72B40 ما را در ابتدای کار به اینجا می‌رساند.

00C617A7	. 8B7D 0C	MOV EDI,DWORD PTR SS:[EBP+C]
00C617AA	. C645 FC 04	MOV BYTE PTR SS:[EBP-4],4
00C617AE	. 837F 08 40	CMP DWORD PTR DS:[EDI+8],40
00C617B2	~ 73 07	JNB SHORT 00C617BB
00C617B4	. 32DB	XOR BL,BL
00C617B6	~ E9 FA020000	JMP 00C61AB5
00C617BB	> 6A 20	PUSH 20

نخستین چیزی که به آن برخورد می‌کنیم یک حلقه است. اینطور به نظر می‌رسد که برنامه نویس در اینجا بایستی یک عمل modulo را روی بایت‌های name-string انجام دهد. در عوض به سادگی مقدار بایت را یکی افزایش می‌دهد.

ادامه می‌دهیم و در اینجا متوقف می‌شویم:

00C608FF	> 52	PUSH EDX	Last 32 bytes of typed serial
00C60C00	. 53	PUSH EBX	NULL
00C60C01	. 51	PUSH ECX	Last 32 bytes of typed serial
00C60C02	. 50	PUSH EAX	Name-string
00C60C03	. 3D45 9C	LEA EAX, DWORD PTR SS:[EBP-64]	
00C60C06	. 50	PUSH EAX	
00C60C07	. E8 30280900	CALL 00CF343C	
00C60C08	. 50	PUSH EAX	

بیاید آدرس 0CF343C را جست‌وجو کنیم زیرا به زودی این کار، مهم و ضروری خواهد شد.

00CF3495	> 6A 20	PUSH 20	
00CF3497	. 50	PUSH EAX	
00CF3498	. 8D85 74FFFFFF	LEA EAX, DWORD PTR SS:[EBP-8C]	
00CF349E	. 50	PUSH EAX	
00CF349F	. F8 BC770000	CALL 00CFAC60	Get ciphered name and rest of subkey
00CF34A4	. 6A 20	PUSH 20	
00CF34A6	. 8D85 74FFFFFF	LEA EAX, DWORD PTR SS:[EBP-8C]	
00CF34AC	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	Last 32 bytes of typed serial
00CF34AF	. 50	PUSH EAX	
00CF34B0	. E8 AB770000	CALL 00CFAC60	Append last 32 bytes and do MD5 hash
00CF34B5	. 3D45 14	LEA EAX, DWORD PTR SS:[EBP+14]	
00CF34B8	. 6A 04	PUSH 4	
00CF34BA	. 50	PUSH EAX	
00CF34BB	. 8D85 74FFFFFF	LEA EAX, DWORD PTR SS:[EBP-8C]	
00CF34C1	. 50	PUSH EAX	
00CF34C2	. E8 99770000	CALL 00CFAC60	Place NULL at start of string
00CF34C7	. 6A 10	PUSH 10	
00CF34C9	. 8D85 74FFFFFF	LEA EAX, DWORD PTR SS:[EBP-8C]	
00CF34CF	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	
00CF34D2	. 50	PUSH EAX	
00CF34D3	. E8 98770000	CALL 00CFAC60	Place byte 32-40 of typed serial in string
00CF34D8	. 8D85 74FFFFFF	LEA EAX, DWORD PTR SS:[EBP-8C]	
00CF34DE	. 50	PUSH EAX	
00CF34DF	. 8D45 D0	LEA EAX, DWORD PTR SS:[EBP-30]	
00CF34E2	. 50	PUSH EAX	
00CF34E3	. C0 107A0000	CALL 00CFAF00	Number of times to loop
00CF34E8	. 6A 32	PUSH 32	
00CF34EA	. 8D45 D0	LEA EAX, DWORD PTR SS:[EBP-30]	
00CF34ED	. 6A 10	PUSH 10	
00CF34EF	. 50	PUSH EAX	MD5 digest
00CF34F0	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00CF34F3	. E8 23000000	CALL 00CF351B	Do MD5 hash loop
00CF34F8	. 50	PUSH EAX	

در ادامه تلاش خواهیم کرد که نشان دهیم چه روی داده است.

Address	Hex dump	ASCII
0012E158	33 6C 42 45 28 BF 4E 5E 4E 75 8A 41 64 00 4E 56	31BE(7N^Nw&Ad.NU
0012E168	FF FA 01 08 2E 2E 00 B6 D0 68 3E 80 2F 0C A9 FE	...>h>?/.0m
0012E178	00 00 00 00 80 2C C6 00 01 00 00 00 04 00 00 00	...&.0...+...
0012E188	24 F3 12 00 04 00 00 00 00 F4 12 00 24 F3 12 00	15* 15*

عمل صدا زدن در 0CF34B0. ۳۲ بایت آخر را به name-string اضافه می‌کند و سپس عمل هش^۱ MD5 را روی string انجام می‌دهد.

