Mặc dù đã lock nhưng firmwave vẫn có thể được đọc ra. Vậy các cao thủ Nam - Bắc có cao kiến gì phòng chống vụ này không?

Xử lý bằng software của mình. Đọc một cái ID nào đó ở đâu đó rồi lên tục check trong vòng lặp chính  Nó hack chip rồi chắc lại phải Disasem để \*\*\*\*\* code 

*Nguyên văn bởi****BinhAnh****[Xem bài viết](http://www.dientuvietnam.net/forums/forum/vi-%C4%91i%E1%BB%81u-khi%E1%BB%83n-mcu-b%E1%BB%99-%C4%91i%E1%BB%81u-khi%E1%BB%83n-t%C3%ADn-hi%E1%BB%87u-s%E1%BB%91-dsc/ki%E1%BA%BFn-th%E1%BB%A9c-chung-v%E1%BB%81-chip-kh%E1%BA%A3-tr%C3%ACnh/4507-ph%C3%B2ng-ch%E1%BB%91ng-hack-firmwave" \l "post4507" \o "Xem bài viết)*

*Mặc dù đã lock nhưng firmwave vẫn có thể được đọc ra.*

Em thấy khi lock rồi mà đọc xuống thì chỉ nhận được toàn giá trị là 0x00 hay 0xFF. Nếu dùng cái firm đọc xuống này nạp vào con IC mới thì nó cũng không chạy.  
Các cao nhân có thể chỉ bảo cho em hiểu cách thức đặt chế độ lock khi nạp được không (cho các loại vxl 89x,PICx,AVRx,....).

|

Tui đưa thử 1 giải pháp :  
\* MCU phải luôn được cấp nguồn, tức là phải có battery.  
\* Power supply phải lọc thật tốt, tránh reset hoặc hang.  
\* MCU khi được cấp điện lần đầu (khi xuất xưởng) sẽ tiến hành đọc 1 đoạn code bí mật (lưu trong 1 con 24C04 chẳng hạn, hoặc EEPROM built-in), nếu đúng sẽ tiến hành xóa toàn bộ con 24C04 rồi chạy chương trình như bình thường.  
  
Như vậy nếu chép được firmware mà không dò được code thì vẫn không chạy !

*Tui đưa thử 1 giải pháp :  
\* MCU phải luôn được cấp nguồn, tức là phải có battery.  
\* Power supply phải lọc thật tốt, tránh reset hoặc hang.  
\* MCU khi được cấp điện lần đầu (khi xuất xưởng) sẽ tiến hành đọc 1 đoạn code bí mật (lưu trong 1 con 24C04 chẳng hạn, hoặc EEPROM built-in), nếu đúng sẽ tiến hành xóa toàn bộ con 24C04 rồi chạy chương trình như bình thường.  
  
Như vậy nếu chép được firmware mà không dò được code thì vẫn không chạy !*

Em không hiểu ý anh !  
  
Nếu so sánh code ID của MCU với một code ngoài nào đó phải không ạ, việc xóa code để việc thực hiện so sánh 1 lần và đảm bảo không có reset để không so sánh lần nữa !  
Nhưng theo em làm thế còn dễ bị hack hơn, nếu đã đọc được firmware thì tất nhiên trong đoạn chương trình của anh sẽ còn một đoạn:  
  
if(code đọc được==code định sẵn) {cho chạy chương trình}  
  
Thì nó sẽ toi ở lệnh này, chỉ cần cho if(1==1) sửa thế thôi, thì chạy ngon ngay.

Về lý thuyết thì \*\*\*\*\* code của VDK dễ hơn rất nhiều so với trên PC vì VDK mã lệnh ASM đơn giản và dễ đọc nên khả năng bị bẻ khóa sẽ rất cao. Các chương trình phần mềm có các đoạn mã chống \*\*\*\*\* thường là các đoạn mã giả nhằm đánh lạc hướng cracker. Nếu như firmware không quan trọng lắm thì chỉ cần bảo vệ đơn giản như KEY hay ID của CHIP. Nếu quan trọng hơn thì có thể gán mỗi firmware(khi nạp) bằng 1 cái KEY. Khi bắt đầu chạy thì bắn ra 1 cái Console đòi: ENTER PASSWORD: \*\*\*\*\*\*\*\*. Dùng thuật toán mã hóa KEY và mã giả khi check KEY thì cracker cũng đã rất rất khó để phá rồi. Thuê một thằng \*\*\*\*\* được nó thì cũng tốn vài K $

Vậy sau khi Password entered thì sao??? Lại:  
*if(Pass đọc được==Pass định sẵn) {cho chạy chương trình}* chăng. Vậy thì lại vô nghĩa rồi  
  
PT.

