Análisis de riesgos y recomendaciones de diseño electrónico con Arduino

Risk analysis and recommendations for electronic design with Arduino

María-Fernanda Yépez, Sergio Martin, Gabriel Díaz, Manuel Castro

Universidad Nacional de Educación a Distancia (España)

DOI: http://dx.doi.org/10.6036/8587

El hardware abierto, y en concreto uno de sus principales exponentes, las placas Arduino, permiten que cualquier persona interesada en el tema pueda investigar, crear sus propias placas y contribuir a sus mejoras, ya que está a disposición todo tipo de información con respecto a su diseño (circuitos, componentes, distribución, etc.). Además al ser dispositivos prácticamente sencillos de utilizar, se los emplea en una gran cantidad de aplicaciones de domótica, arte digital, entretenimiento, etc. Sin embargo pese a sus ventajas no la hacen fiable para ser utilizada en infraestructuras críticas [1].

ANÁLISIS

A. Vulnerabilidades en el hardware

Las placas Arduino no disponen de ningún circuito de protección contra sobretensiones y sobre-corrientes. Existen acciones (por descuido) o ataques (malintencionadamente) al **Hardware** relacionadas con aplicar voltajes inesperados en los pines de E/S cuyo resultado es completamente nefasto para el equipo basado en una placa Arduino [2].

Además existen técnicas invasivas para acceso no autorizado al código dentro del microcontrolador (**microprobing**), que consiste en acceder a los pines del chip directamente de manera que se pueda observar, manipular e interferir el circuito integrado. Otras técnicas menos invasivas se centran en observar las señales de alimentación y de reloj. [3].

B. Vulnerabilidades en el firmware

Si se tiene acceso, como se ha comentado antes, a la placa Arduino, es posible extraer el programa almacenado e incluso llegar a hacer algo de ingeniería inversa, para el caso de los sistemas Arduino, se necesita un programador ISP (**in-system programmer**). Una vez que se tiene acceso al programa almacenado en la **Flash**, se puede intentar convertir el código binario a algún formato legible utilizando algún decompilador [4].

C. Vulnerabilidades en las comunicaciones

Las comunicaciones inalámbricas son susceptibles de ser interferidas si no están cifradas. Por otro lado, al abrir puertos de comunicaciones pueden sufrir ataques, como el de denegación de servicio que conlleva su total pérdida de conexión y pérdida de servicios de red (Figura 1).

RECOMENDACIONES

Desde el punto de vista **Hardware** algunas medidas a tomar serían por ejemplo:

- No dejar ninguna interfaz de acceso al dispositivo a la vista, dotar al contenedor de mecanismos anti-manipulación, como por ejemplo el uso de tornillos unidireccionales [5].
- Sellar la caja de manera que la ruptura de ese sello implicaría automáticamente la anulación de la garantía.
- Incluir sensores que detecten un cambio operacional o medioambiental o circuitos que puedan detectar un pinchazo, rotura o intento de modificación en la caja del dispositivo y que actúen en consecuencia, generando una señal de alarma o bien interrumpiendo el normal funcionamiento del equipo [5].
- Recubrir los circuitos internos que forman parte de la placa Arduino con resina epoxi [5], de manera que su extracción de la placa sea lo más complicada posible.
- No dejar ninguna posibilidad de acceso a los pines del microcontrolador.

Desde el punto de vista del **Firmware** la medida más elemental a tomar es que la protección contra copia del programa residente en el microcontrolador debe haber sido seteada, de manera que no sea posible la transferencia del código binario hacia otro dispositivo en el que pudiera