^۱ hash

Address	Hex dump	ASCII
0012E158	33 6C 42 45 28 BF 4E 5E 4E 75 8A 41 64 00 4E 56	3IBE(7N^NuêAd.NU
0012E168	FF FA 01 08 2E 2E 00 B6 D0 68 3E 80 2F 0C A9 FE	·6...A&h>ç/·0#
0012E178	33 33 33 33 33 33 33 33 33 33 33 33 33 33 33	3333333333333333
0012E188	34 34 34 34 34 34 34 34 34 34 34 34 34 34 34	4444444444444444
0012E198	20 0A 00 00 4C E2 12 00 7C 86 DF 00 00 00 00 00	!0# !&?

در اینجا هش MD5 اجرا شده است:

00CFAD3B	> 6A 01	PUSH 1	
00CFAD3D	. 55	PUSH EBP	Pointer to string
00CFAD3E	. 53	PUSH EBX	MD5 digest
00CFAD3F	. E8 2CF8FFFF	CALL 00CFAS70	MD5 hash

در شکل خلاصه‌ای از آنچه که پس از هش MD5 اتفاق می افتد نشان داده شده است:

Address	Hex dump	ASCII
0012E140	5C 07 9F 4E A1 F7 88 A0 2B 80 E2 C1 23 FA 14 6F	%#Ni·êâ+ç0+ç·?0
0012E150	00 02 00 00 00 00 00 00 33 6C 42 45 28 BF 4E 5E	·0.....3IBE(7N^
0012E160	4E 75 8A 41 64 00 4E 56 FF FA 01 08 2E 2E 00 B6	NuêAd.NU·6...A
0012E170	D0 68 3E 80 2F 0C A9 FE 33 33 33 33 33 33 33 33	%h>ç/·0#33333333
0012E180	33 33 33 33 33 33 33 33 34 34 34 34 34 34 34 34	3333333333333333
0012E190	34 34 34 34 34 34 34 34 20 00 00 00 4C E2 12 00	44444444 ...L0#.
0012E198	7C 86 DF 00 00 00 00 04 00 00 00 00 2B C6 00	!&? *...L0#.

عمل صدا زدن در 0CF34C2 و صدا زدن در 0CF34D3 رشته‌ای (string) طبق شکل زیر می‌سازد:

Address	Hex dump	ASCII
0012E158	00 00 00 00 33 33 33 33 33 33 33 33 33 33 33	...333333333333
0012E168	33 33 33 33 00 00 00 00 00 00 00 00 00 00 00	3333ç.....
0012E178	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012E188	00 00 00 00 00 00 00 00 A0 02 00 00 00 00 00â0.....
0012E198	14 00 00 00 4C E2 12 00 7C 86 DF 00 00 00 00 00	? !0# !&?

دوباره یک هش MD5 اجرا شده است، در نتیجه:

Address	Hex dump	ASCII
0012E140	29 89 B1 62 A4 75 BA 08 DD 30 A4 92 6F BC 0B 0A	!èbku!i!0âèç?â.
0012E150	A0 02 00 00 00 00 00 00 00 00 00 00 33 33 33 33	â0.....3333
0012E160	33 33 33 33 33 33 33 33 33 33 33 33 80 00 00 00	333333333333ç...
0012E170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012E180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012E190	A0 02 00 00 00 00 00 00 14 00 00 00 4C E2 12 00	â0.....?...L0#.

توجه کنید که عملیات صدا زدن پیش از هش کردن این قسمت MD5 را مقداردهی اولیه نمی‌کند!!

عمل صدا زدن¹ در 0CF34F3، یک حلقه MD5 هش را انجام می‌دهد. در این زمان آنها مقداردهی اولیه MD5 را پیش از هش شدن انجام می‌دهند. این حلقه² مانند شکل زیر است:

¹ call

² loop

00CF3545	.	50	PUSH EAX
00CF3546	.	8975 FC	MOV DWORD PTR SS:[EBP-4],ESI
00CF3549	.	FF75 10	PUSH DWORD PTR SS:[EBP+10]
00CF354C	.	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
00CF354F	.	E8 EC7A0000	CALL 00CFB040
00CF3554	.	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]
00CF3557	.	83C4 0C	ADD ESP,0C
00CF355A	.	3BC6	CMP EAX,ESI
00CF355C	.	7E 18	JLE SHORT 00CF3576
00CF355E	.	8D78 FF	LEA EDI,DWORD PTR DS:[EAX-1]
00CF3561	>	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]
00CF3564	.	50	PUSH EAX
00CF3565	.	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]
00CF3568	.	6A 10	PUSH 10
00CF356A	.	50	PUSH EAX
00CF356B	.	E8 D07A0000	CALL 00CFB040
00CF3570	.	83C4 0C	ADD ESP,0C
00CF3573	.	4F	DEC EDI
00CF3574	.	75 EB	JNZ SHORT 00CF3561
00CF3576	>	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]

صدا زدن در 0CF354F و 0CF356B هر دو، هم عملیات مقداره‌ی و هم عملیات هش را انجام می‌دهد. نتیجه حلقه در زیر نشان داده شده است:

Address	Hex dump	ASCII
0012E0BC	A5 09 6C B4 20 27 85 7A 3D 8E 94 B2 E8 D5 A4 F3'az=40&P'8%
0012E0CC	00 2C C6 00 00 00 00 00 00 00 00 00 00 00 00	..š.....
0012E0DC	00 00 00 00 C0 E1 12 00 9B 89 DF 00 01 00 00 00*p#.e=.0...
0012E0EC	00 F1 13 00 F8 34 C5 00 F4 F1 13 00 8C F1 13 00	..e+ 848 7e+ 6e+

این کلید بسیار مهم است و برای معتبر کردن نسخه گواهی (license) استفاده می‌شود که در آینده توضیح خواهیم داد.