Nói chung cách chống hack đơn giản nhất là sử dụng các bảng tham chiếu . Sử dụng ramdom địa chỉ ánh xạ 2 chiều. Để khi người debug hack code thì cũng không thể tìm ra được chỗ chứa câu lệnh IF. Chúc mọi người giáng sinh vui vẻ!

*Vậy sau khi Password entered thì sao??? Lại:  
if(Pass đọc được==Pass định sẵn) {cho chạy chương trình} chăng. Vậy thì lại vô nghĩa rồi  
  
PT.*

thế khi bạn dissasembly ra thì có cái biến nào hiện lên là biến pass không?  
nếu như dễ như thế thì ai cũng có thể trở thành cracker nổi tiếng   
mấy cái đoạn mã giả của tôi nói chẳng lẽ khi dissasembly nó đều biến mất.  
  
thường các đoạn mã check pass sẽ nằm trong 1 lệnh call (cracking for newbie)  
...  
call 0x43738 <--- tính pass  
jnc 0x47643 <---- check nếu carry = 0 ---> false  
mov a,bcda  
lds xymqk  
...  
  
nếu trước lệnh call đặt 1 vài đoạn check giả, trong lệnh call đặt vài cái jump thẳng tới phương trời xa tít nứa. Rồi jump về sau lại call, rồi mov linh tinh sau rồi check jb, jnc, jnb .... linh tinh ----> debug một hồi nó cũng chán luôn.  
  
Vì thế đã có người ví: \*\*\*\*\* là một môn nghệ thuật, cũng như bao môn nghệ thuật khác nó đòi hỏi người chơi phải có năng khiếu, sự tìm tòi và cần mẫn ...

\*\*\*\*\* soft cả trên PC và MCU đều không có gì khó. Cái khó là đọc được firmware. Nếu đọc được thì mọi việc đơn giản.  
To Cuongquay. Trên PC lại càng dễ vì tool rất nhiều.  
MCU cũng không phải là khó.  
Trên dòng MPU như 386 hay cao hơn có khó hơn đôi chút vì nó dùng bảng tham chiếu chồtan bộ ứng dụng global và local. Nhưng kẽ hở lại là debug mode rất mạnh. Nếu bị khai thác tính năng này thì cũng toi ngay.  
Ví dụ về \*\*\*\*\* soft trên PC.  
(ví dụ đơn giản là bản demo cho thời gian thử 30 ngày).  
Bạn chỉ cần tìm hàm thời gian trong 1 DLL ( chạy SIM thấy ngay) Nếu lười thì sau nó là je hay j gì đó sử thành jmp là OK. Không tin bạn cứ thử xem.  
Với phần mềm có bảo mật cao hơn.  
Ví dụ: Dùng các thông số do window lưu trong visual mem ( trên đĩa cứng). Cái này chỉ kiểm tra ngày tháng cũng đã khó. Vì nó không lấy timer hệ thống mà lấy của WINDOWS. Do vậy cần hiểu cấu trúc mà WIN đã dùng để chuyển HD thành RAM ( tổ chức như RAM nhưng trên đĩa cứng).  
Còn trên MCU. Cũng không ai lại đi debug từ đầu cả. Tùy theo cách chống debug thì sẽ có cách debug tương ứng.  
Ví dụ: Nếu kiểm tra bằng pass đọc từ ngoại vi. Đầu tiên tìm tới các lệnh truy cập ngoại vi. Chọn lọc sẽ được lệnh hỏi pass.  
Lúc này cần xem nó xử lý ngay hay lưu vào đâu. Sau đó có cách tiếp theo.  
  