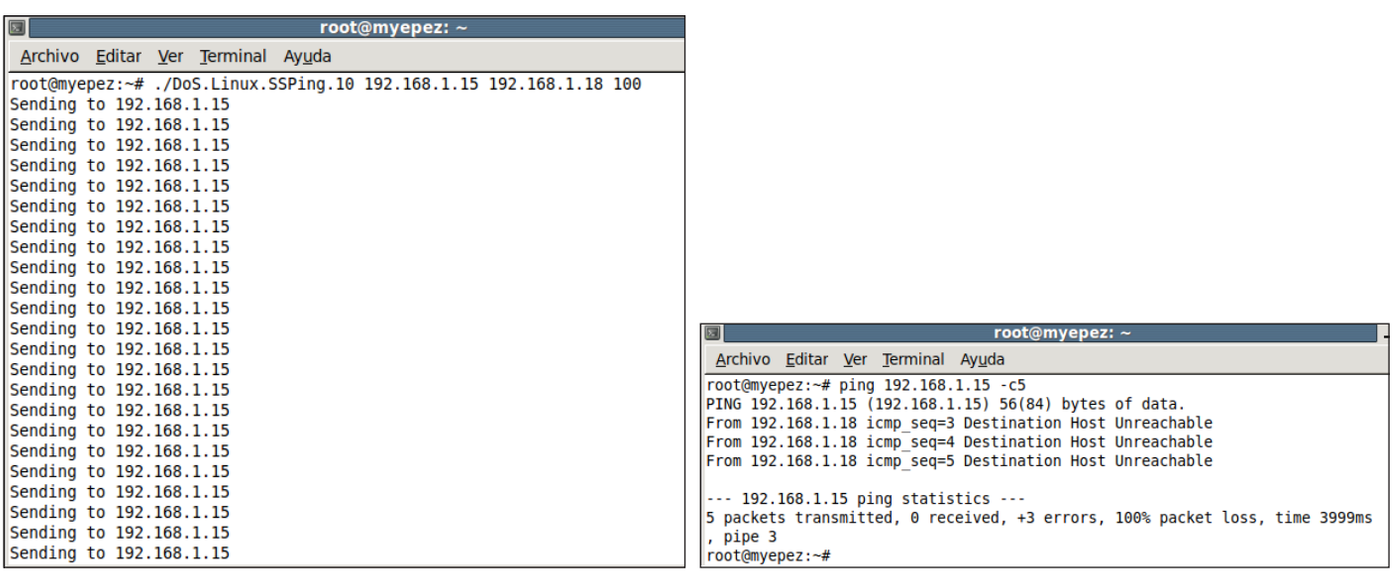

Figura 1. Ataque DoS a placa Arduino MEGA (izquierda) y resultado ataque (derecha)

desensamblarse y también evitar que pueda ser reprogramado. Sería recomendable también que cualquier tipo de información sensible como claves se almacenen no como variables locales del programa sino por ejemplo en elementos de memoria ROM externos y seguros.

Desde el punto de vista de los ataques de comunicaciones, las interfaces inalámbricas con las que se ha hecho diferentes pruebas en este trabajo, muestran la debilidad de los equipos Arduino. En este sentido, lo que se puede recomendar es realizar transmisión de datos utilizando al menos criptografía de clave simétrica, dado que es imposible la implementación de criptografía de clave asimétrica, ya que las características computacionales son muy pobres en cuanto a procesamiento y capacidad de memoria en estos equipos.

REFERENCIAS

[1] YEPEZ-BONILLA, Maria Fernanda, MARTIN-GUTIERREZ, Sergio, DIAZ-ORUETA, Gabriel et al. GOOD PRACTICES TO AVOID VULNERABILITY IN THE DESIGN WITH ARDUINO-LIKE OPEN HARDWARE PLATFORMS. DYNA New Technologies, Enero-Diciembre 2017, vol. 4, no. 1, [7 p.]. DOI: http://dx.doi.org/10.6036/NT8426

[2] Rugged Circuits. Industrial Strength, Hobbyist price. URL: http://www.ruggedcircuits.com/10-ways-to-destroy-an-Arduino/ [Consulta: 5/5/2015]

[3] Skorobogarov, S. (2000). Copy protection in modern microcontrollers. URL: http://www.cl.cam.ac.uk/~sps32/mcu_lock.html [Consulta: 5/5/2015]

[4] Todo sobre seguridad en internet. (2015). URL: http://www.viruslist.com/sp/viruses/encyclopedia?virusid=54204 [Consulta: 5/5/2015]

[5] GRAND, J. (2004). Understanding Hardware Security. Grand Idea Studio, Inc. Black Hat Japan 2004 Briefings.

AGRADECIMIENTOS

ETSI Industriales de la UNED por la ayuda 2017-IEQ13, así como los proyectos eMadrid (S2013/ICE-2715), IoE-EQ (2017-1-IT01-KA202-006251), e IoT4SMEs (nº 2016-1-IT01-KA202-005561).