هنگامی که ما با یک MD5 تازه برگردیم می‌توانیم عمل جست‌وجو¹ را دوباره ادامه دهیم و در مراحل پیش رو چیزی مشابه شکل زیر می‌بینید:

00C60C4A	>	53	PUSH EBX	
00C60C4B	.	68 80000000	PUSH 80	NULL
00C60C50	.	6A 10	PUSH 10	
00C60C52	.	56	PUSH ESI	MD5 Digest
00C60C53	.	6A 10	PUSH 10	
00C60C55	.	50	PUSH EAX	
00C60C56	.	8D45 9C	LEA EAX,DWORD PTR SS:[EBP-64]	First 32 bytes of typed serial (hex)
00C60C59	.	50	PUSH EAX	
00C60C5A	.	E8 DE360900	CALL 00CF433D	
00C60C5F	.	83C4 1C	ADD ESP,1C	
00C60C65	.	53	PUSH EBX	

بیاید 0CF433D را جست‌وجو کنیم و نهایتاً به اینجا می‌رسیم:

¹ tracing

این صدا زدن در آدرس افست 0CF42EE یک «جدول جادویی»^۱ می‌سازد. جدول ابتدا ساخته می‌شود سپس با استفاده از MD5 digest به هم ریخته و مخلوط می‌گردد. در این مثال شروع جدول همانند این شکل است:

Address	Hex dump	ASCII
0113A220	00 00 00 00 00 00 00 00 C8 00 00 00 90 00 00 00@.....
0113A230	38 00 00 00 F2 00 00 00 6E 00 00 00 37 00 00 007.....
0113A240	59 00 00 00 47 00 00 00 82 00 00 00 60 00 00 00	V...G...n...m...
0113A250	48 00 00 00 18 00 00 00 23 00 00 00 BA 00 00 00	H...↑...#... ...
0113A260	B0 00 00 00 B3 00 00 00 65 00 00 00 4A 00 00 00e...J...
0113A270	80 00 00 00 BF 00 00 00 2E 00 00 00 61 00 00 00	G...7...E...a...
0113A280	EE 00 00 00 39 00 00 00 92 00 00 00 25 00 00 00	...9...E...%...
0113A290	75 00 00 00 6C 00 00 00 C1 00 00 00 9F 00 00 00	u...l...+...f...
0113A2A0	20 00 00 00 F3 00 00 00 8F 00 00 00 E8 00 00 00	...%...A...b...
0113A2B0	FC 00 00 00 D1 00 00 00 31 00 00 00 28 00 00 00	*...0...1...(!...
0113A2C0	5E 00 00 00 FB 00 00 00 A8 00 00 00 9E 00 00 00	^...!...~...x...
0113A2D0	96 00 00 00 89 00 00 00 B4 00 00 00 85 00 00 00	ü...ë...t...ä...
0113A2E0	F9 00 00 00 36 00 00 00 4D 00 00 00 52 00 00 00	...6...M...R...
0113A2F0	14 00 00 00 24 00 00 00 29 00 00 00 49 00 00 00	...\$...)....I...
0113A300	07 00 00 00 09 00 00 00 16 00 00 00 06 00 00 00	...!...&...&...

به 0CF438B بر گردید، دیس اسمبل همانند زیر است:

00CF438B	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	
00CF438E	. 57	PUSH EDI	
00CF4391	. FF75 0C	PUSH DWORD PTR SS:[EBP+C]	First 32 bytes of typed serial (hex)
00CF4394	. FF75 F0	PUSH DWORD PTR SS:[EBP-10]	Magic table
00CF4397	. E8 7CFFFFFF	CALL 00CF4316	

صدا زدن در 0CF3416. کاری را انجام می‌دهد که آنرا به عنوان عملیات XOR توضیح داده‌ایم. خودتان روی آن بررسی کوچکی انجام دهید. اگرچه کمی گیج کننده به نظر می‌رسد اما در حقیقت ساده است. شکل زیر پروسیجر 0CF4316 را نشان می‌دهد:

00CF4316	. 55	PUSH EBP	
00CF4317	. 8BEC	MOV EBP,ESP	
00CF4319	. 8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]	
00CF431C	. 85C8	TEST EAX,EAX	
00CF431E	. 75 09	JNZ SHORT 00CF4329	
00CF4320	. FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
00CF4323	. E8 08E00600	CALL 00D62400	
00CF4328	. 59	POP EAX	
00CF4329	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	Output
00CF432C	. FF75 0C	PUSH DWORD PTR SS:[EBP+C]	First 32 bytes of typed serial (hex)
00CF432F	. 50	PUSH EAX	
00CF4330	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	Magic table
00CF4333	. E8 586D0000	CALL 00CFB090	Get new digest (not MD5)
00CF4338	. 83C4 10	ADD ESP,10	
00CF433B	. 5D	POP EBP	
00CF433C	. C3	RETN	