Nếu có firmware thì cái thằng định copy đó chắc chắn nó còn thêm tính năng vào được chứ không nói đến việc bẻ khóa.  
Dẫn chứng: Chúng tôi thêm tính năng gọi thẻ 1719 vào PAYPHONE của ASCOM mặc dù chỉ có file.bin. Sau khi \*\*\*\*\* thì ngoài tính năng cũ còn thêm tính năng mới mà nguyên bản không có. Như dùng được các loại thẻ mà mình sản xuất ra hay thêm tính năng mà khách hàng yêu cầu.  
  
Nói chung khóa chỉ phòng người ngay chứ không chống được kẻ gian.  
Làm sao không khóa mà kẻ gian vẫn chịu.  
1. Phải có thương hiệu khi đó copy vô ích.  
2. Giá thành hợp lý ( Một số sản phẩm telecom không hiểu sao china làm cực rẻ mà tốt. Đã hỏi thử giá linh kiện cho 1K sản phẩm thì tổng giá thành linh kiện chưa kể PCB, công hàn và test mà giá đã bằng 120% giá nó bán. Lúc đó copy vô ích).  
3. Chế độ hậu mãi tốt, quan hệ khách hàng tốt.  
  
Nói chung anh em mình lo làm sản phẩm thì tốt hơn lo bị copy.

*\*\*\*\*\* soft cả trên PC và MCU đều không có gì khó. Cái khó là đọc được firmware. Nếu đọc được thì mọi việc đơn giản.  
To Cuongquay. Trên PC lại càng dễ vì tool rất nhiều.  
MCU cũng không phải là khó.  
Trên dòng MPU như 386 hay cao hơn có khó hơn đôi chút vì nó dùng bảng tham chiếu chồtan bộ ứng dụng global và local. Nhưng kẽ hở lại là debug mode rất mạnh. Nếu bị khai thác tính năng này thì cũng toi ngay.  
Ví dụ về \*\*\*\*\* soft trên PC.*

\*\*\*\*\* trên PC đúng là có rất nhiều tool debug mạnh như W32DASM, OllyDbg, SoftIce nhưng nó chỉ hỗ trợ cho cracker tìm kiếm  
nhanh chóng các điểm gọi hàm, các string tham chiếu chứ nó không thể giúp cho crker biết đâu là điểm check pass chính xác.  
Nếu chúng ta viết 1 CT firmware cho VDK bằng C sau đó dịch ra và debug bằng ASM thì chúng ta sẽ dễ dàng hiểu được CT sử dụng  
những ngắt nào, trong vòng lặp chính sẽ gọi các call nào vì số lượng mã và các thủ tục là hữu hạn và ít.  
  
CT trên PC khi chạy nó load 1 đống DLL hệ thống, có những DLL chỉ dùng có 1 vài hàm. Windows sử dụng user32.dll và kernel32.dll  
là chính. 1 Chương trình trên PC bác có biết nó phải làm bao nhiêu thứ để tạo ra được 1 cái cửa sổ Dialog không? Khi C check nhiều  
điểm ngẫu nhiên thì phải patch đến bao giờ. Thường thì crker phải tìm được cái thủ tục key\_decode để tín ngược trở lại thành key\_gen  
để không phải ngồi patch quá nhiều điểm. Thế khi 1 soft sử dụng dongle làm khoấ thì key\_gen kiểu gì? ---> bắt buộc phả patch hoặc  
dongle\_emulator .... Đấy là chưa kể soft bảo vệ bằng giải thuật pack (Amadillo, ASPack, ASProtect): image của chương trình được mã hoá  
và giải mã runtime  nên debug một cái soft bị pack là vô nghĩa.  <---- cái này có thể áp dụng cho VDK sử dụng DES Bootloader

*Nguyên văn bởi****MinhHa****[Xem bài viết](http://www.dientuvietnam.net/forums/forum/vi-%C4%91i%E1%BB%81u-khi%E1%BB%83n-mcu-b%E1%BB%99-%C4%91i%E1%BB%81u-khi%E1%BB%83n-t%C3%ADn-hi%E1%BB%87u-s%E1%BB%91-dsc/ki%E1%BA%BFn-th%E1%BB%A9c-chung-v%E1%BB%81-chip-kh%E1%BA%A3-tr%C3%ACnh/4507-ph%C3%B2ng-ch%E1%BB%91ng-hack-firmwave?p=270011" \l "post270011" \o "Xem bài viết)*