خروجی همانند این شکل است:

Address	Hex dump	ASCII
01495FD8	4E 85 AF 4D 57 98 C4 04 BF 90 86 0B C1 7C 75 24	Nä»MÜö-4-É&±iu\$
01495FE8	AB AB AB AB AB AB AB AB EE FE EE FE EE FE EE FE	#####
01495FF8	00 00 00 00 00 00 00 00

¹ magic table

باید این احتمال را داده باشید که بدترین قسمت تمام شده است، اما نه، هنوز کار را به اتمام نرسانده‌ایم. به نظر می‌رسد که تولیدکنندگان، این الگوریتم را به منظور پراکنده کردن ذهن ما ساخته‌اند. اما موفق نخواهند شد!

قبلا ما صدا زدن در 0C6191A را دنبال کردیم. اکنون به 0C6191F برگشتیم. کمی به سمت پایین جست‌وجو می‌کنیم تا به این شکل برخورد کنیم:

00C61946	. 8D45 AC	LEA EAX,DWORD PTR SS:[EBP-54]
00C61949	. 50	PUSH EAX
00C6194A	. 8D45 9C	LEA EAX,DWORD PTR SS:[EBP-64]
00C6194D	. 50	PUSH EAX
00C6194E	. 8D45 8C	LEA EAX,DWORD PTR SS:[EBP-74]
00C61951	. FF75 08	PUSH DWORD PTR SS:[EBP+8]
00C61954	. 50	PUSH EAX
00C61955	. E8 1AEFFFFF	CALL 00C60774

عملیات جست‌وجو را در 0C60774 انجام دهید و در ابتدای این پروسه یکبار دیگر عمل هش modulo را روی نام انجام دهید و آنرا در حافظه ذخیره کنید. بعدا به آن خواهیم پرداخت. در ادامه به اینجا می‌رسیم:

00C608C3	> 8D45 B0	LEA EAX,DWORD PTR SS:[EBP-50]
00C608C6	. 50	PUSH EAX
00C608C7	. 8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]
00C608CA	. 50	PUSH EAX
00C608CB	. 8D45 8C	LEA EAX,DWORD PTR SS:[EBP-74]
00C608CE	. 50	PUSH EAX
00C608CF	. E8 5C330900	CALL 00CF3C30

این روند اجرایی در 0CF3C30 همچنان ممکن است برای ما آزار دهنده باشد. اما اجازه دهید که آنرا جست‌وجو کنیم. کمی که جلوتر برویم به شکل زیر می‌رسیم:

00CF3CD9	> 6A 33	PUSH 33	Number of loops
00CF3CDB	. 6A 20	PUSH 20	Size of string to hash
00CF3CDD	. 50	PUSH EAX	
00CF3CDE	. 8D45 B0	LEA EAX,DWORD PTR SS:[EBP-50]	
00CF3CE1	. 50	PUSH EAX	
00CF3CE2	. E8 34F8FFFF	CALL 00CF351B	Do yet another MD5 hash loop

به خاطر داشته باشید که digest جدید شبیه زیر است:

Address	Hex dump	ASCII
0012E09C	4E 85 AF 4D 57 98 C4 04 BF 90 86 0B C1 7C 75 24	Nä»MÜ-«jEg+Iu\$
0012E0AC	28 BF 4E 5E 4E 75 8A 41 64 00 4E 56 FF FA 01 08	(N^NuèAd.NV .6
0012E0BC	80 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ç.....

و این باید در حلقه‌های MD5، ۵۱ بار دیگر اجرا شود. نتیجه این است:

Address	Hex dump	ASCII
0012E0FC	B7 DA 82 59 11 40 3B 91 45 24 DC BC 5D B9 07 8A	AréV40;æE3m"Ji.è
0012E10C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012E11C	00 00 00 00 94 E1 12 00 9B 89 DF 00 01 00 00 00øþ#.øè.ø....

به 0C608D4 برمی‌گردیم و می‌توانیم دوباره کمی جست‌وجو کنیم و به اینجا برسیم:

00CF3D1D	> 6A 20	PUSH 20	
00CF3D1F	. 51	PUSH ECX	Ciphered name-string
00CF3D20	. FF75 E8	PUSH DWORD PTR SS:[EBP-18]	Output size
00CF3D23	. 50	PUSH EAX	MD5 digest
00CF3D24	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00CF3D27	. E8 9AF8FFFF	CALL 00CF35C6	

اگر به جست‌وجو کردن این صدا زدن ادامه دهید خواهید دید که از MD5 digest برای ساختن یک جدول جادویی دیگر استفاده شده است و سپس دوباره عمل XOR انجام می‌گیرد. این قضیه ۲۰ مرتبه انجام شده و نتیجه زیر ظاهر شده است:

Address	Hex dump	ASCII
01495DB8	B8 88 2A 39 17 6D F9 66 CA 60 E3 28 77 47 C6 3A	0è#9#m-f#*d(w0â:
01495DC8	27 1B 07 A1 37 84 0D F3 98 85 24 59 3E 3C DA 83	*+.i7ä.%0â\$Y><rä
01495DD8	AB AB AB AB AB AB AB AB 00 00 00 00 00 00 00	%%%%%%%.....

سرانجام این برنامه پایان پذیرفت. اکنون ما به 0C608D4 برمی‌گردیم. کمی که جلوتر برویم این شکل را خواهیم دید:

00C608F9	. 33FF	XOR EDI,EDI	
00C608FB	. 80B5 68FFFFFF	LEA ESI,DWORD PTR SS:[EBP-98]	
00C60901	> 397D C8	CMP DWORD PTR SS:[EBP-38],EDI	
00C60904	~ 72 14	JB SHORT 00C6091A	
00C60906	. 395D C4	CMP DWORD PTR SS:[EBP-3C],EBX	
00C60909	~ 74 0F	JE SHORT 00C6091A	
00C6090B	. 8D4D C0	LEA ECX,DWORD PTR SS:[EBP-40]	
00C6090E	. E8 50210000	CALL 00C62A63	
00C60913	. 8B45 C4	MOV EAX,DWORD PTR SS:[EBP-3C]	
00C60916	. 03C7	ADD EAX,EDI	
00C60918	~ EB 05	JMP SHORT 00C6091F	
00C6091A	> B8 0C99E000	MOV EAX,00E0990C	
00C6091F	> 0FB600	MOVZX EAX,BYTE PTR DS:[EAX]	
00C60922	. 50	PUSH EAX	
00C60923	. 68 141BE600	PUSH 00E61B14	ASCII "%02x"
00C60928	. 56	PUSH ESI	
00C60929	. E8 1C381000	CALL 00D6414A	
00C6092E	. 83C4 0C	ADD ESP,0C	
00C60931	. 47	INC EDI	
00C60932	. 46	INC ESI	
00C60933	. 46	INC ESI	
00C60934	. 83FF 10	CMP EDI,10	
00C60937	^ 7C C8	JL SHORT 00C60901	

۲ عدد حلقه وجود دارد. نخستین آن (حلقه بالایی)، آخرین کلید ما را به یک text-string تبدیل می‌کند و نتیجه زیر پدیدار می‌گردد:

Address	Hex dump	ASCII
0012E1C0	62 38 38 38 32 61 33 39 31 37 36 64 66 39 36 36	b8882a39176df966
0012E1D0	63 61 36 30 65 33 32 38 37 37 34 37 63 36 33 61	ca60e3287747c63a
0012E1E0	00 0C C6 00 00 00 00 00 00 00 00 00 00 00 00	..8.....

حلقه دوم زیاد مهم نیست. حداقل ما نیاز چندانی به آن نداریم!

اکنون تا زمانی که به 0C6195A برگردیم عمل جستوجو را انجام می‌دهیم. در بالا یک حلقه مقایسه‌ای می‌بینیم، این مقایسه‌ای است بایت به بایت از کلید با ۳۲ بایت آخر که در سریال ما تایپ شده است.

00C61967	> 3975 D4	CMP DWORD PTR SS:[EBP-2C],ESI	
00C6196A	·v 72 16	JB SHORT 00C61982	
00C6196C	· 837D D0 00	CMP DWORD PTR SS:[EBP-30],0	
00C61970	·v 74 10	JE SHORT 00C61982	
00C61972	· 804D CC	LEA ECX,DWORD PTR SS:[EBP-34]	
00C61975	· E8 E9100000	CALL 00C62A63	
00C6197A	· 8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]	
00C6197D	· 8D3C30	LEA EDI,DWORD PTR DS:[EAX+ESI]	
00C61980	·v EB 05	JMP SHORT 00C61987	
00C61982	> BF 0C99E000	MOV EDI,00E0990C	
00C61987	> 3975 B4	CMP DWORD PTR SS:[EBP-4C],ESI	
00C6198A	·v 72 1A	JB SHORT 00C619A6	
00C6198C	· 837D B0 00	CMP DWORD PTR SS:[EBP-50],0	
00C61990	·v 74 14	JE SHORT 00C619A6	
00C61992	· 804D AC	LEA ECX,DWORD PTR SS:[EBP-54]	
00C61995	· E8 EC29FEFF	CALL 00C44386	
00C6199A	· 8B45 B0	MOV EAX,DWORD PTR SS:[EBP-50]	
00C6199D	· BB EC88E000	MOV EBX,00E088EC	
00C619A2	· 03C6	ADD EAX,ESI	
00C619A4	·v EB 07	JMP SHORT 00C619AD	
00C619A6	> BB EC88E000	MOV EBX,00E088EC	
00C619AB	· 8BC3	MOV EAX,EBX	
00C619AD	> 8A0F	MOV CL,BYTE PTR DS:[EDI]	byte-to-byte check
00C619AF	· 3A08	CMP CL,BYTE PTR DS:[EAX]	
00C619B1	·v 75 0D	JNZ SHORT 00C619C0	
00C619B3	· 46	INC ESI	
00C619B4	· 83FE 20	CMP ESI,20	
00C619B7	·^ 7C AE	JL SHORT 00C61967	