*(ví dụ đơn giản là bản demo cho thời gian thử 30 ngày).  
Bạn chỉ cần tìm hàm thời gian trong 1 DLL ( chạy SIM thấy ngay) Nếu lười thì sau nó là je hay j gì đó sử thành jmp là OK. Không tin bạn cứ thử xem.  
Với phần mềm có bảo mật cao hơn.*

Thế bác có biết 1 soft trên windows sẽ sử dụng lời gọi đến các hàm get\_time bao nhiêu lần và trong những trường hợp nào không? Rất rất nhiều, các ngắt timer,  
các quá trình đồ hoạ refresh, .... Cái khó là phải tìm được đúng điểm check <--- đây chính là một nghệ thuật cũng như bao nghệ thuật khác nó cần sự nhạy cảm, tỉ mỉ  
và một chút năng khiếu. Chính vì thế em mới nói rằng \*\*\*\*\* soft trên PC khó hơn rất rất nhiều.

*Nguyên văn bởi****MinhHa****[Xem bài viết](http://www.dientuvietnam.net/forums/forum/vi-%C4%91i%E1%BB%81u-khi%E1%BB%83n-mcu-b%E1%BB%99-%C4%91i%E1%BB%81u-khi%E1%BB%83n-t%C3%ADn-hi%E1%BB%87u-s%E1%BB%91-dsc/ki%E1%BA%BFn-th%E1%BB%A9c-chung-v%E1%BB%81-chip-kh%E1%BA%A3-tr%C3%ACnh/4507-ph%C3%B2ng-ch%E1%BB%91ng-hack-firmwave?p=270011" \l "post270011" \o "Xem bài viết)*

*Ví dụ: Dùng các thông số do window lưu trong visual mem ( trên đĩa cứng). Cái này chỉ kiểm tra ngày tháng cũng đã khó. Vì nó không lấy timer hệ thống mà lấy của WINDOWS. Do vậy cần hiểu cấu trúc mà WIN đã dùng để chuyển HD thành RAM ( tổ chức như RAM nhưng trên đĩa cứng).  
Còn trên MCU. Cũng không ai lại đi debug từ đầu cả. Tùy theo cách chống debug thì sẽ có cách debug tương ứng.  
Ví dụ: Nếu kiểm tra bằng pass đọc từ ngoại vi. Đầu tiên tìm tới các lệnh truy cập ngoại vi. Chọn lọc sẽ được lệnh hỏi pass.  
Lúc này cần xem nó xử lý ngay hay lưu vào đâu. Sau đó có cách tiếp theo.  
  
Nếu có firmware thì cái thằng định copy đó chắc chắn nó còn thêm tính năng vào được chứ không nói đến việc bẻ khóa.  
Dẫn chứng: Chúng tôi thêm tính năng gọi thẻ 1719 vào PAYPHONE của ASCOM mặc dù chỉ có file.bin. Sau khi \*\*\*\*\* thì ngoài tính năng cũ còn thêm tính năng mới mà nguyên bản không có. Như dùng được các loại thẻ mà mình sản xuất ra hay thêm tính năng mà khách hàng yêu cầu.*