Registers (FPU)		
EAX	0113A221	ASCII "b8882a39176df966ca60e3287747c63a"
ECX	0012E29C	
EDX	01130608	
EBX	00E088EC	edocpdfp.00E088EC
ESP	0012E270	
EBP	0012E2F0	
ESI	00000000	
EDI	01495C79	ASCII "33333333333333334444444444444444"
EIP	00C619AD	edocpdfp.00C619AD

هم اکنون وقت آن است که قسمت آخر سریال را که باید در جدول جادویی (نامی است که ما بر آن گذاشته‌ایم) محاسبه شود پاک کنید. بهتر است آنچه را که می‌دانیم جمع کرده و به طور خلاصه بیان کنیم:

آخرین ۳۲ بیت از سریال با استفاده از نخستین کلید، نام، MD5 hashing و جدول جادویی محاسبه می‌شود. این عملیات محاسباتی چک می‌شود و اگر نتیجه این محاسبات با آخرین ۳۲ بیت سریال مساوی نباشد خطا دریافت می‌کنیم.

نخستین کلید در واقع از آخرین ۳۲ بیت سریال با استفاده از قسمت نام ما (name) که در تکس باکس است و MD5 hashing محاسبه شده است.

در این وضعیت چگونه می‌توانیم هر کدام از اینها را محاسبه کنیم در حالی که هر دوی آنها به یکدیگر وابسته‌اند؟

این سوال مطرح می‌شود که این کار امکان دارد یا نه؟ از آنجایی که کلید ۱۲۸ بیتی است عمل بروت فورس^۱ خارج محدوده ماست و نیز ساختن یک MD5 collision به نظر دور از تصور است. پس باید چه کار کرد؟

پس از کمی فکر کردن جواب مشخص می‌شود!! پاسخ به این سوال، جدول جادویی و نخستین کلید است. در قسمت‌های بعد چگونگی عملکرد آن را نشان می‌دهیم.

۵-۱ پیدا کردن یک کلید معتبر

می‌دانیم که هدف ما مقایسه‌ای است بایت به بایت روی آخرین ۳۲ بیت از سریال. اگر موفقیت حاصل نشد، برنامه هدف یک AND را اجرا می‌کند و یک موقعیت پیغام خطا را می‌سازد.

00C619AE	. 8BC3	MOV EAX,EBX	
00C619AD	> 8A0F	MOV CL, BYTE PTR DS:[EDI]	byte-to-byte check
00C619AF	. 3A08	CMP CL, BYTE PTR DS:[EAX]	
00C619B1	..v 75 0D	JNZ SHORT 00C619C0	
00C619B3	. 46	INC ESI	
00C619B4	. 83FE 20	CMP ESI,20	
00C619B7	..^ 7C AE	JL SHORT 00C61967	
00C619B9	. BB EC88E000	MOV EBX,00E088EC	
00C619BE	..v EB 04	JMP SHORT 00C619C4	
00C619C0	> 8065 F3 00	AND BYTE PTR SS:[EBP-D],0	Create BAD-BOY

برای اینکه یک کلید معتبر را بدست بیاوریم نیاز است که بررسی‌هایی را انجام دهیم. از نقطه نظر ما این کار در IDA با استفاده از شکل گرافیکی آسان‌تر انجام می‌شود. بنابراین پس از استفاده از قسمت OLLY قسمت IDA خواهد بود.

¹ Brute-forcing

00C619E2	. 8BC3	MOV EAX,EBX
00C619E4	> 50	PUSH EAX
00C619E5	. 57	PUSH EDI
00C619E6	. E8 5CF9FFFF	CALL 00C61347
00C619EB	. 8945 08	MOV DWORD PTR SS:[EBP+8],EAX
00C619EE	. 59	POP ECX
00C619EF	. 66:C1E8 0D	SHR AX,0D
00C619F3	. 66:3D 0100	CMP AX,1
00C619F7	. 59	POP ECX
00C619F8	∨ 75 03	JNZ SHORT 00C619FD
00C619FA	. 8845 F2	MOV BYTE PTR SS:[EBP-E],AL

در شکل بالا key[0] را می‌گیرد (سایز WORD) و بیت به اندازه ۱۳ بیت به سمت راست شیفت پیدا می‌کند. اگر سیزدهمین بیت تنظیم شود آنگاه یک نسخه تستی^۱ را اجرا می‌کنیم و اگر تنظیم نشود، یک نرم‌افزاری واقعی را اجرا می‌کنیم. فعلا قصد نداریم درباره آن با جزئیات توضیح دهیم. خودتان آزادانه آنرا تست کنید. اگر سیزدهمین بیت تنظیم شود، هدف ما تلاش برای این است که متوجه شویم چند روز از این نسخه آزمایشی باقی مانده است و این کار را با استفاده از key[6] (word) انجام می‌دهیم.

00C61A0B	. 8BC3	MOV EAX,EBX	
00C61A0D	> 50	PUSH EAX	
00C61A0E	. 6A 04	PUSH 4	
00C61A10	. E8 32F9FFFF	CALL 00C61347	
00C61A15	. 807D F2 00	CMP BYTE PTR SS:[EBP-E],0	Are we a test-version ?
00C61A19	. 59	POP ECX	
00C61A1A	. 59	POP ECX	
00C61A1B	∨ 74 11	JE SHORT 00C61A2E	
00C61A1D	. 57	PUSH EDI	
00C61A1E	. 50	PUSH EAX	
00C61A1F	. E8 28CD0600	CALL 00CCE74C	
00C61A24	. 59	POP ECX	
00C61A25	. 84C0	TEST AL,AL	
00C61A27	. 59	POP ECX	
00C61A28	∨ 74 04	JE SHORT 00C61A2E	
00C61A2A	. 8065 F3 00	AND BYTE PTR SS:[EBP-D],0	

همانطور که ممکن است متوجه شوید EBP-0E اگر FALSE باشد چک می‌شود. اگر چنین باشد، ادامه می‌دهیم؛ وگرنه تست می‌کنیم که چند روز از نسخه آزمایشی باقی مانده است. اگر چیزی باقی نمانده بود، پیغام خطا را در 0C61A2A می‌سازیم. مرحله بعد key[4] (word) را بررسی می‌کنیم:

¹ test-version

00C61A35	. 8BC3	MOV EAX,EBX	
00C61A37	> 50	PUSH EAX	
00C61A38	. 6A 02	PUSH 2	
00C61A3A	. E8 08F9FFFF	CALL 00C61347	
00C61A3F	. 59	POP ECX	
00C61A40	. 59	POP ECX	
00C61A41	. 33C9	XOR ECX,ECX	
00C61A43	. 8ACC	MOV CL,AH	
00C61A45	. 25 FF000000	AND EAX,0FF	
00C61A4A	. 394D 1C	CMP DWORD PTR SS:[EBP+1C],ECX	CMP ecx == 5
00C61A4D	> 74 00	JE SHORT 00C61A5C	
00C61A4F	. 3BC7	CMP EAX,EDI	
00C61A51	> 74 05	JE SHORT 00C61A58	
00C61A53	. 3945 1C	CMP DWORD PTR SS:[EBP+1C],EAX	CMP eax >= 5
00C61A56	> 76 04	JBE SHORT 00C61A5C	
00C61A58	> 8065 F3 00	AND BYTE PTR SS:[EBP-D],0	

فرض کنید که کلید این است: xxxxyyzzxxxxxxxxxxxxxxxxxxxxxxxx

می‌توان دید که هم ZZ باید ۰۵ باشد و هم YY باید بالاتر از ۰۵ باشد.

در مرحله بعد با چیزی شبیه به شکل زیر بر خورد می‌کنیم:

00C61A63	. 8BC3	MOV EAX,EBX	
00C61A65	> 50	PUSH EAX	
00C61A66	. 6A 0C	PUSH 0C	
00C61A68	. E8 DAF8FFFF	CALL 00C61347	
00C61A6D	. 3945 18	CMP DWORD PTR SS:[EBP+18],EAX	CMP eax == 5900
00C61A70	. 59	POP ECX	
00C61A71	. 59	POP ECX	
00C61A72	> 74 04	JE SHORT 00C61A78	
00C61A74	. 8065 F3 00	AND BYTE PTR SS:[EBP-D],0	

(word) [12]key بایستی با 0x5900 مساوی باشد. پس اکنون می‌دانیم که یک کلید تا الان می‌تواند

شبه این باشد:

xxxx1205 xxxxxxxxxxxxxxxxxxx0059xxxx

به عنوان آخرین مورد ولی نسبتاً با اهمیت می‌توانیم این شکل را ببینیم:

00C61A84	. 8BC3	MOV EAX,EBX	
00C61A86	> 50	PUSH EAX	
00C61A87	. 6A 0F	PUSH 0F	
00C61A89	. E8 B9F8FFFF	CALL 00C61347	
00C61A8E	. 59	POP ECX	
00C61A8F	. 59	POP ECX	
00C61A90	. 0FB74D 08	MOVZX ECX,WORD PTR SS:[EBP+8]	Get key[0] (word) && 0x1FFF
00C61A94	. 394D 14	CMP DWORD PTR SS:[EBP+14],ECX	
00C61A97	> 74 05	JE SHORT 00C61A9E	
00C61A99	. 394D 20	CMP DWORD PTR SS:[EBP+20],ECX	
00C61A9C	> EB 0E	JMP SHORT 00C61AAC	
00C61A9E	> 8445 24	TEST BYTE PTR SS:[EBP+24],AL	Test AL = 0x80
00C61AA1	> 75 0F	JNZ SHORT 00C61AB2	
00C61AA3	> EB 09	JMP SHORT 00C61AAE	
00C61AA5	> 0FB745 08	MOVZX EAX,WORD PTR SS:[EBP+8]	
00C61AA9	. 3945 14	CMP DWORD PTR SS:[EBP+14],EAX	
00C61AAC	> 74 04	JE SHORT 00C61AB2	
00C61AAE	> 8065 F3 00	AND BYTE PTR SS:[EBP-D],0	
00C61AB2	> 8A5D F3	MOV BL,BYTE PTR SS:[EBP-D]	Move GOOD BOY / BAD BOY to BL

در ابتدا (byte) [15]key را دریافت می‌کنیم. سپس چک می‌کنیم که آیا (word) [0]key هم با 0x1680

و هم با 0x1688 مساوی است یا خیر؟ اگر [0]key با هیچ یک از این مقادیر مساوی نباشد، پیغام خطا

ساخته می شود. شبیه همین برای key[15] نیز انجام می شود. اگر نه، هشتمین بیت تنظیم می شود. قبلا در کد، key[0] با 0x1FFF AND شده است. هم اکنون یک کلید داریم که به شکل زیر است.