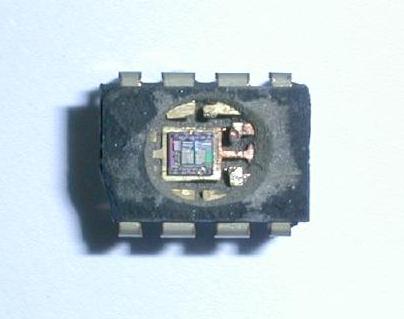
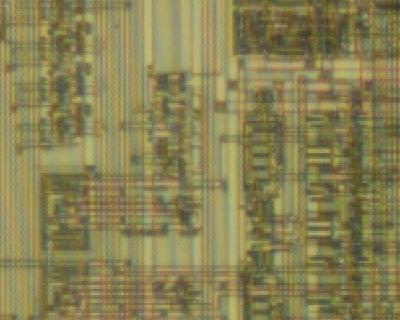
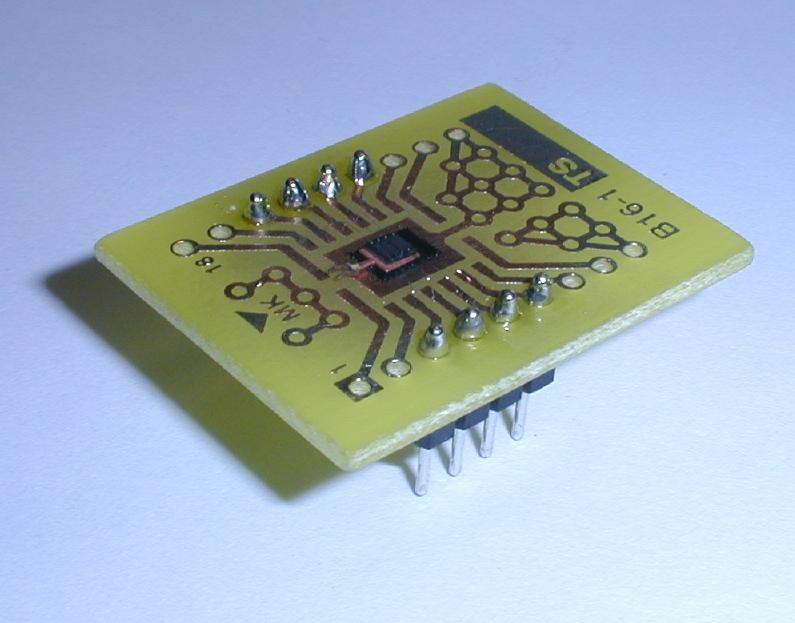
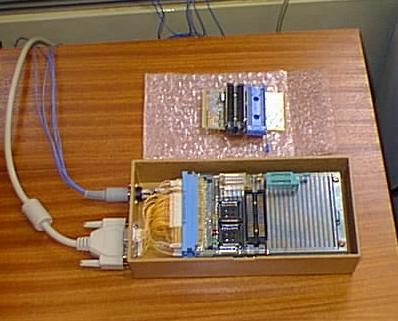
Tất cả những tư tưởng và ý đồ set break point của bác MH hoàn toàn chính xác nhưng đó chỉ là cái cơ bản nhất cần phải nghĩ đến khi đặt 1 break point. Vấn đề là phải phân tích xem đặt ở đâu sẽ có khả năng nhiều nhất. Các cracker kinh nghiệm khi đọc code nhiều sẽ có được  
cảm giác này (như cầu thủ cảm giác bóng khi đánh đầu  )

*Nguyên văn bởi****MinhHa****[Xem bài viết](http://www.dientuvietnam.net/forums/forum/vi-%C4%91i%E1%BB%81u-khi%E1%BB%83n-mcu-b%E1%BB%99-%C4%91i%E1%BB%81u-khi%E1%BB%83n-t%C3%ADn-hi%E1%BB%87u-s%E1%BB%91-dsc/ki%E1%BA%BFn-th%E1%BB%A9c-chung-v%E1%BB%81-chip-kh%E1%BA%A3-tr%C3%ACnh/4507-ph%C3%B2ng-ch%E1%BB%91ng-hack-firmwave?p=270011" \l "post270011" \o "Xem bài viết)*

*Nói chung khóa chỉ phòng người ngay chứ không chống được kẻ gian.  
Làm sao không khóa mà kẻ gian vẫn chịu.  
1. Phải có thương hiệu khi đó copy vô ích.  
2. Giá thành hợp lý ( Một số sản phẩm telecom không hiểu sao china làm cực rẻ mà tốt. Đã hỏi thử giá linh kiện cho 1K sản phẩm thì tổng giá thành linh kiện chưa kể PCB, công hàn và test mà giá đã bằng 120% giá nó bán. Lúc đó copy vô ích).  
3. Chế độ hậu mãi tốt, quan hệ khách hàng tốt.  
  
Nói chung anh em mình lo làm sản phẩm thì tốt hơn lo bị copy.*

Hehe, đúng là như thế!!! Sản phẩm của chúng ta đã ra gì chưa mà lo \*\*\*\*\*???? vài ngày lại phải update lỗi thì cần gì phải bảo vệ  nhưng vẫn phải lock bit.  
Tại sao thuật toán DES lại được coi là an toàn trong 1 trường hợp thông tin chỉ có giá trị trong vòng 72 giờ.  Sau 72 giờ thông tin không còn giá trị thì giải mã được cũng vô ích rồi

**Danh sách các CHIP đã bị hạ gục đến thời điểm năm 2004**  
  
**Motorola HC05**  
  
MC68HC05B6  
MC68HC05B8  
MC68HC05B16  
(MC68HC05B32)  
(MC68HC05X4)  
MC68HC05X16  
MC68HC05X32  
  
 **Motorola HC11**  
  
MC68HC11A8  
MC68HC11E9  
(MC68HC11E20)  
MC68HC11L6  
(MC68HC11KA2)  
(MC68HC11KA4)  
(MC68HC11KG2)  
(MC68HC11KG4)  
  
**Microchip PIC**  
  
PIC16C84  
  
**Microchip PIC**  
  
PIC16F83  
PIC16F84  
HCS512  
  
**Microchip PIC**  
  
PIC16F84A  
PIC16F627  
PIC16F628  
(PIC16F870)  
(PIC16F871)  
(PIC16F872)  
PIC16F873  
PIC16F874  
PIC16F876  
PIC16F877  
  
 **Atmel 8051**  
  
AT89C51  
AT89C52  
AT89C55  
AT89C1051  
AT89C2051  
(AT89C4051)  
  
**Atmel AVR**  
  
AT90S1200  
AT90S2313  
AT90S2323  
(AT90S2343)  
AT90S8515  
  
 **Texas Instruments MSP430**  
  
MSP430F110  
MSP430F112  
MSP430F1101  
MSP430F1121  
MSP430F122  
MSP430F123  
MSP430F133  
MSP430F135  
MSP430F147  
MSP430F148  
MSP430F149  
MSP430F412  
MSP430F413

Thông tin chi tiết xem ở đây <http://www.cl.cam.ac.uk/~sps32/mcu_lock.html>  
  
  
  
  


Còn đây là **how protection can be broken and how the security of new designs can be improved**  
<http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.html>

*Em không hiểu ý anh !  
  
Nếu so sánh code ID của MCU với một code ngoài nào đó phải không ạ, việc xóa code để việc thực hiện so sánh 1 lần và đảm bảo không có reset để không so sánh lần nữa !  
Nhưng theo em làm thế còn dễ bị hack hơn, nếu đã đọc được firmware thì tất nhiên trong đoạn chương trình của anh sẽ còn một đoạn:  
  
if(code đọc được==code định sẵn) {cho chạy chương trình}  
  
Thì nó sẽ toi ở lệnh này, chỉ cần cho if(1==1) sửa thế thôi, thì chạy ngon ngay.*

Đúng là nếu dùng lệnh kiểu CJNE A, #CODE, EXIT... thì rất dễ \*\*\*\*\*.  
Dùng kiểu JMP @A+DPTR (với biến A và/hoặc DPTR được nạp từ EEPROM lần đầu khi xuất xưởng) sẽ khó \*\*\*\*\* hơn. Có thể dùng nhiều lệnh như vậy để làm nản chí cracker !

vậy sau khi xuất xưởng thì sao nhỉ?  
cracker chuyên nghiệp thì không có chuyện nản chí. Đã làm thì làm bằng được. MCU thì như anh em trên diễn đàn viết đến 128KB chứ mấy.  
Đã có HEX file thì chống vô ích.  
Không nên chống mà nên phát triển nhanh. Nó copy được thì mình đã có cái mới. Tính năng nhiều hơn, giá cạnh tranh. Và ăn nhau ở uy tín với khách hàng chứ chống làm gì. Vừa tốn công vừa không có kết quả. Nhưng cũng nên lock bit để phòng người ngay.

Nói thẳng ra là cứ lock bit là OK rồi .  
Mấy ông VN mà có \*\*\*\*\* được thì cũng gọi là è cổ ra . Mà mất công nghệ vào những thằng giỏi như vậy thì chẳng có gì là đáng tiếc cả .   
Còn với những thằng có đầu óc thông minh tí tí nó xơi ý tưởng là chủ yếu rồi tự làm chứ hơi đâu .  
  
Cứ làm với giá bùn là khỏi thèm ăn cắp . ( đây là chiến thuật của mấy ông "trung của " đây .  )