88161205xxxxxxxxxxxxxxxx0059xx80

متاسفانه هنوز کار تمام نشده است و هنوز نیاز داریم که تعداد کمی از جاهای خالی را پر کنیم. برای پیدا کردن قطعات نداشته، یک ایده خوب، تنظیم یک بریک پوینت روی عمل مقایسه بایت به بایت خواهد بود. پنجره ثبت^۱ را ببندید و کلید علامت سوال را بزنید. اکنون دوباره در مقایسه بایت به بایت خواهید بود. اینک از طریق معتبرسازی کلید پیش رفته و در آخر به اینجا می رسید:

00C61E56	. 83C4 24	ADD ESP,24	
00C61E59	. 3AC3	CMP AL,BL	
00C61E5B	. 8845 F3	MOV BYTE PTR SS:[EBP-D],AL	
00C61E5E	√ 74 09	JE SHORT 00C61E69	
00C61E60	. 395D E8	CMP DWORD PTR SS:[EBP-18],EBX	
00C61E63	√ 0F85 3A010000	JNZ 00C61FA3	
00C61E69	> 53	PUSH EBX	
00C61E6A	. FF75 18	PUSH DWORD PTR SS:[EBP+18]	
00C61E6D	. FF75 14	PUSH DWORD PTR SS:[EBP+14]	
00C61E70	. FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
00C61E73	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00C61E76	. E8 A5740600	CALL 00CC9320	
00C61E7B	. 8A45 0F	MOV AL, BYTE PTR SS:[EBP+F]	
00C61E7E	. 83C4 14	ADD ESP,14	
00C61E81	. 8D4D C0	LEA ECX, DWORD PTR SS:[EBP-40]	
00C61E84	. 8845 C0	MOV BYTE PTR SS:[EBP-40],AL	
00C61E87	. 53	PUSH EBX	
00C61E88	. E8 CDC7FEFF	CALL 00C4E65A	
00C61E8D	. 68 041BE600	PUSH 00E61B04	UNICODE "UserID"
00C61E92	. E8 15181000	CALL 00D636AC	
00C61E97	. 59	POP ECX	
00C61E98	. 50	PUSH EAX	
00C61E99	. 68 041BE600	PUSH 00E61B04	UNICODE "UserID"
00C61E9E	. 8D4D C0	LEA ECX, DWORD PTR SS:[EBP-40]	
00C61EA1	. E8 A9CFEFFF	CALL 00C4EE4F	
00C61EA7	. 8845 C0	MOV BYTE PTR SS:[EBP-40],AL	

Shife+F9 را بزنید و شکل زیر را ببینید:

00C92499	. 83C4 28	ADD ESP,28	
00C9249C	. 8408	TEST AL,AL	
00C9249E	√ 75 3E	JNZ SHORT 00C924DE	
00C924A0	. 8A06 64010000	MOV AL, BYTE PTR DS:[ESI+164]	
00C924A6	. 50	PUSH EAX	
00C924A7	. 8D45 CC	LEA EAX, DWORD PTR SS:[EBP-34]	
00C924AA	. FFB6 60010000	PUSH DWORD PTR DS:[ESI+160]	
00C924B0	. FFB6 5C010000	PUSH DWORD PTR DS:[ESI+15C]	
00C924B6	. FFB6 58010000	PUSH DWORD PTR DS:[ESI+158]	
00C924BC	. FFB6 54010000	PUSH DWORD PTR DS:[ESI+154]	
00C924C2	. 53	PUSH EBX	
00C924C3	. 68 1810E600	PUSH 00E61B18	UNICODE "Software\ITEKSOFT\DocPrinter\PDF\5.0
00C924C8	. 50	PUSH EAX	
00C924C9	. 8D45 BC	LEA EAX, DWORD PTR SS:[EBP-44]	
00C924CC	. 50	PUSH EAX	
00C924CD	. 57	PUSH EDI	
00C924CE	. E8 DAF7FCFF	CALL 00C61C9D	
00C924D3	. 83C4 28	ADD ESP,28	
00C924D6	. 8408	TEST AL,AL	
00C924D8	√ 0F84 74060000	JE 00C92E52	
00C924DE	> 8A45 F2	MOV AL, BYTE PTR SS:[EBP-E]	
00C924E1	. 6A 00	PUSH 0	
00C924E3	. 8D4D 9C	LEA ECX, DWORD PTR SS:[EBP-64]	
00C924E6	. 8845 9C	MOV BYTE PTR SS:[EBP-64],AL	
00C924E9	. E8 6CC1FBFF	CALL 00C4E65A	
00C924EE	. BF 041BE600	MOV EDI, 00E61B04	UNICODE "UserID"
00C924F3	. 57	PUSH EDI	
00C924F4	. E8 B3110D00	CALL 00D636AC	
00C924F9	. 59	POP ECX	
00C924FB	. 50	PUSH EAX	

¹ registration